# Chương 1

# CÁC KIẾN THỨC CƠ SỞ

## 1.1. MÃ HÓA

### 1.1.1. Lịch sử mã hóa

Lịch sử mã hóa kéo dài hàng nghìn năm, với sự phát triển từ những hệ thống đơn giản đến các thuật toán phức tạp được sử dụng ngày nay. Mã hóa không chỉ là một công cụ bảo vệ thông tin mà còn là nền tảng của sự an toàn và bảo mật trong thế giới kỹ thuật số.

#### *1.1.1.1. Thời kỳ cổ đại: Bắt đầu của mã hóa*

Trong giai đoạn này, các hệ thống mã hóa đơn giản được tạo ra nhằm mục đích bảo mật thông tin trong các cuộc chiến và giao tiếp quân sự.

Mật mã Atbash (khoảng thế kỷ VI TCN): Sử dụng trong văn bản Hebrew cổ đại, Atbash là một dạng mã hóa hoán vị, trong đó chữ cái đầu tiên bị thay thế bởi chữ cái cuối cùng, chữ cái thứ hai bị thay thế bởi chữ cái áp chót, và tiếp tục như vậy.

Mật mã Polybius (thế kỷ II TCN): Phát minh bởi nhà triết học Hy Lạp Polybius, mật mã này sử dụng một bảng vuông 5x5 để chuyển các chữ cái thành các tọa độ số.

Mật mã Caesar (khoảng năm 58-50 TCN): Julius Caesar sử dụng phương pháp mã hóa thay thế đơn giản, được gọi là Mã Caesar, để mã hóa các thông điệp quân sự. Đây là dạng mã hóa dịch chuyển trong bảng chữ cái với số lần dịch chuyển cố định. Ví dụ, với dịch chuyển 3 vị trí, "A" trở thành "D".

#### *1.1.1.2. Thời Trung đại và Phục hưng: Phát triển mã hóa đa dạng*

Trong thời kỳ này, mã hóa bắt đầu trở nên phức tạp hơn, với sự xuất hiện của các hệ thống mã hóa đa bảng và mã hóa cặp chữ.

Mật mã Vigenère (1553): Blaise de Vigenère, một nhà ngoại giao người Pháp, đã phát triển hệ thống mã hóa đa bảng. Mã hóa này sử dụng một từ khóa để điều chỉnh việc dịch chuyển ký tự trong thông điệp. Với phương pháp này, mỗi chữ cái trong văn bản gốc được mã hóa bằng một bảng dịch chuyển khác nhau, làm cho việc giải mã mà không có khóa trở nên rất khó khăn.

Mật mã Playfair (1854): Charles Wheatstone phát triển hệ thống mã hóa này, trong đó các cặp chữ cái trong văn bản gốc được mã hóa thay vì từng chữ cái đơn lẻ, điều này giúp gia tăng độ phức tạp và bảo mật cho thông tin.

Mật mã Morse (1837): một phương pháp được sử dụng trong viễn thông để mã hóa văn bản ký tự như trình tự chuẩn của hai khoảng thời gian tín hiệu khác nhau, được gọi là *dấu chấm (dots)* và *dấu gạch ngang (dash).*

#### *1.1.1.3. Thế kỷ 20 và hai cuộc Chiến tranh Thế giới: Bước tiến vượt bậc trong mã hóa*

Với sự phát triển của công nghệ và máy tính, mã hóa trở thành một yếu tố quan trọng trong chiến tranh và các hệ thống bảo mật quốc gia.

Mã hoá Enigma (1920s): Máy Enigma được Đức quốc xã sử dụng trong Chiến tranh Thế giới II để mã hóa thông tin quân sự. Mỗi thông điệp được mã hóa bằng cách sử dụng các rô-to xoay có thể thay đổi hàng triệu cấu hình khác nhau. Nhóm của Alan Turing tại Bletchley Park (Anh) đã phát triển các phương pháp giải mã thành công hệ thống mã hóa này, đóng góp to lớn vào kết quả của chiến tranh. Bẻ khóa mã Enigma giúp rút ngắn chiến tranh khoảng 2 năm và cứu sống hàng triệu người.

Claude Shannon (1949): Nhà toán học người Mỹ Claude Shannon, trong nghiên cứu "A Mathematical Theory of Cryptography" đã đưa ra định nghĩa mã hóa theo các nguyên tắc toán học. Nghiên cứu của ông trở thành nền tảng cho lý thuyết mật mã hiện đại.

Mã hóa Huffman (1952) là một thuật toán mã hóa dùng để nén dữ liệu. Nó dựa trên bảng tần suất xuất hiện các ký tự cần mã hóa để xây dựng một bộ mã nhị phân là mã tiền tố cho các ký tự đó sao cho dung lượng (số bit) sau khi mã hóa là nhỏ nhất

#### *1.1.1.4. Sự ra đời của máy tính và mật mã hiện đại (1970 - nay)*

Với sự phát triển của máy tính, các hệ thống mã hóa ngày càng trở nên phức tạp và tinh vi, từ mã hóa đối xứng đến mã hóa bất đối xứng.

Mã hoá DES (1977): Data Encryption Standard (DES) được phát triển bởi IBM và được phê duyệt bởi NIST (Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ). Đây là tiêu chuẩn mã hóa đối xứng với khóa 56-bit, từng được coi là không thể phá vỡ. DES sử dụng khóa 56-bit, tạo ra 72,057,594,037,927,936 (2^56) khả năng khóa.

Mã hoá RSA (1977): Rivest, Shamir và Adleman phát triển thuật toán mã hóa bất đối xứng RSA, sử dụng khóa công khai và khóa riêng. Đây là nền tảng của mã hóa công khai và được sử dụng rộng rãi trong giao dịch bảo mật. RSA sử dụng các số nguyên tố rất lớn, lên đến hàng trăm bit. Hiện nay, khóa 2048-bit RSA được sử dụng phổ biến cho bảo mật cao.

Mã hoá AES (2001): Advanced Encryption Standard (AES) được phát triển để thay thế DES. AES sử dụng các kích thước khóa 128, 192, và 256-bit. Đây là một trong những phương pháp mã hóa nhanh, an toàn và hiệu quả nhất hiện nay.AES 256-bit có

2^256 khả năng khóa, tương đương với một con số gần như không thể bẻ khóa bằng các phương tiện tính toán hiện đại.

#### *1.1.1.5. Sự phát triển của mật mã lượng tử và blockchain (Thế kỷ 21)*

Mã hóa không chỉ phát triển trong không gian kỹ thuật số truyền thống mà còn mở rộng sang các lĩnh vực mới như máy tính lượng tử và công nghệ blockchain.

Shor's Algorithm (1994): Peter Shor phát triển một thuật toán lượng tử có thể phân tích các số nguyên tố rất nhanh, đe dọa tính bảo mật của các hệ thống mã hóa dựa trên RSA[1] và ECC. Một máy tính lượng tử sử dụng thuật toán Shor có thể bẻ khóa RSA 2048-bit chỉ trong vài giờ, trong khi các siêu máy tính hiện nay phải mất hàng triệu năm.

Blockchain và Bitcoin (2009): Bitcoin và blockchain sử dụng các khái niệm mã hóa tiên tiến, bao gồm các hàm băm mật mã và chữ ký số để bảo mật các giao dịch. Đây là một trong những ứng dụng lớn nhất của mật mã trong hệ thống tài chính. Hàm băm SHA-256, được sử dụng trong blockchain Bitcoin, tạo ra 2^256 kết quả khả dĩ.

### 1.1.2. Khái niệm mã hóa và giải mã

Mã hóa là quá trình chuyển đổi thông tin từ dạng dễ đọc (plaintext) sang dạng không thể đọc được nếu không có công cụ hoặc khóa giải mã (ciphertext). Mục tiêu của mã hóa là bảo vệ thông tin nhạy cảm, đảm bảo rằng chỉ những người có quyền truy cập mới có thể đọc và hiểu được thông tin.

Giải mã là quá trình ngược lại, tức là chuyển đổi thông tin từ dạng mã hóa (ciphertext) về lại dạng nguyên bản (plaintext). Quá trình này chỉ có thể được thực hiện nếu biết chính xác khóa giải mã hoặc thuật toán tương ứng.

Mã hóa dựa vào hai thành phần cơ bản:

Thuật toán mã hóa (Cipher): Đây là phương thức hoặc tập hợp các quy tắc dùng để chuyển đổi từ bản rõ sang bản mã. Một số thuật toán phổ biến bao gồm AES, RSA, và DES.

Khóa mã hóa (Key): Đây là giá trị đặc biệt mà cả quá trình mã hóa và giải mã dựa vào. Nếu không có khóa này, quá trình giải mã không thể thực hiện.

Có 2 loại mã hóa là mã hóa đối xứng và mã hóa bất đối xứng:

Mã hóa đối xứng (Symmetric Encryption): Cả mã hóa và giải mã sử dụng cùng một khóa bí mật. Thuật toán này nhanh hơn nhưng yêu cầu phải bảo mật khóa khi chia sẻ với người nhận (ví dụ: AES, DES).

[1] RSA, ECC: xxxx

Mã hóa bất đối xứng (Asymmetric Encryption): Sử dụng một cặp khóa, bao gồm khóa công khai để mã hóa và khóa riêng tư để giải mã. Phương pháp này an toàn hơn nhưng chậm hơn (ví dụ: RSA, ECC).

Mã hóa có nhiều ứng ụng trong thực tế

Bảo mật thông tin cá nhân và tài chính: Mã hóa được sử dụng rộng rãi trong các giao dịch ngân hàng và thương mại điện tử để bảo vệ dữ liệu thẻ tín dụng và thông tin cá nhân.

Bảo mật email và tin nhắn: Công nghệ mã hóa như PGP (Pretty Good Privacy) được sử dụng để bảo vệ nội dung email khỏi việc bị đọc trộm.

Mạng riêng ảo (VPN): Sử dụng mã hóa để bảo vệ các kết nối mạng khỏi tin tặc hoặc các gián điệp mạng.

### 1.1.3. Mã hóa cổ điển

Mã hóa cổ điển (Classical Cryptography) là lĩnh vực mã hóa xuất hiện trước khi máy tính ra đời. Các kỹ thuật mã hóa này dựa trên sự thay thế hoặc hoán vị ký tự và chủ yếu phục vụ cho các mục đích quân sự, chính trị, và ngoại giao. Phần này trình bày các kiến thức quan trọng về mã hóa cổ điển, bao gồm những phương pháp nổi tiếng và các khái niệm cơ bản.

#### *1.1.3.1 Mã hóa thay thế (Substitution Cipher)*

Mã hóa thay thế là kỹ thuật mã hóa cổ điển trong đó mỗi ký tự trong văn bản gốc (plaintext) được thay thế bằng một ký tự khác. Các loại mã hóa thay thế phổ biến bao gồm:

***Mã Caesar (Caesar Cipher):*** Một trong những phương pháp mã hóa lâu đời nhất, được Julius Caesar sử dụng để mã hóa các thông điệp quân sự. Mã Caesar là một loại mã hóa thay thế trong đó mỗi ký tự trong văn bản gốc được dịch chuyển một số vị trí cố định trong bảng chữ cái.

Nếu p là ký tự gốc và p là số vị trí dịch chuyển, thì ký tự mã hóa c được tính theo công thức: $c = (p+k) \bmod 26$. Để giải mã: $p = (c-k) \bmod 26$.

Ví dụ: Với dịch chuyển 3, chữ "A" trở thành "D", "B" trở thành "E", và tiếp tục như vậy.

***Mật mã Atbash (Atbash Cipher):*** Đây là một dạng mã hóa thay thế ngược, trong đó chữ cái đầu tiên của bảng chữ cái bị thay thế bởi chữ cái cuối cùng, chữ cái thứ hai bị thay thế bởi chữ cái áp chót, và tiếp tục như vậy.

Ví dụ: "A" sẽ trở thành "Z", "B" sẽ trở thành "Y", và "C" sẽ trở thành "X".

***Mật mã đa bảng (Polyalphabetic Cipher):*** Mã hóa này sử dụng nhiều bảng thay thế khác nhau để mã hóa các ký tự ở các vị trí khác nhau. Mỗi ký tự trong văn bản gốc được mã hóa theo một bảng khác nhau, do đó việc bẻ khóa trở nên khó khăn hơn.

### *1.1.3.2 Mã hóa hoán vị (Transposition Cipher)*

Mã hóa hoán vị thay đổi thứ tự của các ký tự trong văn bản gốc thay vì thay thế chúng bằng các ký tự khác. Kỹ thuật này thường được sử dụng kết hợp với mã hóa thay thế để tăng độ phức tạp.

***Mã hóa Scytale:*** Scytale là một phương pháp mã hóa của người Spartan, sử dụng một cây gậy (scytale) để viết các thông điệp trên một dải giấy quấn quanh cây gậy. Khi giấy được tháo ra, văn bản trở thành không thể đọc được trừ khi nó được quấn lại trên một cây gậy có cùng đường kính.

***Mã hóa cột (Columnar Transposition):*** Trong mã hóa cột, văn bản gốc được viết thành các hàng của một bảng với số cột xác định trước. Sau đó, các ký tự được đọc theo thứ tự từ trên xuống dưới qua các cột thay vì từ trái sang phải.

Ví dụ: Văn bản gốc “HELLOWORLD” với khoá là HACK

Dữ liệu được biểu diễn theo bảng 4 cột theo độ dài của HACK

| H | A | C | K |
|---|---|---|---|
| H | E | L | L |
| O | W | O | R |
| L | D | | |

Sắp xếp các cột lại theo bảng chữ cái của khoá ta được

| A | C | H | K |
|---|---|---|---|
| E | L | H | L |
| W | O | O | R |
| D | | L | |

Bây giờ ta lấy dữ liệu theo từng cột được kết quả mã hoá “EWDLOHOLLR”

### *1.1.3.3 Mã hóa Vigenère (Vigenère Cipher)*

Đây là một dạng mã hóa đa bảng, sử dụng một từ khóa để quyết định cách mã hóa từng ký tự trong văn bản gốc. Mỗi chữ cái trong từ khóa được sử dụng để xác định bảng dịch chuyển cho các ký tự tương ứng trong văn bản gốc. Điều này làm cho mã hóa Vigenère trở nên khó bẻ khóa hơn nhiều so với mã hóa thay thế đơn giản.

Nếu ký tự thứ i trong văn bản gốc là $p_i$ và ký tự thứ i trong từ khóa là $k_i$, thì ký tự mã hóa $c_i$ được tính như sau: $c_i = (p_i + k_i) \bmod 26$. Để giải mã: $p_i = (c_i - k_i) \bmod 26$

Ví dụ: Với văn bản gốc "ATTACKATDAWN" với khóa "LEMON":

Plaintext: A T T A C K A T D A W N

Key: L E M O N L E M O N L E

Ciphertext: L X F O P V E F R N H R

### *1.1.3.4. Mã hóa Playfair (Playfair Cipher)*

Phát minh bởi Charles Wheatstone năm 1854, mã hóa Playfair là một hệ thống mã hóa cặp ký tự. Mỗi cặp chữ cái được mã hóa dựa trên một bảng vuông 5x5 chứa các chữ cái trong bảng chữ cái (loại bỏ chữ Q hoặc I/J được dùng chung).

Quy tắc mã hóa: Với một cặp chữ cái:

Nếu hai chữ cái nằm trên cùng một hàng, thay thế mỗi chữ bằng chữ bên phải nó.

Nếu hai chữ cái nằm trên cùng một cột, thay thế mỗi chữ bằng chữ bên dưới nó.

Nếu hai chữ cái tạo thành một hình chữ nhật, thay thế mỗi chữ bằng chữ nằm cùng hàng ở góc còn lại của hình chữ nhật đó.

Ví dụ:

Bảng mã hóa Playfair (không chứa "Q"):

P L A Y F

I/J B C D E

G H K M N

O R S T U

V W X Y Z

Mã hoá cặp “HI” nằm khác hàng khác cột nên ta lấy theo góc hình chữ nhật cùng hàng “H” thay thế bằng “G” còn “I” thay thế bằng “B”

Mã hoá cặp “SC” nằm cùng cột nên lấy ký tự dịch xuống 1 hàng “S” thành “X” còn “C” thành “K”.

Mã hoá cặp “EB” nằm cùng hàng nên lấy dịch sang phải 1 hàng “E” thành “I/J” còn “B“ thành “C”.

### *1.1.3.5 Mã hóa Hill (Hill Cipher)*

Mã hóa Hill được phát minh bởi Lester Hill vào năm 1929, sử dụng toán học tuyến tính và đại số ma trận để mã hóa các khối ký tự. Phương pháp này sử dụng ma trận vuông để mã hóa các ký tự trong văn bản gốc.

Công thức: Cho một khối ký tự có n ký tự, mã hóa Hill sử dụng một ma trận n×n khóa để thực hiện phép nhân ma trận với khối văn bản gốc. Mỗi chữ cái được chuyển thành một số từ 0 đến 25 (A=0, B=1, C=2, D=3, E=4, F=5, G=6, H = 7 ... … Z=25), sau đó áp dụng phép nhân ma trận để tính toán văn bản mã hóa.

Giải mã được thực hiện bằng cách nhân với ma trận nghịch đảo của ma trận khóa.

Ví dụ: Với văn bản gốc "HOME" và ma trận khóa 2x2 là:

$$(3\ 3\ 2\ 5)$$

Mã hóa Hill sẽ chuyển "HOME" thành chuỗi số “HO” (7, 14) và “ME” (10, 4) nhân modulo 26 với ma trận khóa và sau đó chuyển kết quả thành văn bản mã hóa

Với “HO” xét $(3\ 3\ 2\ 5) * (7\ 14) = (63\ \ 84)\ mod\ 26 = (11\ \ 6)$ tương ứng là “LG”

Với “ME” xét $(3\ 3\ 2\ 5) * (10\ 4) = (42\ 40)\ mod\ 26 = (16\ \ 14)$ tương ứng là “QO”

Vậy “HOME” sẽ mã hoá thành “LGQO”.

#### *1.1.3.6. Mã hóa Affine (Affine Cipher)*

Đây là dạng mã hóa thay thế dựa trên phép biến đổi affine trong toán học. Mỗi ký tự trong văn bản gốc được mã hóa bằng một hàm tuyến tính với dạng: $c=(a \cdot p+b) \bmod 26$.

Trong đó: p là vị trí của ký tự trong bảng chữ cái, a và b là các tham số của hàm tuyến tính (với a phải nguyên tố cùng nhau với 26) c là ký tự mã hóa.

Mã hóa cổ điển là nền tảng quan trọng của lĩnh vực mật mã. Mặc dù các phương pháp này hiện nay không đủ an toàn trước những công nghệ giải mã hiện đại, nhưng chúng vẫn có giá trị lịch sử lớn. Các hệ thống mã hóa cổ điển là bước đệm để phát triển các kỹ thuật mã hóa phức tạp hơn, như mã hóa đối xứng và bất đối xứng, được sử dụng trong thế giới kỹ thuật số ngày nay.

### 1.1.4. Mã hóa đối xứng (Symmetric Encryption)

Mã hóa đối xứng là một trong những phương pháp mã hóa cơ bản và lâu đời nhất, trong đó cùng một khóa bí mật được sử dụng để mã hóa và giải mã dữ liệu. Khóa này phải được giữ bí mật giữa hai bên giao tiếp để bảo đảm an toàn. Mã hóa đối xứng thường

được sử dụng cho việc truyền tải dữ liệu nhanh chóng do tốc độ cao của các thuật toán. Đặc điểm chính của mã hóa đối xứng:

*Sử dụng một khóa bí mật chung*: Cả hai bên (người gửi và người nhận) phải chia sẻ chung một khóa bí mật.

*Tốc độ cao*: Các thuật toán mã hóa đối xứng thường nhanh hơn các thuật toán mã hóa bất đối xứng.

*Vấn đề phân phối khóa*: Việc bảo mật và phân phối khóa giữa các bên là một thách thức lớn trong mã hóa đối xứng.

Mã hóa đối xứng thường được sử dụng trong các hệ thống cần tốc độ cao và khối lượng dữ liệu lớn, chẳng hạn như mã hóa dữ liệu đĩa, giao thức bảo mật mạng (VPN, SSL/TLS).

*1.1.4.1. Một số mật mã đối xứng*

***a. Data Encryption Standard (DES)***

DES được phát triển bởi IBM vào những năm 1970 và được NIST (Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ) phê duyệt làm tiêu chuẩn mã hóa vào năm 1977. DES sử dụng khóa 56-bit (thực tế khóa là 64-bit, nhưng 8 bit được sử dụng để kiểm tra tính toàn vẹn). DES là một khối mã hóa (block cipher) với kích thước khối 64-bit, sử dụng thuật toán mã hóa khối với chuỗi các phép biến đổi (Feistel network). Với sự gia tăng sức mạnh tính toán, khóa 56-bit của DES trở nên không an toàn. DES đã bị tấn công thành công nhiều lần thông qua các cuộc tấn công vét cạn (brute force).

Ví dụ: Giả sử bạn muốn mã hóa thông điệp "HELLO", DES sẽ chia văn bản thành các khối 64-bit và mã hóa từng khối bằng khóa 56-bit.

***b. Triple DES (3DES)***

Triple DES được phát triển để tăng cường độ bảo mật của DES bằng cách áp dụng thuật toán DES ba lần với hai hoặc ba khóa khác nhau. 3DES sử dụng khóa khóa 112-bit (sử dụng hai khóa) hoặc 168-bit (sử dụng ba khóa). 3DES mã hóa dữ liệu bằng cách sử dụng ba lần DES: mã hóa, giải mã, và mã hóa lần nữa (Encrypt-Decrypt-Encrypt). Mặc dù 3DES cải thiện bảo mật, nhưng nó vẫn bị coi là chậm và có độ dài khóa ngắn so với các thuật toán hiện đại.

Ví dụ: Thay vì mã hóa "HELLO" một lần với DES, 3DES sẽ thực hiện ba lần mã hóa với các khóa khác nhau để tăng cường tính an toàn.

***c. Advanced Encryption Standard (AES)***

AES được phát triển vào cuối những năm 1990 để thay thế DES và 3DES, và được phê chuẩn bởi NIST vào năm 2001 sau một cuộc thi toàn cầu tìm kiếm một thuật toán mã hóa an toàn và hiệu quả. Thuật toán này hỗ trợ ba độ dài khóa: 128-bit, 192-bit, và 256-bit. AES là thuật toán mã hóa khối với kích thước khối cố định 128-bit, sử dụng mạng biến đổi thay thế (Substitution-Permutation Network). AES có ưu điểm an toàn, nhanh chóng và được sử dụng rộng rãi trong các ứng dụng bảo mật hiện đại như SSL/TLS, WPA2 (Wi-Fi), và mã hóa dữ liệu đĩa.

Ví dụ: Mã hóa văn bản "HELLO WORLD" bằng AES với khóa 128-bit: Chia văn bản thành các khối: AES yêu cầu đầu vào là các khối 128-bit, vì vậy văn bản "HELLO WORLD" sẽ được chuyển đổi thành các khối nhị phân, sau đó được chia thành các khối phù hợp (bao gồm cả việc thêm padding nếu cần).

AES sẽ sử dụng khóa 128-bit để mã hóa từng khối, trải qua 10 vòng biến đổi bao gồm các bước thay thế byte (SubBytes), hoán vị hàng (ShiftRows), trộn cột (MixColumns), và thêm khóa (AddRoundKey).

Kết quả: Kết quả là một chuỗi dữ liệu nhị phân được mã hóa mà chỉ có thể giải mã bằng khóa bí mật 128-bit ban đầu.

***d. Blowfish***

Blowfish được Bruce Schneier phát triển vào năm 1993 như một sự thay thế miễn phí cho DES. Khóa của thuật toán này có thể thay đổi từ 32-bit đến 448-bit, cho phép tính linh hoạt về độ bảo mật. Blowfish là mã hóa khối với kích thước khối 64-bit, sử dụng mạng Feistel với 16 vòng lặp. Blowfish rất nhanh trong các hệ thống 32-bit, có thể thay đổi độ dài khóa, và được sử dụng rộng rãi trong các ứng dụng như bảo mật mật khẩu.

Ví dụ: Bạn có thể sử dụng khóa 128-bit để mã hóa "HELLO" thành khối 64-bit.

***e. Twofish***

Twofish được phát triển bởi các nhà thiết kế của Blowfish như một phiên bản nâng cao vào năm 1998 và là một trong những ứng viên cuối cùng của cuộc thi AES. Twofish hỗ trợ các khóa 128-bit, 192-bit, và 256-bit. Twofish là mã hóa khối với kích thước khối 128-bit, có cấu trúc Feistel với 16 vòng lặp. Nó cũng sử dụng các phép toán XOR, hoán vị, và các phép biến đổi trên các byte dữ liệu. Twofish có độ linh hoạt và bảo mật cao, được thiết kế cho tốc độ và an toàn trên cả phần cứng và phần mềm.

Ví dụ: Khi mã hóa chuỗi "HELLO" bằng khóa 256-bit trong Twofish, thông điệp sẽ được chia thành các khối 128-bit và mã hóa qua 16 vòng biến đổi.

***f. RC4 (Rivest Cipher 4)***

RC4 là một mã hóa dòng (stream cipher) do Ron Rivest phát triển vào năm 1987. Thay vì mã hóa khối, nó mã hóa từng byte dữ liệu một cách tuần tự. Thuật toán RC4 hỗ trợ các khóa từ 40-bit đến 2048-bit. RC4 tạo ra một dòng khóa ngẫu nhiên (keystream) và XOR nó với từng byte trong văn bản gốc để tạo ra văn bản mã hóa. Mặc dù RC4 rất nhanh và được sử dụng rộng rãi (trong SSL, WEP), nhưng nó đã bị phát hiện có nhiều lỗ hổng bảo mật, dẫn đến việc nó bị thay thế trong nhiều ứng dụng.

Ví dụ: Nếu mã hóa chuỗi "HELLO" bằng RC4, mỗi ký tự sẽ được XOR với một byte của dòng khóa để tạo ra văn bản mã hóa.

***g. Serpent***

Serpent được phát triển vào năm 1998 bởi Anderson, Biham, và Knudsen như một ứng viên cho AES. Thuật toán Serpent hỗ trợ các khóa 128-bit, 192-bit, và 256-bit. Serpent sử dụng một cấu trúc mạng hoán vị thay thế với 32 vòng biến đổi, với mục tiêu ưu tiên tính an toàn hơn là tốc độ. Serpent được thiết kế để chống lại tất cả các cuộc tấn công đã biết vào thời điểm phát triển và được coi là an toàn hơn AES, mặc dù chậm hơn.

Ví dụ: Khi mã hóa "HELLO" bằng khóa 128-bit trong Serpent, thông điệp sẽ được chia thành khối 128-bit và trải qua 32 vòng biến đổi.

*1.1.4.2. Chế độ hoạt động của mã hóa khối (Block Cipher Modes of Operation)*

Các thuật toán mã hóa khối như DES, AES, và Blowfish hoạt động trên các khối dữ liệu có kích thước cố định (ví dụ: AES mã hóa các khối 128-bit). Tuy nhiên, dữ liệu thực tế thường không chia đều thành các khối có kích thước cố định. Để giải quyết vấn đề này và xử lý các khối dữ liệu có độ dài thay đổi, cũng như tăng cường tính bảo mật, các chế độ hoạt động (modes of operation) đã được phát triển. Các chế độ hoạt động phổ biến của mã hóa khối được trình bày dưới đây:

***ECB (Electronic Codebook):*** Trong chế độ ECB, mỗi khối dữ liệu sẽ được mã hóa độc lập. Nếu có hai khối giống nhau trong văn bản gốc, chúng sẽ được mã hóa thành hai khối giống nhau trong văn bản mã hóa.

ECB đơn giản và nhanh chóng do mỗi khối có thể được mã hóa song song. Tuy nhiên ECB không bảo mật khi dữ liệu có các mẫu lặp lại, vì các khối giống nhau sẽ tạo ra văn bản mã hóa giống nhau, làm lộ cấu trúc dữ liệu. Ví dụ, hình ảnh mã hóa bằng ECB vẫn giữ lại các mẫu của hình ảnh ban đầu.

ECB hiếm khi được sử dụng trong các hệ thống bảo mật hiện đại do những lỗ hổng bảo mật của nó.

Ví dụ: Khi mã hóa một file hình ảnh có các vùng màu giống nhau, ECB sẽ tạo ra các khối mã hóa giống nhau cho các vùng này, làm lộ mẫu hình ảnh.

***CBC (Cipher Block Chaining):*** Mỗi khối dữ liệu sẽ được XOR với khối mã hóa trước đó trước khi được mã hóa. Khối đầu tiên được XOR với một giá trị khởi tạo ngẫu nhiên gọi là Initialization Vector (IV).

CBC giải quyết vấn đề lặp lại của ECB bằng cách đảm bảo rằng các khối giống nhau sẽ tạo ra các khối mã hóa khác nhau, miễn là IV khác nhau. Nó cũng tạo ra tính ngẫu nhiên ngay cả khi văn bản gốc có các mẫu lặp lại. CBC không hỗ trợ mã hóa song song, vì mỗi khối phụ thuộc vào khối trước đó. Điều này khiến CBC chậm hơn khi xử lý nhiều dữ liệu cùng lúc.

CBC được sử dụng rộng rãi trong các giao thức bảo mật như SSL/TLS và IPsec.

Ví dụ: Khi mã hóa chuỗi "HELLO WORLD" bằng CBC, mỗi khối sẽ được XOR với khối mã hóa trước đó, tạo ra các khối mã hóa khác nhau ngay cả khi chuỗi gốc có các phần giống nhau.

***CFB (Cipher Feedback):*** CFB là một chế độ mã hóa dòng (stream mode) dựa trên mã hóa khối. Thay vì mã hóa toàn bộ khối, CFB mã hóa một phần dữ liệu theo từng đoạn (ví dụ: 8-bit hoặc 64-bit), sau đó XOR đoạn này với văn bản gốc để tạo thành văn bản mã hóa.

CFB cho phép mã hóa và giải mã các đoạn dữ liệu có kích thước thay đổi, do đó phù hợp cho truyền thông dữ liệu liên tục (streaming). Tuy vậy, giống như CBC, CFB không hỗ trợ mã hóa song song.

CFB thường được sử dụng cho mã hóa dòng dữ liệu trong mạng truyền tải và liên lạc thời gian thực.

Ví dụ: Khi mã hóa chuỗi ký tự bằng CFB, hệ thống sẽ xử lý từng đoạn nhỏ của dữ liệu và XOR với văn bản mã hóa trước đó.

***OFB (Output Feedback):*** OFB cũng là chế độ mã hóa dòng, nhưng thay vì sử dụng đầu ra của khối trước đó, nó tạo ra một chuỗi khóa ngẫu nhiên từ mã hóa của IV. Chuỗi khóa này sẽ được XOR với từng đoạn dữ liệu của văn bản gốc để tạo ra văn bản mã hóa.

OFB cho phép xử lý song song và bảo vệ dữ liệu khỏi lỗi truyền tải, vì lỗi trong một khối không ảnh hưởng đến các khối sau. Nhưng OFB yêu cầu IV được giữ bí mật và không được sử dụng lại, vì việc sử dụng lại IV có thể dẫn đến lỗ hổng bảo mật.

OFB phù hợp cho các ứng dụng yêu cầu bảo vệ mạnh mẽ trước lỗi truyền thông, như mã hóa liên lạc vệ tinh.

Ví dụ: Khi mã hóa chuỗi dữ liệu bằng OFB, mỗi khối sẽ được XOR với một phần của chuỗi khóa ngẫu nhiên để tạo ra văn bản mã hóa.

***CTR (Counter Mode):*** Trong CTR, mỗi khối sẽ được XOR với đầu ra của một bộ đếm (counter) đã được mã hóa bằng thuật toán mã hóa khối. Bộ đếm tăng dần theo mỗi khối, tạo ra một chuỗi khóa duy nhất cho mỗi khối.

CTR cho phép mã hóa và giải mã song song, làm tăng hiệu suất đáng kể, đặc biệt là trên phần cứng đa lõi. Nó cũng không yêu cầu padding, giúp đơn giản hóa mã hóa các dữ liệu có độ dài thay đổi. Giống như OFB, CTR có nhược điểm là yêu cầu giá trị bộ đếm không được tái sử dụng, nếu không sẽ tạo ra lỗ hổng bảo mật.

CTR được sử dụng rộng rãi trong các giao thức bảo mật như IPsec và các ứng dụng mã hóa dữ liệu lớn.

Ví dụ: Khi mã hóa một file lớn bằng CTR, bộ đếm sẽ được mã hóa để tạo ra các giá trị khác nhau cho mỗi khối và XOR với văn bản gốc.

### 1.1.5. Mã hóa bất đối xứng (Asymmetric Encryption)

Mã hóa bất đối xứng, còn được gọi là mã hóa khóa công khai, là một hệ thống mã hóa sử dụng hai khóa khác nhau nhưng liên kết với nhau: một khóa công khai (public key) và một khóa bí mật (private key). Khóa công khai được sử dụng để mã hóa dữ liệu và khóa bí mật dùng để giải mã dữ liệu. Điều đặc biệt là dữ liệu được mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa bí mật tương ứng và ngược lại.

Khóa công khai: Được chia sẻ rộng rãi và dùng để mã hóa dữ liệu.

Khóa bí mật: Được giữ bí mật và dùng để giải mã dữ liệu.

Một số thuật toán mã hóa bất đối xứng điển hình

***a. RSA (Rivest-Shamir-Adleman)***

RSA dựa trên độ khó của việc phân tích một số lớn thành các thừa số nguyên tố. Khóa công khai và khóa bí mật được tạo dựa trên các cặp số nguyên tố lớn.

Đây là thuật toán phổ biến và đã được sử dụng rộng rãi trong các hệ thống như SSL/TLS, email bảo mật (PGP), và chữ ký số. Nhưng RSA yêu cầu độ dài khóa lớn để đảm bảo an toàn, gây tốn tài nguyên tính toán.

Ví dụ: Trong một phiên giao dịch bảo mật trên web, khóa công khai RSA của máy chủ được sử dụng để mã hóa dữ liệu truyền giữa máy chủ và trình duyệt.

***b. DSA (Digital Signature Algorithm)***

DSA được thiết kế riêng để tạo và xác minh chữ ký số. DSA không trực tiếp mã hóa dữ liệu, mà chỉ tạo chữ ký số để xác thực tính toàn vẹn và nguồn gốc của thông tin.

Ưu điểm của DSA là tối ưu hóa cho việc ký và xác minh chữ ký số. Nhược điểm của nó là không thể sử dụng để mã hóa dữ liệu.

Ví dụ: DSA thường được sử dụng trong các hệ thống như GPG để xác thực email và tài liệu.

***c. ECC (Elliptic Curve Cryptography)***

ECC dựa trên các tính chất toán học của đường cong elliptic để tạo khóa công khai và khóa bí mật. ECC cung cấp mức độ bảo mật tương tự như RSA với kích thước khóa nhỏ hơn nhiều.

ECC có hiệu suất tốt hơn RSA, đặc biệt trên các thiết bị có tài nguyên hạn chế (ví dụ: điện thoại di động, IoT). Tuy nhiên cần phải chọn đúng các tham số đường cong elliptic để đảm bảo tính bảo mật.

Ví dụ: ECC được sử dụng trong các giao thức bảo mật hiện đại như HTTPS, VPN và blockchain (ví dụ: Bitcoin sử dụng ECC để quản lý các khóa ví).

***d. ElGamal***

ElGamal là một thuật toán mã hóa dựa trên độ khó của bài toán logarithm rời rạc. Nó cung cấp cả tính năng mã hóa dữ liệu và tạo chữ ký số.

ElGamal có ưu điểm là bảo mật tốt với các khóa lớn, có khả năng chống lại nhiều loại tấn công nhưng kích  thước văn bản mã hóa lớn hơn so với RSA và ECC.

Ví dụ: ElGamal thường được sử dụng trong các hệ thống yêu cầu mã hóa và chữ ký số đồng thời.

***e. Paillier***

Paillier là một thuật toán mã hóa đồng cấu (homomorphic encryption), cho phép thực hiện các phép toán trên văn bản mã hóa mà không cần giải mã nó.

Paillier có thể áp dụng cho các hệ thống yêu cầu tính toán bảo mật trên dữ liệu mã hóa, như các dịch vụ lưu trữ đám mây. Tuy vậy, nó có hiệu suất kém hơn so với các thuật toán khác trong các ứng dụng thông thường.

Ví dụ: Paillier thường được sử dụng trong các ứng dụng yêu cầu tính toán trên dữ liệu bí mật mà không cần giải mã, như xử lý dữ liệu tài chính.

***1.1.4.3. Một vài ví dụ thực tiễn***

Giao thức SSL/TLS: Trong một phiên SSL/TLS, mã hóa bất đối xứng (thường là RSA hoặc ECC) được sử dụng để trao đổi khóa đối xứng ban đầu. Sau đó, mã hóa đối

xứng (thường là AES) được sử dụng để mã hóa các dữ liệu tiếp theo nhằm tối ưu hóa hiệu suất.

Chữ ký số trong email (PGP/GPG): Người gửi sử dụng khóa bí mật của mình để tạo chữ ký số, và người nhận sử dụng khóa công khai của người gửi để xác minh rằng email chưa bị thay đổi và đúng là từ người gửi.

Blockchain: Trong blockchain như Bitcoin, ECC được sử dụng để quản lý khóa ví. Người dùng tạo địa chỉ ví công khai để nhận tiền, và chỉ họ với khóa bí mật tương ứng mới có thể chi tiêu số tiền đó.

## 1.2. HÀM BĂM

### 1.2.1. Khái niệm và vai trò của hàm băm trong bảo mật

Hàm băm (hash function) là một hàm toán học có khả năng chuyển đổi dữ liệu đầu vào có kích thước bất kỳ (ví dụ: văn bản, tệp tin, số liệu) thành một chuỗi ký tự có độ dài cố định, gọi là giá trị băm (hash value) hoặc băm (hash). Giá trị này đại diện duy nhất cho dữ liệu ban đầu và thay đổi hoàn toàn nếu chỉ một chút thông tin trong dữ liệu đầu vào bị thay đổi.

***Hàm băm có một số đặc tính chủ yếu***

Xác định (Deterministic): Cùng một đầu vào luôn cho ra cùng một giá trị băm.

Kích thước cố định: Dù đầu vào có độ dài bất kỳ, đầu ra của hàm băm luôn có kích thước cố định.

Tính nhanh chóng: Hàm băm phải tính toán giá trị băm nhanh chóng ngay cả với lượng dữ liệu lớn.

Kháng va chạm (Collision-resistant): Rất khó hoặc không thể tìm được hai dữ liệu khác nhau có cùng giá trị băm.

Một chiều (One-way): Rất khó hoặc không thể tái tạo dữ liệu gốc từ giá trị băm.

Tính lan truyền (Avalanche Effect): Một thay đổi nhỏ trong dữ liệu đầu vào sẽ gây ra sự thay đổi đáng kể trong giá trị băm.

***Vai trò của hàm băm trong bảo mật***

Hàm băm đóng vai trò quan trọng trong nhiều lĩnh vực của bảo mật thông tin, giúp đảm bảo tính toàn vẹn và bảo vệ dữ liệu khỏi các cuộc tấn công. Dưới đây là một số vai trò nổi bật:

Đảm bảo tính toàn vẹn của dữ liệu: Hàm băm được sử dụng để kiểm tra tính toàn vẹn của dữ liệu, đảm bảo rằng dữ liệu không bị thay đổi trong quá trình truyền tải hoặc lưu trữ. Ví dụ, khi tải một tệp tin từ internet, một giá trị băm của tệp (thường là SHA-256 hoặc MD5) được cung cấp để người dùng có thể so sánh giá trị băm của tệp tải về với giá trị băm đã được công bố. Nếu các giá trị này khớp nhau, điều đó chứng tỏ tệp không bị thay đổi.

Chữ ký số (Digital Signatures): Trong quá trình tạo chữ ký số, hàm băm được sử dụng để tạo ra một giá trị đại diện duy nhất cho tài liệu hoặc thông điệp. Sau đó, giá trị băm này sẽ được mã hóa bằng khóa bí mật của người gửi để tạo chữ ký số. Người nhận sử dụng khóa công khai của người gửi để kiểm tra chữ ký và đối chiếu giá trị băm với tài liệu ban đầu để xác thực tính toàn vẹn và nguồn gốc.

Mật mã hóa mật khẩu: Trong các hệ thống đăng nhập, thay vì lưu trữ mật khẩu gốc của người dùng, hệ thống thường lưu trữ giá trị băm của mật khẩu. Khi người dùng nhập mật khẩu, hệ thống sẽ băm mật khẩu đó và so sánh giá trị băm với giá trị đã lưu. Điều này giúp bảo vệ mật khẩu khỏi việc bị lộ khi dữ liệu bị tấn công. Các kỹ thuật như salting (thêm một chuỗi ngẫu nhiên vào mật khẩu trước khi băm) giúp tăng cường bảo mật, làm cho việc tấn công từ điển hoặc tấn công Rainbow Table (bảng băm sẵn) trở nên khó khăn hơn.

Blockchain: Trong các hệ thống blockchain như Bitcoin, hàm băm được sử dụng để liên kết các khối (blocks) với nhau và đảm bảo tính toàn vẹn của chuỗi. Mỗi khối chứa một giá trị băm của khối trước đó, làm cho việc thay đổi dữ liệu trong một khối yêu cầu phải thay đổi tất cả các khối sau nó, điều này gần như là không thể. Quá trình khai thác (mining) trong blockchain cũng dựa trên việc tìm ra giá trị băm phù hợp với các yêu cầu nhất định, thông qua các phép tính toán rất phức tạp.

***Xác thực dữ liệu trong giao thức truyền thông***

Hàm băm được sử dụng trong các giao thức bảo mật như SSL/TLS, IPsec, HMAC (Hashed Message Authentication Code) để đảm bảo rằng các thông điệp không bị thay đổi trong quá trình truyền và để xác thực tính xác thực của các bên tham gia.

HMAC là một kỹ thuật kết hợp giữa khóa bí mật và hàm băm, giúp bảo vệ dữ liệu khỏi tấn công sửa đổi.

Phát hiện virus và malware: Các phần mềm phát hiện virus thường sử dụng hàm băm để so sánh tệp tin với các mẫu virus đã biết. Nếu giá trị băm của một tệp khớp với giá trị băm của một mẫu virus, phần mềm sẽ cảnh báo về sự tồn tại của malware trong hệ thống.

### 1.2.2. Phân biệt mã hóa và hàm băm

Cả mã hóa và hàm băm đều là các kỹ thuật bảo mật được sử dụng để bảo vệ thông tin, nhưng chúng có bản chất và mục tiêu hoàn toàn khác nhau.

***a. Mã hóa (Encryption)***

Mã hóa là quá trình chuyển đổi dữ liệu gốc (plaintext) thành dữ liệu đã được mã hóa (ciphertext) sao cho người không được ủy quyền không thể đọc được dữ liệu. Quá trình này có thể đảo ngược: người nhận có thể giải mã dữ liệu để khôi phục lại thông tin ban đầu bằng cách sử dụng một khóa bí mật (trong mã hóa đối xứng) hoặc một cặp khóa công khai và khóa bí mật (trong mã hóa bất đối xứng).

Mục tiêu của mã hóa là đảm bảo bí mật của dữ liệu bằng cách ẩn nội dung của nó khỏi các bên không có thẩm quyền.

Mã hóa sử dụn 2 loại khóa: Mã hóa đối xứng: Sử dụng cùng một khóa cho cả mã hóa và giải mã (ví dụ: AES, DES). Và mã hóa bất đối xứng: Sử dụng cặp khóa công khai và khóa bí mật, khóa công khai dùng để mã hóa và khóa bí mật dùng để giải mã (ví dụ: RSA, ECC).

Ví dụ thực tiễn: Giao thức HTTPS: Khi bạn truy cập một trang web qua HTTPS, dữ liệu giữa trình duyệt và máy chủ web được mã hóa bằng các thuật toán như AES (đối xứng) và RSA (bất đối xứng) để bảo vệ nội dung khỏi các kẻ tấn công trong quá trình truyền.

Email bảo mật (PGP): Khi gửi một email bảo mật bằng PGP, dữ liệu được mã hóa bằng khóa công khai của người nhận, chỉ có khóa bí mật của họ mới có thể giải mã và đọc nội dung email.

Quá trình đảo ngược: Có thể giải mã được để lấy lại thông tin ban đầu (plaintext).

***b. Hàm băm (Hash Function)***

Hàm băm là một hàm toán học chuyển đổi dữ liệu đầu vào có độ dài bất kỳ thành một chuỗi ký tự cố định (giá trị băm) mà không có quá trình giải mã để quay lại dữ liệu ban đầu. Hàm băm được thiết kế để không thể đảo ngược (một chiều), có nghĩa là không thể khôi phục lại dữ liệu gốc từ giá trị băm.

Mục tiêu của hàm băm là nhằm đảm bảo tính toàn vẹn của dữ liệu bằng cách kiểm tra xem dữ liệu có bị thay đổi hay không. Hàm băm không dùng để bảo mật nội dung, mà dùng để so sánh, xác minh dữ liệu.

Hàm băm không yêu cầu khóa trong quá trình tính toán. Giá trị băm phụ thuộc hoàn toàn vào nội dung của dữ liệu.

Một số ví dụ thực tiễn:

Kiểm tra tính toàn vẹn của tệp tin: Khi tải xuống một tệp từ internet, trang web có thể cung cấp giá trị băm (ví dụ: SHA-256) để người dùng có thể kiểm tra xem tệp tin tải về có bị thay đổi hoặc hỏng hóc trong quá trình tải hay không.

Mật khẩu: Trong các hệ thống đăng nhập, thay vì lưu trữ mật khẩu ở dạng văn bản thuần túy, hệ thống lưu trữ giá trị băm của mật khẩu. Khi người dùng nhập mật khẩu, hệ thống sẽ băm mật khẩu đó và so sánh với giá trị băm đã lưu. Điều này giúp bảo vệ mật khẩu khỏi bị lộ khi cơ sở dữ liệu bị xâm phạm.

Quá trình đảo ngược: Không thể giải mã, chỉ có thể so sánh giá trị băm để kiểm tra tính toàn vẹn của dữ liệu.

***c. So sánh mã hóa và hàm băm***

Mã hóa trong giao dịch ngân hàng trực tuyến: Khi bạn thực hiện giao dịch qua ngân hàng trực tuyến, dữ liệu tài chính của bạn được mã hóa bằng các thuật toán đối xứng (như AES) và bất đối xứng (như RSA) để đảm bảo rằng chỉ có ngân hàng mới có thể đọc được thông tin nhạy cảm này.

Hàm băm trong kiểm tra tính toàn vẹn của tệp tin: Khi tải một bản cập nhật phần mềm từ trang web của nhà phát triển, họ cung cấp giá trị băm SHA-256 của tệp tin. Sau khi tải xuống, bạn có thể so sánh giá trị băm của tệp tin trên máy tính với giá trị băm đã cung cấp. Nếu khớp, tệp tin không bị thay đổi hoặc lỗi trong quá trình tải xuống.

Mã hóa và hàm băm kết hợp trong bảo mật mật khẩu: Khi đăng nhập vào một hệ thống, mật khẩu của bạn được mã hóa trong quá trình truyền từ trình duyệt đến máy chủ để bảo mật (ví dụ: qua HTTPS). Sau khi đến máy chủ, mật khẩu được băm (thường là với SHA-256) và giá trị băm đó được so sánh với giá trị băm đã lưu trong cơ sở dữ liệu để xác nhận tính đúng đắn của mật khẩu mà không cần biết mật khẩu gốc.

Như vậy, mã hóa và hàm băm đều là những công cụ bảo mật quan trọng, nhưng chúng phục vụ các mục đích khác nhau. Mã hóa được dùng để bảo vệ tính bí mật của thông tin trong quá trình truyền tải hoặc lưu trữ, trong khi hàm băm được dùng để kiểm tra tính toàn vẹn của dữ liệu và bảo vệ hệ thống khỏi sự thay đổi không mong muốn.

### 1.2.3. Các hàm băm phổ biến

Hàm băm là một thành phần quan trọng trong các hệ thống bảo mật và mật mã học. Nó giúp bảo vệ tính toàn vẹn của dữ liệu, xác thực và bảo mật trong nhiều ứng dụng. Phần này trình bày chi tiết hơn về các hàm băm phổ biến

#### *1.2.3.1 MD5 (Message Digest Algorithm 5)*

Thuật toán MD5 do Ronald Rivest phát triển năm 1991 là phiên bản kế tiếp của MD4, với mục tiêu cải thiện tính bảo mật và hiệu suất. Trước khi bị lỗ hổng bảo mật phát hiện, MD5 được sử dụng rộng rãi trong việc xác minh tính toàn vẹn của dữ liệu.

Ý tưởng của MD5 được thiết kế để chuyển đổi đầu vào có độ dài bất kỳ thành một giá trị băm cố định 128-bit. MD5 sử dụng một quá trình bao gồm 4 vòng lặp xử lý dữ liệu qua một loạt các phép toán bổ sung và dịch chuyển bit.

Thuật toán MD5 chia thông tin đầu vào thành các khối 512-bit, sau đó thực hiện một loạt các phép toán khối XOR, phép toán bổ sung, và dịch chuyển để tạo ra giá trị băm cuối cùng. Quy trình này bao gồm một số vòng lặp với các phép toán bí mật.

Ví dụ: Chuỗi "hello" có mã băm MD5: 5d41402abc4b2a76b9719d911017c592

MD5 có tốc độ tính toán nhanh chóng và dễ dàng triển khai và sử dụng trong nhiều ứng dụng.

Nhược điểm của MD5 là dễ bị tấn công va chạm, nơi hai thông điệp khác nhau có thể cho cùng một giá trị băm. Điều này làm giảm tính an toàn của nó. Một nhược điểm nữa là do các lỗ hổng bảo mật, MD5 không còn an toàn cho các ứng dụng bảo mật như xác thực mật khẩu hay chữ ký số.

MD 5 thường được áp dụng:

+ Kiểm tra tính toàn vẹn của tệp tin: MD5 được sử dụng để xác nhận rằng tệp tin không bị thay đổi khi truyền qua internet.

+ Mã hóa mật khẩu: Trước đây, MD5 được sử dụng để băm mật khẩu trong các hệ thống, nhưng hiện nay đã bị thay thế bởi các thuật toán mạnh mẽ hơn.

#### *1.2.3.2. SHA-1 (Secure Hash Algorithm 1)*

Thuật toán SHA-1 phát triển bởi NSA (Cơ quan An ninh Quốc gia Hoa Kỳ) năm 1993. Thuật toán SHA-1 được sử dụng rộng rãi trong nhiều ứng dụng bảo mật, bao gồm chứng chỉ SSL/TLS, nhưng các nghiên cứu cho thấy SHA-1 không còn an toàn.

Ý tưởng của SHA-1 là tạo ra một giá trị băm 160-bit từ đầu vào có độ dài bất kỳ. Quá trình băm bao gồm các phép toán XOR, phép dịch chuyển bit và bổ sung trong các vòng lặp để tạo ra giá trị băm.

Thuật toán SHA-1 sử dụng 5 khối dữ liệu 32-bit trong 80 vòng lặp. Mỗi vòng lặp áp dụng một phép toán XOR, phép toán bổ sung, và một phép toán logic.

Ví dụ: Chuỗi "hello" băm SHA-1: 2ef7bde608ce5404e97d5f042f95f89f1c232871

SHA-1 là được sử dụng rộng rãi trong các giao thức bảo mật và xác thực số. Đảm bảo tính toàn vẹn của thông tin trong các ứng dụng như chữ ký số và chứng chỉ SSL.

Tuy vậy, SHA-1 có một số nhược điểm:

+ Khả năng va chạm: Từ năm 2005, các nhà nghiên cứu đã chỉ ra rằng SHA-1 dễ bị tấn công va chạm.

+ Khuyến cáo ngừng sử dụng: NIST đã khuyến cáo không sử dụng SHA-1 cho các ứng dụng bảo mật vì tính dễ bị tấn công của nó.

Áp dụng:

+ Chữ ký số và chứng chỉ SSL/TLS: SHA-1 được sử dụng trong quá trình tạo chữ ký số cho các chứng chỉ SSL.

+ Xác thực thông điệp: Trước khi bị thay thế, SHA-1 được sử dụng để xác thực tính toàn vẹn của thông điệp trong giao thức bảo mật.

#### *1.2.3.3. SHA-2 (Secure Hash Algorithm 2)*

Thuật toán SHA-2 phát triển bởi NSA năm 2001 là phiên bản cải tiến của SHA-1, được thiết kế để khắc phục các lỗ hổng của SHA-1.

Ý tưởng của SHA-2 là một họ các hàm băm bao gồm các phiên bản SHA-224, SHA-256, SHA-384, và SHA-512, mỗi phiên bản có độ dài băm khác nhau. Quá trình băm trong SHA-2 sử dụng các phép toán bổ sung, XOR và dịch chuyển để tạo ra giá trị băm an toàn.

Thuật toán SHA-2 áp dụng một loạt các phép toán logic mạnh mẽ hơn SHA-1, với độ dài băm thay đổi tùy thuộc vào phiên bản được sử dụng.

Ví dụ: Chuỗi: "hello" có mã băm SHA-256 là

2cf24dba5fb0a30e26e83b2ac5b9e29e1b169e7e9e1c8e4e58c96d9e2b9bda94

SHA-2 an toàn hơn SHA-1 và MD5, được khuyến nghị cho các ứng dụng bảo mật hiện đại như blockchain, SSL/TLS. Tính bảo mật cao, kháng lại các cuộc tấn công va chạm.

Tốc độ SHA-2 có thể chậm hơn MD5 và SHA-1, nhưng đổi lại nó mang lại tính bảo mật cao hơn.

Áp dụng:

+ Blockchain: SHA-256 được sử dụng trong các blockchain như Bitcoin để bảo mật các giao dịch.

+ Chữ ký số: SHA-2 được sử dụng trong các chứng chỉ SSL/TLS và chữ ký số để đảm bảo tính toàn vẹn và xác thực của dữ liệu.

#### *1.2.3.4. SHA-3 (Secure Hash Algorithm 3)*

Thuật toán SHA-3 được phát triển bởi NIST năm 2015 để thay thế SHA-2 trong trường hợp SHA-2 bị phá vỡ trong tương lai. Thuật toán SHA-3 sử dụng một cấu trúc khác với SHA-2 (Keccak), mang lại tính an toàn cao hơn.

Ý tưởng của SHA-3 sử dụng một cấu trúc khác biệt hoàn toàn gọi là Keccak, một cấu trúc dựa trên phương pháp sponge, khác với các phương pháp dựa trên Merkle–Damgård của SHA-1 và SHA-2.

Thuật toán của Keccak chia thông tin thành các khối và sử dụng các phép toán cộng, XOR, và dịch chuyển bit để tạo ra giá trị băm.

Ví dụ: Chuỗi: "hello" có mã băm SHA-3-256 là
b94d27b9934d3e08a52e52d7da7dabfad3a9c404b85e347506f906420dce2086

Ưu điểm của SHA-3 là có tính an toàn cao, không có lỗi va chạm được phát hiện. Cấu trúc Keccak mạnh mẽ hơn SHA-2 trong việc bảo vệ chống lại các tấn công tiềm ẩn.

Mặc dù mạnh mẽ nhưng SHA-3 chưa được áp dụng rộng rãi như SHA-2.

Áp dụng:

+ Ứng dụng bảo mật cao: SHA-3 có thể được sử dụng trong các ứng dụng bảo mật yêu cầu độ an toàn cực kỳ cao.

+ Blockchain tương lai: Có thể sử dụng SHA-3 thay thế SHA-2 trong các hệ thống blockchain trong tương lai.

#### *1.2.3.5. BLAKE2*

Thuật toán BLAKE2 được phát triển bởi các nhà nghiên cứu Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn năm 2012. Sự phát triển: BLAKE2 được thiết kế như một phiên bản thay thế cho MD5 và SHA-2, với hiệu suất cao và bảo mật tốt hơn.

BLAKE2 là một hàm băm nhanh và an toàn, có thể thay thế MD5 và SHA-2 trong nhiều ứng dụng cần tốc độ cao và tính bảo mật.

BLAKE2 sử dụng một cấu trúc Merkle tree và có hiệu suất rất cao khi so với SHA-2.

Ví dụ: Chuỗi "hello" băm BLAKE2b-256: f1d2d2f924e986ac86fdf7b2a5a7f52d

Ưu điểm: Tốc độ nhanh chóng, hiệu suất cao, Bảo mật mạnh mẽ và dễ triển khai.

Nhược điểm: Chưa phổ biến rộng rãi như SHA-2, mặc dù nó mạnh mẽ và nhanh.

Áp dụng:

+ Hệ thống tệp tin: BLAKE2 được sử dụng trong nhiều hệ thống tệp tin và ứng dụng cần tính toán băm nhanh.

+ Phần mềm bảo mật: BLAKE2 được sử dụng trong các phần mềm bảo mật và xác minh dữ liệu.

Mỗi hàm băm có những đặc điểm riêng biệt và phù hợp với các nhu cầu khác nhau. MD5 và SHA-1 đã dần lỗi thời do các lỗ hổng bảo mật, trong khi SHA-2 và SHA-3 đang là các lựa chọn chính trong các ứng dụng bảo mật hiện đại. BLAKE2 là một lựa chọn mới mạnh mẽ với hiệu suất nhanh chóng.

### 1.2.4. Một số ứng dụng của hàm băm

Hàm băm có nhiều ứng dụng trong bảo mật và các hệ thống thông tin, đặc biệt là trong việc đảm bảo tính toàn vẹn của dữ liệu, bảo vệ mật khẩu, và hỗ trợ các giao dịch an toàn. Dưới đây là một số ứng dụng phổ biến của hàm băm:

#### *1.2.4.1. Kiểm tra tính toàn vẹn của phần mềm và tệp tin (Software and File Integrity Checking)*

Hàm băm được sử dụng rộng rãi để kiểm tra tính toàn vẹn của phần mềm hoặc tệp tin tải về từ internet, nhằm đảm bảo rằng tệp không bị thay đổi hoặc bị tấn công.

- Tải phần mềm từ website: Các trang web cung cấp phần mềm hoặc tệp tin (như ISO hệ điều hành, phần mềm, tài liệu) thường công bố giá trị băm (MD5, SHA-1 hoặc SHA-256) của tệp. Người dùng có thể tải về phần mềm và tính toán lại giá trị băm của tệp sau khi tải xuống, nếu giá trị băm trùng khớp, tệp không bị thay đổi.

- Phần mềm bảo mật: Các công cụ như Tripwire kiểm tra sự thay đổi của các tệp trong hệ thống và cảnh báo người dùng khi có sự thay đổi không mong muốn. Những thay đổi này có thể là kết quả của phần mềm độc hại hoặc thao tác bất hợp pháp.

#### *1.2.4.2. Hệ thống quản lý mật khẩu (Password Management Systems)*

Hàm băm giúp bảo vệ mật khẩu của người dùng trong hệ thống đăng nhập. Khi người dùng tạo mật khẩu, hệ thống không lưu trữ mật khẩu gốc mà lưu trữ giá trị băm của mật khẩu đó. Khi người dùng đăng nhập, hệ thống sẽ băm mật khẩu và so sánh với giá trị băm đã lưu trữ.

Để làm tăng tính bảo mật, khi băm mật khẩu, các hệ thống thêm một chuỗi ngẫu nhiên (salt) vào mật khẩu trước khi tính toán giá trị băm, giúp ngăn chặn các cuộc tấn

công từ điển (dictionary attacks) và tấn công vết dầu (rainbow table attacks). Ví dụ: bcrypt và Argon2 sử dụng salt để bảo vệ mật khẩu người dùng.

#### *1.2.4.3. Blockchain và Tiền mã hóa (Cryptocurrency and Blockchain)*

Hàm băm là thành phần thiết yếu trong các hệ thống blockchain như Bitcoin, Ethereum, giúp bảo vệ tính toàn vẹn của dữ liệu giao dịch và tạo ra các khối (block) trong chuỗi. Mỗi khối chứa giá trị băm của khối trước đó và giao dịch mới, tạo thành một chuỗi không thể thay đổi.

- Bitcoin: SHA-256 được sử dụng trong việc bảo vệ giao dịch và các khối trong blockchain Bitcoin. Mỗi khối trong chuỗi chứa giá trị băm của khối trước đó, giúp tạo ra một chuỗi liên kết mà không thể thay đổi.

- Ethereum: Ethereum sử dụng SHA-3 để băm các giao dịch và khối trong mạng blockchain của mình.

#### *1.2.4.4. Chữ ký số (Digital Signatures)*

Hàm băm là một phần quan trọng trong chữ ký số, giúp xác thực tính toàn vẹn của dữ liệu và xác minh danh tính người ký. Quy trình tạo chữ ký số bao gồm băm dữ liệu và mã hóa giá trị băm bằng khóa riêng của người ký.

- Chứng chỉ SSL/TLS: Hàm băm như SHA-256 được sử dụng để băm dữ liệu chứng chỉ số trong quy trình tạo và kiểm tra chữ ký số, giúp đảm bảo tính toàn vẹn và xác thực nguồn gốc của các chứng chỉ SSL/TLS.

- Chữ ký điện tử: Các tổ chức, ngân hàng, và dịch vụ điện tử sử dụng chữ ký số để ký hợp đồng, tài liệu, hoặc giao dịch trực tuyến. Ví dụ, chữ ký số trong giao dịch ngân hàng điện tử giúp xác nhận tính hợp pháp của giao dịch.

#### *1.2.4.5. Kiểm tra tính toàn vẹn của cơ sở dữ liệu (Database Integrity Checking)*

Trong các hệ thống quản lý cơ sở dữ liệu (DBMS), hàm băm có thể được sử dụng để kiểm tra sự thay đổi của các bản ghi trong cơ sở dữ liệu. Điều này giúp xác định xem có sự truy cập trái phép hoặc thay đổi dữ liệu không mong muốn trong cơ sở dữ liệu hay không.

- Phát hiện gian lận trong giao dịch ngân hàng: Các ngân hàng có thể sử dụng hàm băm để theo dõi các giao dịch tài chính và kiểm tra tính toàn vẹn của dữ liệu.

- Kiểm soát thay đổi: Các công cụ quản lý cơ sở dữ liệu có thể sử dụng hàm băm để xác thực rằng không có sự thay đổi trái phép trong các bản ghi quan trọng của cơ sở dữ liệu.

#### *1.2.4.6. Xác thực thông điệp (Message Authentication)*

Hàm băm kết hợp với một khóa bí mật có thể tạo ra mã xác thực thông điệp (MAC) để đảm bảo tính toàn vẹn và xác thực của thông điệp trong các giao dịch điện tử hoặc giao tiếp qua mạng.

- HMAC (Hash-based Message Authentication Code): HMAC sử dụng một hàm băm (như SHA-256) kết hợp với một khóa bí mật để tạo ra mã xác thực cho thông điệp. Điều này giúp bảo vệ các thông điệp khỏi việc bị thay đổi trong quá trình truyền tải. HMAC được sử dụng trong các giao thức bảo mật như SSL/TLS, IPsec và SSH.

- API và Web Services: Trong các giao dịch API hoặc giao tiếp giữa các máy chủ, HMAC được sử dụng để bảo vệ tính toàn vẹn của dữ liệu và ngăn chặn các cuộc tấn công man-in-the-middle.

#### *1.2.4.7. Quản lý dữ liệu và phân tán (Distributed File Systems)*

Hàm băm giúp đảm bảo tính toàn vẹn của dữ liệu trong các hệ thống lưu trữ phân tán như IPFS (InterPlanetary File System) và Ceph. Các tệp trong các hệ thống này thường được băm và lưu trữ với giá trị băm làm chỉ mục, giúp xác thực dữ liệu và tìm kiếm nhanh chóng.

- IPFS: Dữ liệu trong IPFS được chia thành các khối, mỗi khối được gán với một giá trị băm duy nhất. Điều này giúp đảm bảo tính toàn vẹn của tệp tin khi lưu trữ và chia sẻ trên mạng phân tán.

- Ceph: Hệ thống lưu trữ phân tán Ceph sử dụng hàm băm để theo dõi và kiểm tra tính toàn vẹn của các tệp trong hệ thống.

#### *1.2.4.8. Xác thực mạng và giao thức an toàn (Network Authentication and Secure Protocols)*

Hàm băm là thành phần quan trọng trong việc xác thực các giao thức an toàn như SSL/TLS và IPsec. Chúng giúp bảo vệ dữ liệu khỏi bị thay đổi và đảm bảo tính xác thực của các giao dịch.

- SSL/TLS: Các giao thức bảo mật web như SSL/TLS sử dụng hàm băm để xác thực dữ liệu trong quá trình kết nối giữa máy khách và máy chủ. SHA-256 là một trong các hàm băm phổ biến trong SSL/TLS.

+ IPsec: Trong các mạng riêng ảo (VPN), IPsec sử dụng HMAC để bảo vệ tính toàn vẹn và xác thực của các gói dữ liệu trong khi truyền tải qua mạng.

#### *1.2.4.9. Phát hiện gian lận và chống tấn công (Fraud Detection and Anti-Attack)*

Hàm băm có thể giúp phát hiện các hành vi gian lận và các cuộc tấn công trong các hệ thống bảo mật. Ví dụ, trong các hệ thống giao dịch tài chính, hàm băm có thể được sử dụng để kiểm tra các giao dịch nhằm phát hiện các hoạt động bất thường.

- Hệ thống thanh toán trực tuyến: Hệ thống có thể sử dụng hàm băm để kiểm tra các giao dịch bất hợp pháp hoặc gian lận trong các giao dịch tài chính.

- Phát hiện tấn công từ chối dịch vụ (DDoS): Các công cụ bảo mật có thể sử dụng hàm băm để theo dõi các mô hình truy cập và phát hiện các cuộc tấn công từ chối dịch vụ (DDoS) hoặc các cuộc tấn công mạng khác.

Hàm băm có một loạt ứng dụng trong các lĩnh vực bảo mật và công nghệ thông tin hiện đại, từ bảo vệ dữ liệu cá nhân, xác thực thông điệp, đến bảo mật trong các giao dịch blockchain và hệ thống mạng. Việc hiểu rõ và ứng dụng đúng các hàm băm trong từng ngữ cảnh là yếu tố quan trọng để đảm bảo an toàn và tính toàn vẹn của hệ thống.

## 1.3. CHỮ KÝ SỐ

### 1.3.1. Khái niệm chữ ký số

Chữ ký số (Digital Signature) là một công nghệ bảo mật dùng để xác thực và bảo vệ tính toàn vẹn của thông tin trong môi trường điện tử. Nó giống như một chữ ký viết tay, nhưng được sử dụng trong các giao dịch và tài liệu điện tử. Chữ ký số giúp đảm bảo rằng thông điệp hoặc tài liệu không bị thay đổi sau khi ký và xác nhận rằng người ký là chính chủ thể mà họ tuyên bố.

Chữ ký số không chỉ giúp xác thực danh tính của người gửi mà còn chứng minh rằng dữ liệu không bị sửa đổi sau khi chữ ký được tạo ra. Để thực hiện ký số, sử dụng các thuật toán mật mã, đặc biệt là mã hóa bất đối xứng, bao gồm khóa công khai và khóa riêng.

### 1.3.2. Nguyên lý hoạt động của chữ ký số

Chữ ký số (Digital Signature) hoạt động dựa trên nguyên lý của mã hóa bất đối xứng (asymmetric cryptography) và hàm băm. Quá trình này có mục đích chính là xác thực danh tính của người ký và đảm bảo rằng nội dung của tài liệu hoặc thông điệp không bị thay đổi sau khi ký.

Chữ ký số gồm hai bước chính: tạo chữ ký số và kiểm tra chữ ký số. Dưới đây là mô tả chi tiết về nguyên lý hoạt động của nó:

**a. Tạo chữ ký số**

Quá trình tạo chữ ký số của người ký có thể được mô tả qua các bước sau:

Bước 1: Tạo giá trị băm của thông điệp

Người gửi (hoặc người ký) sẽ tính toán giá trị băm của thông điệp hoặc tài liệu cần ký (ví dụ: một hợp đồng điện tử). Để tính toán giá trị băm, người ký sẽ sử dụng một

hàm băm (ví dụ: SHA-256). Giá trị băm này là một chuỗi số và ký tự có độ dài cố định, đại diện cho nội dung của thông điệp.

Bước 2: Mã hóa giá trị băm bằng khóa riêng

Sau khi tính toán giá trị băm, người ký sẽ sử dụng khóa riêng của mình (private key) để mã hóa giá trị băm này. Quá trình mã hóa này tạo ra chữ ký số. Chữ ký số này sẽ được đính kèm cùng với thông điệp gửi đến người nhận. Chỉ người sở hữu khóa riêng mới có thể mã hóa và tạo ra chữ ký số, đảm bảo rằng chữ ký này là duy nhất và không thể giả mạo.

Bước 3: Gửi thông điệp và chữ ký số

Người ký sẽ gửi cả thông điệp và chữ ký số đến người nhận. Lúc này, người nhận sẽ có thể sử dụng chữ ký số và thông điệp để xác minh tính hợp lệ của thông điệp và người ký.

**b. Kiểm tra chữ ký số**

Khi người nhận nhận được thông điệp và chữ ký số, họ sẽ thực hiện các bước sau để kiểm tra tính hợp lệ của chữ ký:

Bước 1: Tạo giá trị băm của thông điệp nhận được

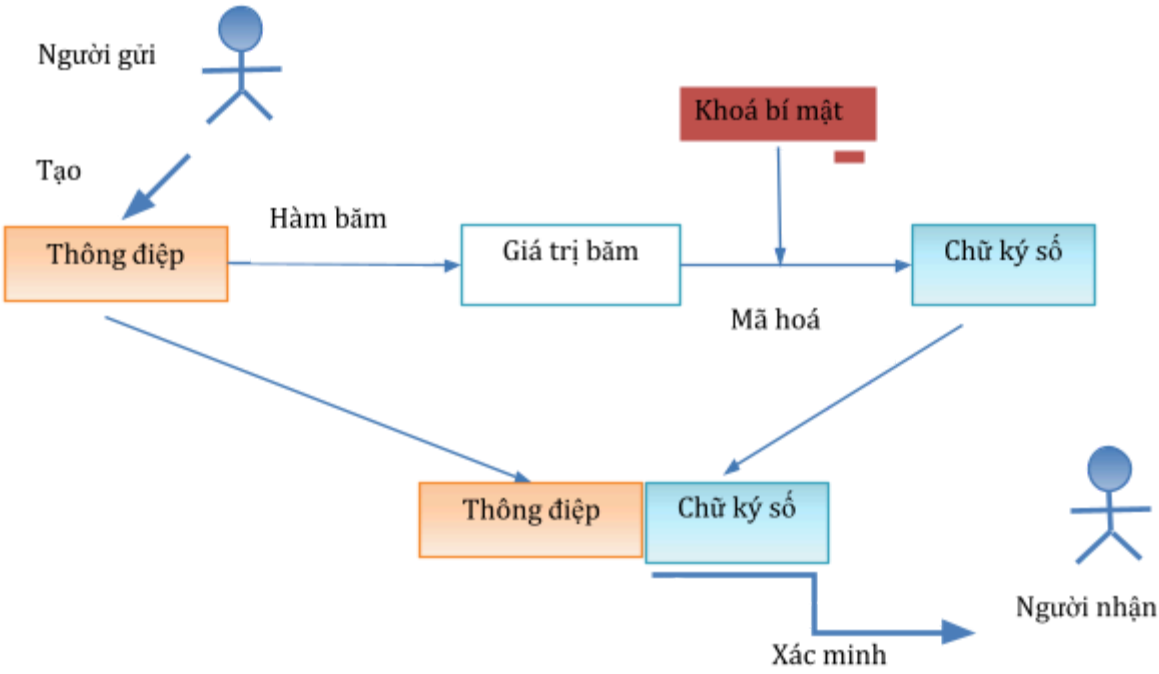
Người nhận sẽ tính toán lại giá trị băm của thông điệp hoặc tài liệu mà họ nhận được bằng cách sử dụng cùng một hàm băm mà người ký đã sử dụng (ví dụ: SHA-256). Điều này tạo ra một giá trị băm mới của thông điệp.

Bước 2: Giải mã chữ ký số bằng khóa công khai

Người nhận sẽ sử dụng khóa công khai của người ký (public key) để giải mã chữ ký số mà họ nhận được. Việc giải mã sẽ trả lại giá trị băm gốc mà người ký đã mã hóa.

Bước 3: So sánh giá trị băm

Người nhận sẽ so sánh giá trị băm mà họ tính toán được (từ thông điệp) với giá trị băm mà họ thu được từ việc giải mã chữ ký số. Nếu hai giá trị băm này khớp nhau, điều đó có nghĩa là thông điệp không bị thay đổi và chữ ký là hợp lệ. Nếu không khớp, nghĩa là thông điệp đã bị thay đổi hoặc chữ ký không hợp lệ.




Hình 1: Sơ đồ minh họa quy trình chữ ký số

### 1.3.3. Vai trò của chữ ký số

Chữ ký số dùng cho các văn bản số, cho biết toàn bộ văn bản đã được ký bởi người ký. Và người khác có thể xác minh điều này. Chữ ký số tương tự như chữ ký thông thường, đảm bảo nội dung tài liệu là đáng tin cậy, chính xác, không hề thay đổi trên đường truyền và cho biết người tạo ra tài liệu là ai.

Tuy nhiên, chữ ký số khác chữ ký thường, vì nó tuỳ thuộc vào văn bản. Chữ ký số sẽ thay đổi theo văn bản còn chữ ký thường thì không hề thay đổi.

Chữ ký số được sử dụng để cung cấp chứng thực chủ sở hữu, tính toàn vẹn dữ liệu và chống chối bỏ nguồn gốc trong rất nhiều các lĩnh vực.

### 1.3.4. Các thành phần của chữ ký số

Chữ ký số (Digital Signature) bao gồm các thành phần chính sau, giúp xác thực danh tính của người ký và bảo vệ tính toàn vẹn của thông điệp. Các thành phần này được tạo ra và sử dụng trong quá trình ký và kiểm tra chữ ký số:

#### a. Khóa riêng (Private Key)

Khóa riêng là một phần của cặp khóa bất đối xứng, chỉ thuộc về người ký và được sử dụng để mã hóa giá trị băm của thông điệp.

Khóa riêng giúp người ký tạo ra chữ ký số bằng cách mã hóa giá trị băm của thông điệp. Khóa này được bảo vệ nghiêm ngặt và không được chia sẻ với bất kỳ ai.

Khóa riêng phải được giữ bí mật tuyệt đối. Nếu bị lộ, kẻ tấn công có thể giả mạo chữ ký số của người ký.

**b. Khóa công khai (Public Key)**

Khóa công khai là phần còn lại của cặp khóa bất đối xứng và được chia sẻ công khai với tất cả người nhận.

Khóa công khai dùng để giải mã chữ ký số mà người nhận nhận được. Bằng cách sử dụng khóa công khai, người nhận có thể kiểm tra xem chữ ký số có hợp lệ hay không và liệu thông điệp có bị thay đổi không.

Khóa công khai có thể được phát tán rộng rãi, miễn là khóa riêng vẫn được bảo vệ an toàn.

**c. Thông điệp hoặc tài liệu (Message/Document)**

Đây là thông điệp hoặc tài liệu mà người ký muốn bảo vệ và xác thực.

Thông điệp là nội dung mà người ký muốn truyền đạt. Chữ ký số đảm bảo rằng thông điệp này không bị thay đổi sau khi ký và xác nhận danh tính của người ký.

Tính toàn vẹn của thông điệp được bảo vệ nhờ vào việc sử dụng hàm băm và mã hóa giá trị băm.

**d. Giá trị băm (Hash Value)**

Giá trị băm là kết quả của quá trình sử dụng hàm băm (ví dụ: SHA-256) trên thông điệp hoặc tài liệu. Đây chính là đại diện số học của thông điệp, được mã hóa để tạo ra chữ ký số.

Giá trị băm là đại diện số học của thông điệp, giúp giảm thiểu kích thước của dữ liệu và tăng tốc quá trình ký. Nó cũng đảm bảo rằng bất kỳ thay đổi nhỏ nào trong thông điệp sẽ dẫn đến sự thay đổi đáng kể trong giá trị băm.

Giá trị băm không thể đảo ngược và không thể bị giả mạo. Điều này giúp đảm bảo tính toàn vẹn của thông điệp trong quá trình truyền tải.

**e. Chữ ký số (Digital Signature)**

Chữ ký số là kết quả của việc mã hóa giá trị băm của thông điệp bằng khóa riêng của người ký. Dùng để xác nhận tính hợp lệ của thông điệp

Chữ ký số đảm bảo tính xác thực của người ký và bảo vệ tính toàn vẹn của thông điệp. Chữ ký này sẽ được gửi kèm với thông điệp để người nhận có thể kiểm tra tính hợp lệ.

Chữ ký số không thể giả mạo, vì chỉ có khóa riêng của người ký mới có thể tạo ra chữ ký số. Nếu thông điệp bị thay đổi, chữ ký số sẽ không còn hợp lệ.

**f. Hàm băm (Hash Function)**

Hàm băm là một thuật toán mã hóa một chiều được sử dụng để chuyển đổi thông điệp thành giá trị băm có độ dài cố định. Thuật toán dùng để tạo giá trị băm của thông điệp, bảo vệ tính toàn vẹn của nó

Hàm băm đảm bảo tính toàn vẹn của thông điệp. Nếu thông điệp bị thay đổi, giá trị băm sẽ thay đổi và người nhận sẽ nhận ra sự thay đổi này khi kiểm tra chữ ký số.

Các hàm băm phổ biến như SHA-256, SHA-3 có tính chất mạnh mẽ như không có va chạm (collision-resistant) và một chiều (one-way function), giúp bảo vệ thông tin khỏi bị giả mạo.

Chữ ký số là một công cụ mạnh mẽ trong việc xác thực và bảo vệ thông tin trong môi trường điện tử, giúp đảm bảo tính toàn vẹn của dữ liệu và xác minh danh tính của người ký. Các thành phần của chữ ký số, bao gồm khóa riêng, khóa công khai, giá trị băm, và hàm băm, cùng nhau tạo ra một hệ thống bảo mật mạnh mẽ cho các giao dịch và tài liệu điện tử.

### 1.3.5. Chi tiết quy trình tạo và xác minh chữ ký số

#### *1.3.5.1. Tạo chữ ký số*

Quá trình tạo chữ ký số bao gồm ba bước chính: tạo hàm băm, mã hóa hàm băm, và ghép nối thông tin. Các bước này được mô tả chi tiết như sau:

Bước 1: Tạo hàm băm (Hashing)

Đầu vào: Dữ liệu cần ký (thông điệp hoặc tài liệu số).

Thực hiện: Áp dụng một hàm băm mật mã (cryptographic hash function) lên dữ liệu đầu vào. Hàm băm phổ biến thường được sử dụng bao gồm SHA-256, SHA-3, hoặc SHA-512.

Kết quả: Một giá trị băm có kích thước cố định, gọi là "thông điệp băm" (message digest). Giá trị băm này là duy nhất đối với nội dung của thông điệp, tức là nếu thông điệp thay đổi, giá trị băm cũng sẽ thay đổi theo.

Ví dụ: Giả sử thông điệp cần ký là "Hello, World!".

Áp dụng hàm băm SHA-256, kết quả sẽ là giá trị băm (hash value): a591a6d40bf420404a011733cfb7b190d62c65bf0bcda32b59f1b4d89013dd31.

Bước 2: Mã hóa hàm băm

Đầu vào: Thông điệp băm từ bước 1 và khóa riêng (private key) của người ký.

Thực hiện: Sử dụng thuật toán mã hóa bất đối xứng (như RSA hoặc ECDSA) để mã hóa giá trị băm. Khóa riêng được sử dụng để thực hiện mã hóa.

Kết quả: Chữ ký số, là một giá trị được mã hóa dựa trên thông điệp băm và khóa riêng của người ký.

Ví dụ: Giả sử sử dụng thuật toán RSA và khóa riêng của người ký. Giá trị băm từ bước 1 được mã hóa với khóa riêng, tạo ra chữ ký số (digital signature).

Nếu sử dụng một khóa riêng, chữ ký số sau khi mã hóa có thể là: 93844c24b8f502bc32ab8a... (giá trị này sẽ khác nhau tùy thuộc vào khóa và thuật toán mã hóa).

Bước 3: Ghép nối thông tin

Đầu vào: Thông điệp gốc và chữ ký số.

Thực hiện: Chữ ký số được ghép cùng với thông điệp gốc để tạo thành gói dữ liệu đã ký (signed data).

Kết quả: Gói dữ liệu chứa thông điệp và chữ ký số, có thể gửi đi hoặc lưu trữ.

Thông điệp "Hello, World!" được ghép với chữ ký số 93844c24b8f502bc32ab8a..., tạo thành gói dữ liệu đã ký.

#### *1.3.5.2. Xác minh chữ ký số*

Quá trình xác minh chữ ký số liên quan đến việc xác minh xem chữ ký có chính xác và không bị giả mạo hay không. Quy trình xác minh bao gồm các bước sau:

Bước 1: Tạo lại hàm băm từ thông điệp

Đầu vào: Thông điệp gốc nhận được từ gói dữ liệu đã ký.

Thực hiện: Áp dụng cùng một hàm băm đã sử dụng khi tạo chữ ký để tạo lại giá trị băm từ thông điệp.

Kết quả: Giá trị băm mới từ thông điệp nhận được.

Ví dụ: Người nhận lấy thông điệp "Hello, World!" từ gói dữ liệu đã ký và áp dụng hàm băm SHA-256 lên thông điệp. Kết quả là giá trị băm:

a591a6d40bf420404a011733cfb7b190d62c65bf0bcda32b59f1b4d89013dd31.

Bước 2: Giải mã chữ ký số

Đầu vào: Chữ ký số và khóa công khai (public key) của người ký.

Thực hiện: Sử dụng khóa công khai và thuật toán mã hóa bất đối xứng để giải mã chữ ký số, thu được giá trị băm đã được mã hóa trong quá trình tạo chữ ký.

Kết quả: Giá trị băm đã được mã hóa trước đó.

Ví dụ: Người nhận sử dụng khóa công khai của người ký để giải mã chữ ký số 93844c24b8f502bc32ab8a.... Quá trình giải mã sẽ thu được giá trị băm gốc (được mã hóa từ thông điệp ban đầu):

a591a6d40bf420404a011733cfb7b190d62c65bf0bcda32b59f1b4d89013dd31.

Bước 3: So sánh hai giá trị băm

Đầu vào: Giá trị băm từ thông điệp nhận được (bước 2.1) và giá trị băm giải mã từ chữ ký (bước 2.2).

Thực hiện: So sánh hai giá trị băm này.

Kết quả: Nếu hai giá trị băm giống nhau, chữ ký số được xác minh là hợp lệ; nếu không, chữ ký không hợp lệ hoặc thông điệp đã bị thay đổi.

#### *1.3.5.3. Ví dụ cụ thể*

Giả sử Alice muốn ký một tài liệu bằng chữ ký số để gửi cho Bob:

Alice:

+ Sử dụng hàm băm SHA-256 trên tài liệu của cô và nhận được giá trị băm.

+ Sử dụng khóa riêng RSA để mã hóa giá trị băm và tạo chữ ký số.

+ Gửi tài liệu cùng với chữ ký số cho Bob.

Bob:

+ Nhận tài liệu và chữ ký số từ Alice.

+ Tính toán giá trị băm của tài liệu mà anh nhận được.

+ Sử dụng khóa công khai của Alice để giải mã chữ ký số và thu được giá trị băm gốc.

+ So sánh hai giá trị băm. Nếu chúng giống nhau, Bob biết rằng tài liệu không bị thay đổi và chữ ký số là hợp lệ, xác nhận Alice là người đã ký.

### 1.3.6. Ứng dụng chữ ký số

Chữ ký số có nhiều ứng dụng trong các lĩnh vực khác nhau, nhờ tính năng xác thực, bảo mật và đảm bảo tính toàn vẹn của dữ liệu. Dưới đây là các ứng dụng quan trọng của chữ ký số:

**a. Giao dịch điện tử và thương mại điện tử**

Chữ ký số được sử dụng để ký các hợp đồng điện tử, hóa đơn điện tử, và các tài liệu liên quan đến giao dịch kinh doanh. Nó đảm bảo rằng người gửi và người nhận đều có thể xác minh danh tính của nhau và nội dung không bị thay đổi sau khi ký.

Ví dụ: Trong các hệ thống thương mại điện tử, doanh nghiệp và khách hàng sử dụng chữ ký số để ký hợp đồng mua bán, thanh toán và các thỏa thuận pháp lý khác.

**b. Chính phủ điện tử**

Chữ ký số là một thành phần quan trọng trong việc triển khai chính phủ điện tử, giúp xác thực danh tính công dân và các cơ quan chính phủ trong các giao dịch trực tuyến.

Ví dụ: Công dân có thể nộp thuế, đăng ký dịch vụ công, hoặc ký các tài liệu hành chính trực tuyến mà không cần phải đến các cơ quan chính phủ.

**c. Hóa đơn điện tử (e-Invoice)**

Chữ ký số được sử dụng để ký và xác thực tính hợp lệ của hóa đơn điện tử, đảm bảo rằng hóa đơn được gửi đi không thể bị sửa đổi và bên nhận có thể tin tưởng vào tính xác thực của hóa đơn.

Ví dụ: Trong các hệ thống kế toán, doanh nghiệp ký hóa đơn điện tử bằng chữ ký số để gửi đến khách hàng và cơ quan thuế.

**d. Bảo mật email và tin nhắn**

Chữ ký số giúp bảo mật các email và tin nhắn quan trọng bằng cách đảm bảo rằng nội dung không bị thay đổi và danh tính người gửi được xác thực.

Ví dụ: Trong giao tiếp nội bộ của doanh nghiệp, chữ ký số đảm bảo tính bảo mật và toàn vẹn của thông tin nhạy cảm được truyền qua email.

**e. Chứng khoán và giao dịch tài chính**

Chữ ký số được sử dụng trong các giao dịch chứng khoán và tài chính để xác thực danh tính của các bên tham gia và đảm bảo rằng các tài liệu tài chính được ký là chính xác và không thể bị thay đổi.

Ví dụ: Các giao dịch mua bán cổ phiếu trực tuyến thường yêu cầu chữ ký số để đảm bảo tính an toàn và hợp pháp của giao dịch.

**f. Ngân hàng trực tuyến**

Trong ngân hàng trực tuyến, chữ ký số giúp xác thực các giao dịch tài chính, chẳng hạn như chuyển khoản, vay vốn, và mở tài khoản. Điều này giúp giảm thiểu rủi ro gian lận và bảo vệ quyền lợi của khách hàng.

Ví dụ: Khi khách hàng thực hiện giao dịch lớn hoặc ký hợp đồng vay tiền trực tuyến, họ có thể sử dụng chữ ký số để xác nhận.

**g. Ký hợp đồng từ xa**

Chữ ký số cho phép các bên ký kết hợp đồng từ xa mà không cần gặp mặt trực tiếp. Điều này giúp tiết kiệm thời gian và chi phí trong quá trình ký kết hợp đồng.

Ví dụ: Trong các công ty đa quốc gia, các đối tác ở các quốc gia khác nhau có thể sử dụng chữ ký số để ký kết hợp đồng mà không cần gặp trực tiếp.

**h. Xác thực phần mềm**

Các nhà phát triển phần mềm sử dụng chữ ký số để ký các ứng dụng, nhằm xác nhận rằng phần mềm đó là đáng tin cậy và không bị thay đổi kể từ khi phát hành.

Ví dụ: Khi tải xuống một ứng dụng từ nhà phát triển đáng tin cậy, chữ ký số trên phần mềm giúp người dùng xác nhận rằng nó không bị chỉnh sửa hoặc chèn mã độc.

**i. Blockchain và tiền mã hóa**

Chữ ký số là một phần không thể thiếu trong các giao dịch blockchain và tiền mã hóa, giúp xác thực quyền sở hữu và tính toàn vẹn của các giao dịch.

Ví dụ: Trong hệ thống blockchain của Bitcoin và Ethereum, chữ ký số giúp xác nhận rằng một giao dịch được thực hiện bởi chủ sở hữu thực sự của địa chỉ ví tiền mã hóa.

**j. Y tế điện tử**

Trong lĩnh vực y tế, chữ ký số giúp xác thực hồ sơ bệnh nhân và các tài liệu y khoa điện tử, đảm bảo tính bảo mật và toàn vẹn của thông tin y tế.

Ví dụ: Bác sĩ có thể sử dụng chữ ký số để ký kết hồ sơ khám bệnh và đơn thuốc điện tử, giúp bệnh nhân dễ dàng lưu trữ và quản lý hồ sơ y tế trực tuyến.

### 1.3.7. Lợi ích và hạn chế của chữ ký số

Chữ ký số mang lại nhiều lợi ích trong việc bảo mật và xác thực giao dịch trực tuyến, đặc biệt trong các lĩnh vực thương mại điện tử, chính phủ điện tử và tài chính. Tuy nhiên, việc triển khai chữ ký số cũng gặp phải một số hạn chế liên quan đến chi phí, tính phức tạp, và các vấn đề pháp lý. Việc sử dụng chữ ký số hiệu quả đòi hỏi phải cân nhắc kỹ lưỡng giữa các lợi ích và hạn chế này.

#### *1.3.7.1. Lợi ích của chữ ký số*

**a. Tính bảo mật cao**

Chữ ký số sử dụng mã hóa bất đối xứng, đảm bảo tính bảo mật cao nhờ việc tách biệt giữa khóa công khai và khóa riêng. Chỉ có người giữ khóa riêng mới có thể tạo ra chữ ký số, trong khi bất kỳ ai cũng có thể sử dụng khóa công khai để xác minh. Chữ ký số giảm thiểu rủi ro giả mạo hoặc chỉnh sửa thông tin trong quá trình truyền tải.

**b Xác thực danh tính**

Chữ ký số giúp xác thực danh tính của người ký, đảm bảo rằng tài liệu hoặc giao dịch được thực hiện bởi đúng đối tượng. Chữ ký số giúp đảm bảo tính tin cậy trong giao dịch điện tử, ngăn chặn gian lận và giả mạo.

**c. Tính toàn vẹn dữ liệu**

Hàm băm trong quy trình tạo chữ ký số giúp phát hiện bất kỳ sự thay đổi nào đối với thông tin sau khi ký. Nếu dữ liệu bị chỉnh sửa sau khi ký, chữ ký số sẽ trở nên vô hiệu. Nó đảm bảo rằng thông tin không bị thay đổi trong quá trình truyền tải hoặc lưu trữ.

**d. Tính pháp lý**

Chữ ký số có tính pháp lý tại nhiều quốc gia, được coi là bằng chứng hợp pháp tương đương với chữ ký tay trong nhiều quy định và luật pháp liên quan đến giao dịch điện tử. Chữ ký số giúp người ký có thể ký kết hợp đồng và các thỏa thuận pháp lý từ xa, giúp tăng cường hiệu quả và tiết kiệm thời gian.

**e. Tiết kiệm chi phí và thời gian**

Sử dụng chữ ký số giúp loại bỏ nhu cầu in ấn, chuyển phát tài liệu vật lý, cũng như việc ký kết và xác minh thủ công. Chữ ký số giúp tăng cường tốc độ xử lý các giao dịch và giảm chi phí vận hành liên quan đến giấy tờ và chuyển phát.

**g. Dễ dàng tích hợp với các hệ thống số hóa**

Chữ ký số có thể dễ dàng tích hợp vào các hệ thống quản lý tài liệu, giao dịch điện tử, và các nền tảng thương mại điện tử. Nó giúp nâng cao hiệu suất hoạt động và tự động hóa nhiều quy trình kinh doanh.

#### *1.3.7.1. Hạn chế của chữ ký số*

**a. Chi phí thiết lập ban đầu**

Việc triển khai chữ ký số yêu cầu thiết lập hạ tầng khóa công khai (PKI), bao gồm việc cấp và quản lý chứng thư số (digital certificate) từ các cơ quan chứng thực (CA - Certificate Authority). Do đó chi phí thiết lập và duy trì hệ thống PKI có thể cao, đặc biệt là đối với các tổ chức nhỏ và vừa.

**b. Phụ thuộc vào cơ quan chứng thực (CA)**

Chữ ký số yêu cầu sự tin cậy vào cơ quan chứng thực (CA) để phát hành và xác nhận chứng thư số của người dùng. Do vậy nếu cơ quan chứng thực bị xâm nhập hoặc ngừng hoạt động, tính bảo mật và tính xác thực của chữ ký số có thể bị ảnh hưởng.

**c. Khó khăn trong việc sử dụng cho người không quen thuộc với công nghệ**

Người dùng không quen thuộc với các khái niệm liên quan đến chữ ký số, chứng thư số, và mã hóa có thể gặp khó khăn trong việc tạo, quản lý, và sử dụng chữ ký số. Vì vậy cần đào tạo và hỗ trợ kỹ thuật để người dùng nắm bắt cách sử dụng hiệu quả chữ ký số.

**d. Thời hạn của chứng thư số**

Chứng thư số có thời hạn hiệu lực, và sau khi hết hạn, người dùng cần gia hạn hoặc đăng ký lại chứng thư số mới. Do đó, việc quản lý thời hạn và gia hạn chứng thư số có thể gây ra sự bất tiện, đặc biệt nếu quá trình gia hạn không được thực hiện kịp thời.

**e. Vấn đề pháp lý quốc tế**

Mặc dù chữ ký số được công nhận tại nhiều quốc gia, nhưng vẫn tồn tại sự khác biệt trong quy định pháp lý giữa các quốc gia về việc công nhận chữ ký số, đặc biệt là trong các giao dịch quốc tế. Như vậy một hạn chế là giao dịch giữa các bên ở các quốc gia khác nhau có thể gặp rào cản pháp lý liên quan đến sự công nhận và sử dụng chữ ký số.

**g. Rủi ro mất khóa riêng**

Chữ ký số dựa trên việc giữ bí mật khóa riêng của người dùng. Nếu khóa riêng bị mất hoặc bị đánh cắp, chữ ký số của người đó có thể bị giả mạo. Do đó, người dùng cần quản lý khóa riêng một cách cẩn thận để tránh mất mát hoặc lạm dụng.

## 1.4. HỆ PHI TẬP TRUNG

### 1.4.1. Khái niệm hệ phi tập trung (Decentralized System)

Hệ phi tập trung là một hệ thống mà quyền kiểm soát và ra quyết định không tập trung vào một thực thể duy nhất, mà được phân chia giữa nhiều nút (nodes) hoặc thành phần riêng lẻ. Trong hệ thống này, không có một đơn vị quyền lực trung tâm có toàn quyền điều khiển, thay vào đó các nút trong hệ thống có vai trò tương đương và hoạt động độc lập hoặc cùng nhau để đạt được mục tiêu chung.

Hệ phi tập trung thường được áp dụng trong các lĩnh vực như blockchain, mạng máy tính, tổ chức tự trị phi tập trung (DAO), và quản trị phi tập trung. Trong hệ thống này, việc phân phối quyền kiểm soát và dữ liệu giúp giảm thiểu rủi ro tập trung, tăng cường bảo mật, và cải thiện tính minh bạch.

### 1.4.2. Đặc điểm của hệ phi tập trung

Không có trung tâm kiểm soát duy nhất: Quyền ra quyết định và dữ liệu không phụ thuộc vào một thực thể duy nhất.

Tự quản lý và phân tán: Các nút trong hệ thống tự hoạt động và quản lý mà không cần sự can thiệp của trung tâm.

Tăng cường khả năng chịu lỗi: Nếu một hoặc nhiều nút gặp sự cố, hệ thống vẫn có thể tiếp tục hoạt động, giúp tăng độ bền vững.

Minh bạch và không dễ bị giả mạo: Nhờ vào tính phân tán, mọi thay đổi trong hệ thống đều có thể được kiểm tra và theo dõi.

### 1.4.3. Cấu trúc và mô hình của hệ phi tập trung

Hệ phi tập trung có cấu trúc khác biệt so với hệ tập trung, vì quyền kiểm soát và tài nguyên không tập trung vào một thực thể duy nhất mà được phân phối giữa nhiều nút (nodes). Các thành phần chính trong cấu trúc của một hệ phi tập trung:

**a. Các nút (Nodes)**

Nút (node) là các thành phần cơ bản trong hệ phi tập trung, có thể là máy tính, máy chủ, hoặc thiết bị cá nhân, mỗi nút có khả năng xử lý và lưu trữ dữ liệu độc lập. Các nút có quyền và trách nhiệm tương đương nhau.

Các nút thực hiện các tác vụ như xác minh, xử lý giao dịch, lưu trữ dữ liệu, và đảm bảo tính toàn vẹn của hệ thống. Một số hệ thống có thể có các nút chuyên biệt như:

- Nút xác minh (Validators): Trong blockchain, các nút này tham gia vào quá trình xác thực giao dịch.

- Nút lưu trữ (Storage Nodes): Lưu trữ dữ liệu và phân phối lại khi được yêu cầu, ví dụ như trong các hệ thống mạng ngang hàng (P2P).

**b. Giao thức đồng thuận (Consensus Protocol)**

Trong hệ phi tập trung, các nút phải đồng thuận về trạng thái của hệ thống mà không cần sự can thiệp từ một thực thể trung tâm. Để làm điều này, một giao thức đồng thuận được sử dụng nhằm đạt được sự thống nhất giữa các nút về các hành động và thay đổi trong hệ thống.

Ví dụ: Các giao thức đồng thuận[2] phổ biến trong blockchain bao gồm:

- Proof of Work (PoW): Các nút giải các bài toán mật mã để xác nhận giao dịch (sử dụng trong Bitcoin).

- Proof of Stake (PoS): Các nút đóng góp theo số lượng tài sản kỹ thuật số để xác nhận giao dịch (sử dụng trong Ethereum 2.0).

- Delegated Proof of Stake (DPoS): Các nút bỏ phiếu chọn ra các đại diện xác thực giao dịch.

**c. Sổ cái phân tán[3] (Distributed Ledger)**

Sổ cái phân tán là nơi lưu trữ tất cả các giao dịch hoặc sự kiện đã được xác nhận bởi các nút trong hệ thống. Mỗi nút có một bản sao của sổ cái này và có quyền kiểm tra các giao dịch bất kỳ lúc nào.

Sổ cái phân tán đảm bảo tính minh bạch và toàn vẹn dữ liệu, vì mọi thay đổi đều được ghi lại và phân phối cho tất cả các nút.

Ví dụ: Blockchain là một dạng sổ cái phân tán phổ biến, trong đó mỗi khối (block) chứa các giao dịch và được liên kết với khối trước đó.

**d. Cơ chế truyền thông giữa các nút (Peer-to-Peer Communication)**

Các nút trong hệ phi tập trung giao tiếp trực tiếp với nhau mà không cần thông qua một máy chủ trung tâm. Giao tiếp giữa các nút thường diễn ra thông qua các giao thức mạng ngang hàng (P2P).

Cơ chế truyền thông giúp đảm bảo rằng các thông điệp và dữ liệu có thể được trao đổi nhanh chóng giữa các nút. Điều này giúp hệ thống tiếp tục hoạt động ngay cả khi một số nút bị ngắt kết nối.

[2] Khái niệm này được trình bày chi tiết trong chương 4 của giáo trình này
[3] Khái niệm này được trình bày chi tiết trong chương 2 của giáo trình này

Ví dụ: Trong các mạng P2P như BitTorrent, các máy tính chia sẻ tệp trực tiếp với nhau thay vì thông qua máy chủ trung tâm.

**e. Tài nguyên phân tán (Distributed Resources)**

Tài nguyên trong hệ phi tập trung không tập trung tại một nơi mà được phân tán giữa các nút. Điều này có thể bao gồm dữ liệu, băng thông, khả năng xử lý hoặc tài sản kỹ thuật số.

Phân tán tài nguyên giúp tăng cường khả năng chịu lỗi của hệ thống và đảm bảo tính sẵn có cao hơn so với hệ thống tập trung.

Ví dụ: Hệ thống lưu trữ phi tập trung như IPFS phân tán dữ liệu trên nhiều nút thay vì lưu trữ toàn bộ tệp tin tại một vị trí duy nhất.

**f. Hợp đồng thông minh[4](Smart Contracts) (tùy chọn)**

Hợp đồng thông minh là các chương trình tự động hóa các thỏa thuận hoặc giao dịch khi các điều kiện được đáp ứng. Chúng thường được triển khai trên các nền tảng phi tập trung như blockchain.

Hợp đồng thông minh cho phép tự động hóa các quy trình và thực hiện giao dịch mà không cần trung gian.

Ví dụ: Trên blockchain Ethereum, các hợp đồng thông minh có thể tự động thực hiện các giao dịch khi các điều kiện xác định trước được đáp ứng.

**g. Bảo mật và mật mã hóa (Security and Cryptography)**

Hệ phi tập trung dựa vào các kỹ thuật mật mã để đảm bảo rằng thông tin và giao dịch được bảo vệ khỏi các hành vi giả mạo và xâm nhập. Điều này bao gồm mã hóa dữ liệu, chữ ký số, và các hàm băm để bảo vệ tính toàn vẹn.

Mật mã giúp xác thực danh tính, bảo vệ quyền riêng tư, và đảm bảo tính toàn vẹn của dữ liệu trong hệ thống.

### 1.4.4. Ưu điểm của cấu trúc phi tập trung và hệ phi tập trung

Với đặc điểm, mô hình như trên cấu trúc phi tập trung có các ưu điểm sau:

- Chống lỗi và tấn công: Không có điểm thất bại duy nhất, hệ thống vẫn có thể hoạt động dù một số nút bị xâm nhập.

- Tăng cường bảo mật: Sử dụng mật mã mạnh và phân tán giúp ngăn ngừa các hành vi giả mạo và đảm bảo an toàn dữ liệu.

[4] Khái niệm hợp đồng thông minh được trình bày chi tiết ở chương xx của giáo trình này

- Tính minh bạch cao: Tất cả các giao dịch và thay đổi đều có thể kiểm tra bởi các nút, giúp hệ thống trở nên minh bạch.

Cùng với đó hệ phi tập trung mang lại các lợi ích như:

- Bảo mật và riêng tư cao: Vì không có một điểm tập trung duy nhất để tấn công, hệ thống phi tập trung khó bị xâm phạm hoặc lạm dụng dữ liệu hơn so với hệ thống tập trung.

- Chống kiểm duyệt: Không có cơ quan trung tâm nào có thể can thiệp hoặc kiểm duyệt thông tin trong hệ thống.

- Tăng cường minh bạch và tin cậy: Tất cả các giao dịch hoặc thay đổi trong hệ thống đều có thể được ghi lại và kiểm chứng bởi tất cả các nút.

- Giảm rủi ro lỗi hệ thống: Hệ thống phi tập trung thường có khả năng chịu lỗi cao hơn, bởi không phụ thuộc vào một điểm duy nhất.

### 1.4.5. Ứng dụng của hệ phi tập trung

Hệ phi tập trung (decentralized system) có rất nhiều ứng dụng trong các lĩnh vực khác nhau nhờ vào tính bảo mật, minh bạch, khả năng chịu lỗi cao, và khả năng loại bỏ sự phụ thuộc vào các trung gian. Dưới đây là các ứng dụng phổ biến của hệ phi tập trung:

***1. Blockchain và tiền mã hóa***

Blockchain là một dạng hệ phi tập trung trong đó các giao dịch được lưu trữ trong các khối và liên kết với nhau thành chuỗi. Không có một cơ quan trung ương nào kiểm soát toàn bộ hệ thống, mà thay vào đó các nút mạng (nodes) cùng xác minh và duy trì sổ cái. Ví dụ:

- Bitcoin: Một loại tiền mã hóa phi tập trung, nơi các giao dịch được xác minh thông qua cơ chế đồng thuận Proof of Work (PoW) và không có ngân hàng hoặc tổ chức tài chính trung gian.

- Ethereum: Nền tảng blockchain hỗ trợ hợp đồng thông minh (smart contracts), cho phép thực hiện các giao dịch phi tập trung tự động mà không cần trung gian.

***2. Tổ chức tự trị phi tập trung (DAO - Decentralized Autonomous Organization)***

DAO là các tổ chức tự vận hành mà không có một quản lý trung tâm. Các quy tắc và quá trình hoạt động của DAO được mã hóa trong các hợp đồng thông minh và các thành viên của DAO có quyền bỏ phiếu quyết định các vấn đề. Ví dụ:

- MakerDAO: Một tổ chức cho phép người dùng tạo ra stablecoin DAI thông qua hợp đồng thông minh mà không cần ngân hàng hoặc tổ chức tài chính.

- Aragon: Một nền tảng giúp dễ dàng tạo và quản lý các tổ chức phi tập trung trên blockchain.

### *3. Tài chính phi tập trung (DeFi - Decentralized Finance)*

DeFi là hệ sinh thái tài chính mà không cần đến các trung gian truyền thống như ngân hàng, công ty tài chính. Các giao dịch như vay, cho vay, mua bán, bảo hiểm, và nhiều dịch vụ tài chính khác được thực hiện thông qua hợp đồng thông minh. Ví dụ:

- Uniswap: Một sàn giao dịch phi tập trung (DEX) cho phép người dùng mua bán tiền mã hóa mà không cần qua trung gian.

- Aave: Nền tảng cho vay phi tập trung cho phép người dùng vay và cho vay tiền mã hóa mà không cần qua ngân hàng.

### *4. Mạng ngang hàng (P2P - Peer-to-Peer)*

Mạng P2P là mạng phi tập trung nơi các thiết bị có thể kết nối và chia sẻ tài nguyên trực tiếp với nhau mà không cần qua máy chủ trung tâm. Ví dụ:

- BitTorrent: Một giao thức chia sẻ tệp ngang hàng cho phép người dùng chia sẻ và tải xuống tệp mà không cần máy chủ tập trung.

- IPFS (InterPlanetary File System): Hệ thống lưu trữ và chia sẻ tệp phi tập trung, giúp cải thiện khả năng truy cập và bảo mật dữ liệu.

### *5. Truyền thông phi tập trung*

Các nền tảng truyền thông phi tập trung không phụ thuộc vào các nhà cung cấp dịch vụ trung gian, giúp tăng cường quyền riêng tư và tránh sự kiểm duyệt. Ví dụ:

- Matrix: Một giao thức truyền thông mã nguồn mở phi tập trung cho phép các hệ thống truyền thông kết nối với nhau.

- Signal: Mặc dù chưa hoàn toàn phi tập trung, Signal sử dụng mã hóa đầu cuối (end-to-end encryption) và đang phát triển theo hướng tăng cường tính phi tập trung.

### *6. Mạng xã hội phi tập trung*

Mạng xã hội phi tập trung cho phép người dùng kiểm soát hoàn toàn nội dung và dữ liệu cá nhân của họ mà không phụ thuộc vào các công ty hoặc nhà cung cấp dịch vụ trung gian. Điều này giúp tránh sự kiểm duyệt và bảo vệ quyền riêng tư tốt hơn. Ví dụ:

- Mastodon: Mạng xã hội phi tập trung nơi người dùng có thể tự tạo máy chủ riêng để quản lý và kiểm soát dữ liệu cá nhân.

- Steemit: Một nền tảng truyền thông xã hội phi tập trung chạy trên blockchain Steem, nơi người dùng được thưởng bằng tiền mã hóa khi đóng góp nội dung.

***7. Lưu trữ dữ liệu phi tập trung***

Hệ thống lưu trữ dữ liệu phi tập trung giúp phân tán dữ liệu trên nhiều nút, loại bỏ sự phụ thuộc vào các dịch vụ lưu trữ tập trung như Google Drive, Amazon S3. Điều này giúp bảo mật và đảm bảo rằng dữ liệu không dễ bị kiểm duyệt hay mất mát. Ví dụ:

- Storj: Nền tảng lưu trữ dữ liệu phi tập trung sử dụng blockchain và mạng P2P để lưu trữ dữ liệu một cách an toàn.

- Filecoin: Mạng lưu trữ phi tập trung cho phép người dùng thuê không gian lưu trữ từ những người khác trên toàn cầu, đồng thời đảm bảo dữ liệu an toàn và không bị kiểm duyệt.

***8. Quản trị phi tập trung***

Hệ phi tập trung có thể áp dụng vào các mô hình quản trị, nơi các quyết định được đưa ra dựa trên sự đồng thuận của cộng đồng hoặc tổ chức, mà không phụ thuộc vào một cơ quan hoặc lãnh đạo trung tâm. Ví dụ:

- Liquid Democracy: Mô hình quản trị phi tập trung cho phép công dân hoặc thành viên của một tổ chức bỏ phiếu trực tiếp hoặc ủy quyền cho người khác bỏ phiếu thay mình.

- Tezos: Một blockchain tự quản lý nơi người dùng có thể bỏ phiếu cho các thay đổi hoặc cải tiến của giao thức mà không cần sự can thiệp từ bên ngoài.

***9. Hệ thống y tế phi tập trung***

Các hệ thống y tế phi tập trung có thể cải thiện việc lưu trữ và chia sẻ dữ liệu bệnh nhân giữa các tổ chức y tế mà không cần phụ thuộc vào cơ quan trung ương. Điều này giúp bảo mật thông tin y tế và cho phép truy cập nhanh chóng đến hồ sơ bệnh nhân. Ví dụ:

- MedRec: Một dự án sử dụng blockchain để lưu trữ và quản lý hồ sơ y tế, cho phép bệnh nhân kiểm soát và chia sẻ dữ liệu y tế của họ.

***10. Internet of Things (IoT) phi tập trung***

Hệ thống IoT phi tập trung sử dụng blockchain hoặc mạng P2P để kết nối và quản lý các thiết bị mà không cần qua các máy chủ trung tâm. Điều này giúp tăng cường bảo mật và giảm thiểu các điểm thất bại duy nhất trong mạng lưới. Ví dụ:

- IOTA: Một nền tảng blockchain phi tập trung được thiết kế đặc biệt cho mạng IoT, cho phép các thiết bị giao tiếp và trao đổi dữ liệu một cách an toàn mà không cần trung gian.

Ứng dụng của hệ phi tập trung đang mở rộng mạnh mẽ trong nhiều lĩnh vực từ tài chính, y tế, lưu trữ dữ liệu cho đến quản trị. Với khả năng bảo mật cao, minh bạch và loại bỏ sự phụ thuộc vào các trung gian, hệ phi tập trung không chỉ giúp giảm chi phí mà còn tăng cường quyền kiểm soát của người dùng và tính bền vững của hệ thống.

## 1.5. HỆ PHÂN TÁN

### 1.5.1. Khái niệm hệ phân tán

Hệ phân tán (distributed system) là một tập hợp các máy tính độc lập kết hợp với nhau để tạo thành một hệ thống thống nhất, hoạt động như một thực thể duy nhất. Trong hệ thống này, các thành phần có thể nằm ở các vị trí khác nhau, và chúng giao tiếp với nhau qua mạng để thực hiện các tác vụ chung. Mục tiêu của hệ phân tán là cung cấp hiệu suất cao, tính sẵn sàng, và khả năng chịu lỗi trong khi hoạt động trên nhiều máy tính riêng biệt.

### 1.5.2. Đặc điểm của hệ phân tán

Tính độc lập: Các thành phần của hệ thống phân tán có thể hoạt động độc lập và không cần phải ở cùng một địa điểm.

Tính đồng thời: Các quá trình hoặc thành phần trong hệ phân tán có thể hoạt động đồng thời, thực hiện các tác vụ một cách song song.

Khả năng mở rộng: Hệ phân tán dễ dàng mở rộng bằng cách thêm các tài nguyên (máy tính hoặc phần cứng) mà không làm gián đoạn hệ thống.

Tính chịu lỗi: Hệ phân tán có khả năng chịu lỗi tốt, tức là nếu một hoặc một vài thành phần bị hỏng, hệ thống vẫn có thể hoạt động bình thường hoặc với hiệu suất giảm nhẹ.

### 1.5.3. Các mô hình hệ phân tán

Trong các hệ thống phân tán, có nhiều mô hình khác nhau để tổ chức và quản lý các thành phần của hệ thống. Mỗi mô hình sẽ phù hợp với các mục tiêu, yêu cầu khác nhau của hệ thống như hiệu suất, tính sẵn sàng, bảo mật, khả năng mở rộng, và độ phức tạp. Dưới đây là một số mô hình phổ biến trong hệ phân tán:

#### *1. Mô hình Máy khách - Máy chủ (Client-Server)*

Mô hình máy khách - máy chủ là một trong những mô hình đơn giản và phổ biến nhất trong hệ phân tán. Trong mô hình này, có hai loại nút chính:

- Máy khách (Client): Máy khách là các nút gửi yêu cầu đến máy chủ và nhận phản hồi từ máy chủ. Máy khách thường là các thiết bị đầu cuối như trình duyệt web hoặc ứng dụng người dùng.

- Máy chủ (Server): Máy chủ là các nút cung cấp dịch vụ hoặc tài nguyên cho các máy khách. Máy chủ nhận yêu cầu từ máy khách, xử lý chúng và trả về kết quả.

Ưu điểm của mô hình là dễ triển khai, dễ bảo trì, dễ kiểm soát vì có máy chủ trung tâm. Nhược điểm của nó là máy chủ có thể bị quá tải nếu có quá nhiều máy khách đồng thời yêu cầu dịch vụ, và mô hình này có thể không chịu lỗi tốt nếu máy chủ trung tâm gặp sự cố.

Ví dụ: Mô hình ứng dụng web truyền thống, nơi máy khách (trình duyệt) gửi yêu cầu HTTP đến máy chủ web và nhận lại trang web.

***2. Mô hình Ngang hàng (Peer-to-Peer - P2P)***

Trong mô hình Peer-to-Peer, tất cả các nút trong hệ thống đều có quyền và trách nhiệm như nhau. Không có một máy chủ trung tâm, và mỗi nút có thể vừa là máy khách vừa là máy chủ, chia sẻ tài nguyên và dịch vụ trực tiếp với các nút khác.

Ưu điểm của P2P là không có điểm thất bại duy nhất, dễ dàng mở rộng, và tài nguyên được phân phối đồng đều giữa các nút. P2P có nhược điểm là quản lý hệ thống và bảo mật phức tạp hơn do không có máy chủ trung tâm, và các nút có thể không luôn sẵn sàng hoặc đáng tin cậy.

Ví dụ: Mạng chia sẻ tệp như BitTorrent, nơi người dùng tải lên và tải xuống tệp từ các máy tính khác mà không cần một máy chủ trung tâm.

***3. Mô hình Đa tầng (Multi-tier)***

Mô hình đa tầng (hay còn gọi là mô hình ba tầng trong ứng dụng web) chia hệ thống phân tán thành nhiều tầng để phân chia các nhiệm vụ cụ thể, giúp hệ thống có tính linh hoạt và dễ bảo trì hơn.

Các tầng phổ biến trong mô hình đa tầng bao gồm:

- Tầng giao diện người dùng (Presentation Layer): Đây là tầng mà người dùng tương tác, chẳng hạn như trình duyệt web hoặc ứng dụng di động.

- Tầng logic xử lý (Business Logic Layer): Tầng này chứa các logic xử lý nghiệp vụ và quy trình xử lý yêu cầu từ người dùng.

- Tầng cơ sở dữ liệu (Data Layer): Tầng này quản lý cơ sở dữ liệu, nơi lưu trữ và truy xuất dữ liệu.

Mô hình này có ưu điểm là dễ dàng bảo trì, mở rộng và phân chia trách nhiệm giữa các tầng. Tuy vậy các tầng có thể tạo ra độ trễ do yêu cầu phải đi qua nhiều bước, đặc biệt là trong môi trường mạng.

Ví dụ: Ứng dụng web ba tầng (web application) với tầng giao diện người dùng (UI), tầng xử lý nghiệp vụ (Business Logic), và tầng cơ sở dữ liệu (Database).

***4. Mô hình Hệ thống Tệp phân tán (Distributed File System - DFS)***

Trong mô hình này, hệ thống tệp phân tán cung cấp một cách tiếp cận để lưu trữ và truy cập dữ liệu từ nhiều nút phân tán mà người dùng có thể truy cập như thể dữ liệu đó nằm trên một hệ thống tệp duy nhất.

Các tệp được chia thành các khối dữ liệu nhỏ, phân tán trên nhiều máy chủ, và người dùng có thể truy cập chúng thông qua một hệ thống giao diện chung.

Ưu điểm của mô hình là dễ dàng mở rộng, có khả năng chịu lỗi tốt, và cung cấp khả năng truy cập tệp từ bất kỳ nút nào trong hệ thống. Tuy vậy, trong mô hình này việc quản lý nhất quán tệp và đồng bộ hóa giữa các nút có thể gặp khó khăn.

Ví dụ: Hadoop Distributed File System (HDFS), Google File System (GFS).

***5. Mô hình Hệ thống Cơ sở dữ liệu phân tán (Distributed Database System)***

Hệ thống cơ sở dữ liệu phân tán chia cơ sở dữ liệu thành nhiều phần và phân phối chúng trên nhiều nút trong hệ thống. Các nút này có thể thực hiện các tác vụ lưu trữ và truy vấn dữ liệu đồng thời, nhằm cải thiện hiệu suất và khả năng chịu lỗi của hệ thống.

Ưu điểm của mô hình này là dễ dàng mở rộng, dữ liệu được phân phối đồng đều, và hệ thống có khả năng chịu lỗi cao. Tuy vậy sự đồng bộ hóa và duy trì tính nhất quán giữa các bản sao dữ liệu có thể phức tạp.

Ví dụ: Hệ thống cơ sở dữ liệu NoSQL như MongoDB, Cassandra, hoặc hệ thống cơ sở dữ liệu SQL phân tán như Google Spanner.

***6. Mô hình Đám mây (Cloud Computing)***

Mô hình đám mây sử dụng các tài nguyên phân tán từ nhiều máy chủ đặt tại các trung tâm dữ liệu khác nhau và cung cấp các dịch vụ qua Internet. Người dùng có thể truy cập các dịch vụ này mà không cần phải lo lắng về cơ sở hạ tầng phần cứng.

Mô hình đám mây có khả năng mở rộng linh hoạt, tiết kiệm chi phí phần cứng, và dễ dàng tiếp cận các dịch vụ và tài nguyên từ xa. Mô hình này phụ thuộc vào kết nối Internet, và các vấn đề bảo mật có thể phát sinh khi lưu trữ dữ liệu trên đám mây.

Ví dụ: Amazon Web Services (AWS), Microsoft Azure, Google Cloud.

*7. Mô hình Tính toán đám mây phân tán (Edge Computing)*

Trong mô hình tính toán đám mây phân tán, các tài nguyên và dịch vụ được phân tán ra gần với người dùng hoặc thiết bị (edge devices), thay vì tập trung vào các trung tâm dữ liệu lớn.

Ưu điểm của mô hình là giảm độ trễ, tiết kiệm băng thông, và xử lý dữ liệu nhanh chóng gần nguồn gốc. Mô hình có nhược điểm là quản lý và bảo mật dữ liệu trở nên phức tạp hơn.

Ví dụ: Các ứng dụng IoT, nơi dữ liệu được xử lý ngay tại các thiết bị cảm biến hoặc các nút mạng gần người dùng.

### 1.5.4. Tính chất của hệ phân tán

- Tính minh bạch (Transparency):

+ Minh bạch về truy cập: Người dùng không cần biết vị trí thực của tài nguyên (dữ liệu, dịch vụ) trong hệ phân tán, chỉ cần sử dụng như thể nó cục bộ.

+ Minh bạch về vị trí: Người dùng không cần biết tài nguyên đang nằm ở máy nào trong mạng.

+ Minh bạch về lỗi: Hệ thống có khả năng che giấu lỗi và phục hồi một cách tự động, giúp hệ thống vẫn duy trì hoạt động.

+ Minh bạch về tính di động: Tài nguyên hoặc tiến trình có thể di chuyển trong hệ thống mà không ảnh hưởng đến người dùng.

- Tính mở (Openness):

Hệ phân tán có thể dễ dàng mở rộng và kết hợp thêm các thành phần mới mà không ảnh hưởng đến hệ thống hiện tại.

Các thành phần trong hệ thống giao tiếp với nhau qua các giao thức chuẩn và dễ tương tác với các hệ thống khác.

- Tính đồng bộ và không đồng bộ (Synchronous and Asynchronous):

Trong hệ phân tán, có thể có các hệ thống đồng bộ, nơi các thành phần giao tiếp với nhau với một tốc độ xác định. Tuy nhiên, phần lớn hệ thống phân tán là không đồng bộ, trong đó các thành phần không cần phải chờ đợi nhau.

- Tính không tập trung (Decentralization):

Không có máy chủ trung tâm quản lý tất cả, các máy tính trong hệ phân tán thường có quyền quản lý và điều hành ngang nhau, giúp giảm thiểu rủi ro tập trung và tăng cường khả năng chịu lỗi.

- Tính chịu lỗi (Fault Tolerance):

Hệ phân tán có khả năng tiếp tục hoạt động dù một số thành phần bị lỗi. Các cơ chế như sao lưu, dự phòng và phát hiện lỗi giúp hệ thống duy trì được tính khả dụng cao.

- Tính nhất quán (Consistency):

Dữ liệu và trạng thái của hệ thống phải được duy trì nhất quán giữa các thành phần trong hệ phân tán. Có thể có nhiều mức độ nhất quán như nhất quán mạnh, nhất quán yếu hoặc nhất quán cuối cùng (eventual consistency).

- Tính khả dụng (Availability):

Hệ thống phải luôn sẵn sàng để phục vụ các yêu cầu của người dùng, ngay cả khi một số nút trong hệ thống không hoạt động.

- Tính mở rộng (Scalability):

Hệ phân tán có khả năng mở rộng dễ dàng khi nhu cầu tài nguyên tăng lên. Điều này bao gồm cả mở rộng về kích thước của hệ thống lẫn số lượng người dùng hoặc khối lượng công việc.

- Tính đối xứng (Symmetry):

Trong một số hệ phân tán, các thành phần có thể có vai trò tương đương nhau, không phân biệt rõ ràng máy chủ (server) và máy khách (client).

Những tính chất này giúp hệ phân tán trở thành một giải pháp hiệu quả trong nhiều tình huống, từ hệ thống lưu trữ dữ liệu, dịch vụ đám mây đến mạng lưới Internet of Things (IoT).

### 1.5.5. Quản lý đồng bộ và tính toàn vẹn

Quản lý đồng bộ (synchronization) và tính toàn vẹn (integrity) là những thách thức quan trọng trong hệ phân tán. Các khía cạnh này liên quan đến việc đảm bảo rằng các tiến trình hoặc nút trong hệ thống có thể hoạt động cùng nhau mà không xảy ra mâu thuẫn dữ liệu hay sai lệch trạng thái.

#### *1. Quản lý đồng bộ trong hệ phân tán:*

Đồng bộ trong hệ phân tán liên quan đến việc đảm bảo các tiến trình hoạt động một cách phối hợp và tránh xung đột truy cập tài nguyên hoặc dữ liệu dùng chung.

Cơ chế khóa (Locking Mechanisms): Một phương pháp phổ biến để quản lý đồng bộ là sử dụng các cơ chế khóa như mutexes, semaphores, hoặc giao thức phân phối token. Điều này ngăn chặn nhiều tiến trình cùng truy cập một tài nguyên tại cùng thời điểm, dẫn đến tranh chấp tài nguyên.

Đồng bộ theo thời gian (Clock Synchronization): Trong hệ phân tán, các tiến trình trên các máy khác nhau có thể có thời gian hệ thống không đồng bộ. Một số thuật toán được sử dụng để đồng bộ hóa thời gian giữa các nút như thuật toán NTP (Network Time Protocol), thuật toán Berkeley và Lamport Timestamps (để đồng bộ thứ tự các sự kiện).

***2. Tính toàn vẹn dữ liệu trong hệ phân tán:***

Tính toàn vẹn dữ liệu đảm bảo rằng dữ liệu không bị thay đổi hoặc hỏng hóc khi di chuyển qua các nút khác nhau của hệ phân tán.

Phân cấp đồng bộ hóa (Hierarchical Synchronization): Một số hệ phân tán sử dụng các phương pháp phân cấp để đảm bảo tính toàn vẹn dữ liệu trong các trường hợp không đồng bộ hoàn toàn giữa các nút.

Kiểm soát truy cập song song (Concurrency Control): Kiểm soát song song là một kỹ thuật đảm bảo rằng các giao dịch hoặc truy vấn từ nhiều người dùng được xử lý theo cách duy trì tính nhất quán của dữ liệu. Ví dụ: cơ chế Two-Phase Locking (2PL) thường được sử dụng trong hệ thống cơ sở dữ liệu phân tán.

Các mô hình nhất quán (Consistency Models): Trong hệ phân tán, có nhiều cấp độ nhất quán khác nhau, từ nhất quán mạnh (strong consistency) cho đến nhất quán cuối cùng (eventual consistency). Việc lựa chọn mô hình nhất quán phụ thuộc vào yêu cầu của hệ thống, giữa tính khả dụng và tính nhất quán.

***3. Thách thức trong quản lý đồng bộ và tính toàn vẹn:***

Độ trễ mạng (Network Latency): Do các nút trong hệ phân tán có thể nằm ở những vị trí địa lý khác nhau, độ trễ mạng và băng thông có thể ảnh hưởng đến việc đồng bộ hóa dữ liệu, dẫn đến sự sai lệch thời gian.

Phân vùng mạng (Network Partitioning): Các vấn đề về phân vùng mạng có thể gây ra các trạng thái không đồng bộ, ảnh hưởng đến tính toàn vẹn dữ liệu. Các thuật toán như Paxos và Raft được sử dụng để duy trì sự đồng thuận trong điều kiện mạng không đáng tin cậy.

***4. Các giải pháp quản lý đồng bộ và tính toàn vẹn:***

Thuật toán phân tán (Distributed Algorithms): Nhiều thuật toán đã được phát triển để quản lý đồng bộ và tính toàn vẹn như Chandy-Lamport Snapshot Algorithm, Bully Algorithm, và Paxos.

Replication: Bản sao (replica) dữ liệu trên nhiều nút giúp đảm bảo tính sẵn sàng và toàn vẹn. Tuy nhiên, việc đồng bộ giữa các bản sao cần các chiến lược mạnh mẽ để tránh các vấn đề về nhất quán.

Cơ chế đồng thuận (Consensus Mechanisms): Các hệ thống phân tán có thể sử dụng các cơ chế đồng thuận như Paxos hoặc Raft để đảm bảo rằng tất cả các nút đồng thuận về một giá trị cụ thể trước khi tiếp tục các tác vụ khác.

### 1.5.6. Ứng dụng của hệ phân tán

Hệ phân tán có nhiều ứng dụng trong các lĩnh vực khác nhau, từ công nghệ thông tin, tài chính, y tế đến giải trí và nghiên cứu khoa học. Dưới đây là một số ứng dụng phổ biến của hệ phân tán:

***1. Điện toán đám mây (Cloud Computing):***

Điện toán đám mây là một trong những ứng dụng phổ biến nhất của hệ phân tán, cho phép người dùng truy cập tài nguyên tính toán và lưu trữ dữ liệu qua mạng Internet.

Các nền tảng như Amazon Web Services (AWS), Google Cloud Platform (GCP) và Microsoft Azure đều dựa trên mô hình hệ phân tán để cung cấp dịch vụ cơ sở hạ tầng (IaaS), nền tảng (PaaS) và phần mềm (SaaS) cho người dùng toàn cầu.

Các dịch vụ như lưu trữ đám mây (cloud storage), tính toán đám mây (cloud computing), và mạng phân phối nội dung (Content Delivery Networks - CDN) đều sử dụng hệ phân tán để đảm bảo tính khả dụng và khả năng mở rộng.

***2. Cơ sở dữ liệu phân tán (Distributed Databases):***

Cơ sở dữ liệu phân tán cho phép lưu trữ và quản lý dữ liệu trên nhiều máy chủ trong một mạng. Điều này giúp tăng khả năng chịu lỗi, mở rộng dễ dàng và giảm độ trễ truy cập.

Các hệ thống như Google Spanner, Amazon DynamoDB, Apache Cassandra, và MongoDB sử dụng kiến trúc phân tán để quản lý dữ liệu hiệu quả và đảm bảo tính nhất quán.

Các cơ chế đồng bộ hóa và phân vùng dữ liệu giúp đảm bảo dữ liệu được sao lưu và đồng bộ giữa các máy chủ, phục vụ cho các ứng dụng yêu cầu hiệu suất cao.

***3. Blockchain và tiền điện tử (Cryptocurrency):***

Blockchain là một ứng dụng điển hình của hệ thống phân tán, nơi dữ liệu được lưu trữ trên một mạng các nút (nodes) và được duy trì thông qua các thuật toán đồng thuận như Proof of Work (PoW) hoặc Proof of Stake (PoS).

Các ứng dụng như Bitcoin, Ethereum và nhiều loại tiền điện tử khác hoạt động dựa trên hệ thống phân tán, không có một thực thể trung tâm quản lý, giúp tăng cường tính bảo mật, minh bạch và phi tập trung.

Blockchain cũng có các ứng dụng khác ngoài tiền điện tử, bao gồm hợp đồng thông minh (smart contracts), quản lý chuỗi cung ứng, và quản lý tài sản số.

***4. Hệ thống file phân tán (Distributed File Systems):***

Hệ thống file phân tán giúp lưu trữ và truy xuất dữ liệu trên nhiều máy chủ, cho phép các ứng dụng và người dùng có thể truy cập dữ liệu từ xa một cách dễ dàng.

Các hệ thống như Google File System (GFS), Hadoop Distributed File System (HDFS), và Ceph là những ví dụ điển hình về hệ thống file phân tán, thường được sử dụng trong các ứng dụng dữ liệu lớn (Big Data).

Những hệ thống này đảm bảo tính sẵn sàng và khả năng mở rộng, đồng thời giúp dữ liệu an toàn và đồng bộ.

***5. Ứng dụng mạng xã hội và truyền thông (Social Networking and Communication):***

Các mạng xã hội như Facebook, Twitter, Instagram, và các ứng dụng nhắn tin như WhatsApp, Telegram đều dựa vào hệ phân tán để xử lý hàng tỷ yêu cầu mỗi ngày.

Hệ thống phân tán giúp các mạng xã hội mở rộng quy mô, lưu trữ và quản lý lượng dữ liệu khổng lồ, đồng thời cung cấp nội dung theo thời gian thực cho người dùng trên toàn cầu.

Các dịch vụ truyền thông dựa trên hệ phân tán cũng sử dụng các kỹ thuật caching (bộ nhớ đệm) và CDN để cải thiện tốc độ truy cập và độ ổn định.

***6. Hệ thống quản lý tài chính và ngân hàng (Financial and Banking Systems):***

Các hệ thống thanh toán điện tử, ngân hàng trực tuyến và hệ thống giao dịch chứng khoán cũng sử dụng hệ phân tán để đảm bảo tính khả dụng, khả năng mở rộng và bảo mật.

Ví dụ, Visa, MasterCard, và các hệ thống giao dịch chứng khoán quốc tế đều dựa trên mô hình phân tán để xử lý hàng triệu giao dịch mỗi giây.

Các dịch vụ thanh toán như PayPal, Stripe, Square và các hệ thống quản lý tiền tệ khác cũng sử dụng hệ phân tán để đảm bảo giao dịch nhanh chóng và an toàn.

***7. Ứng dụng dữ liệu lớn (Big Data Analytics):***

Hệ phân tán là nền tảng của các ứng dụng dữ liệu lớn, nơi các công cụ như Apache Hadoop, Apache Spark, và Google MapReduce được sử dụng để xử lý và phân tích khối lượng lớn dữ liệu trên các hệ thống máy tính phân tán.

Các tổ chức như Google, Amazon, Facebook sử dụng các công cụ này để phân tích hành vi người dùng, tối ưu hóa hệ thống và cải thiện hiệu suất dịch vụ.

Hệ thống dữ liệu lớn yêu cầu khả năng mở rộng cao và chịu lỗi tốt, điều mà các hệ thống phân tán có thể cung cấp.

***8. IoT (Internet of Things):***

Mạng IoT là một hệ thống phân tán với hàng triệu thiết bị kết nối và chia sẻ dữ liệu với nhau. Các ứng dụng của IoT trong nhà thông minh, y tế, giao thông và công nghiệp đều dựa vào hệ thống phân tán để quản lý và xử lý dữ liệu từ các cảm biến và thiết bị.

Các nền tảng như AWS IoT, Google Cloud IoT, và Microsoft Azure IoT Hub cung cấp các dịch vụ dựa trên hệ phân tán để quản lý các thiết bị IoT và dữ liệu được tạo ra.

***9. Trò chơi trực tuyến (Online Gaming):***

Các trò chơi trực tuyến nhiều người chơi (MMO) như World of Warcraft, Fortnite, và League of Legends sử dụng hệ phân tán để quản lý hàng triệu người chơi trên toàn thế giới.

Hệ thống phân tán giúp giảm độ trễ và đảm bảo các trải nghiệm game thời gian thực ổn định cho người chơi dù họ ở các vị trí địa lý khác nhau.

Việc sử dụng các máy chủ phân tán cũng giúp trò chơi có khả năng mở rộng khi số lượng người chơi tăng lên.

***10. Ứng dụng trí tuệ nhân tạo (Artificial Intelligence):***

Các hệ thống AI và học máy (machine learning) thường yêu cầu khối lượng lớn tài nguyên tính toán và dữ liệu. Các mô hình như TensorFlow và PyTorch có thể được triển khai trên các hệ phân tán để đào tạo và suy luận trên dữ liệu lớn.

Các ứng dụng AI như nhận diện hình ảnh, xử lý ngôn ngữ tự nhiên và phân tích dữ liệu phức tạp đều có thể được triển khai trên hệ thống phân tán để tối ưu hóa hiệu suất.

Nhờ vào các đặc tính như khả năng chịu lỗi, mở rộng và tính khả dụng cao, hệ phân tán trở thành nền tảng quan trọng cho rất nhiều ứng dụng trong thế giới hiện đại.

## 1.6 MẠNG NGANG HÀNG

### 1.6.1. Khái niệm mạng ngang hàng

Mạng ngang hàng (Peer-to-Peer - P2P) là một mô hình mạng trong đó các nút (nodes) trong mạng đều có vai trò bình đẳng. Các nút này vừa có thể đóng vai trò là máy khách (client) vừa là máy chủ (server), tức là có thể vừa nhận vừa cung cấp tài nguyên. Không có máy chủ trung tâm quản lý, tất cả các thành viên trong mạng đều có thể trực tiếp trao đổi dữ liệu với nhau.

Mô hình P2P thường được sử dụng để chia sẻ tệp, tài nguyên tính toán hoặc kết nối giữa nhiều thiết bị trong hệ thống phân tán mà không cần trung gian.

### 1.6.2. Đặc điểm của mạng ngang hàng

Phân tán (Decentralized): Không có một máy chủ trung tâm quản lý toàn bộ hệ thống. Các thành viên đều bình đẳng và tự quản lý tài nguyên của mình.

Khả năng mở rộng (Scalability): Khi có nhiều nút tham gia mạng, khả năng mở rộng là rất cao. Điều này làm tăng băng thông và tài nguyên tính toán mà không cần thêm máy chủ trung tâm.

Chịu lỗi (Fault Tolerance): Mạng ngang hàng thường có khả năng chịu lỗi tốt, vì sự hỏng hóc của một vài nút không ảnh hưởng nhiều đến toàn bộ hệ thống.

Cân bằng tải (Load Balancing): Vì không có máy chủ trung tâm, việc lưu trữ và chia sẻ dữ liệu được phân phối đều trên các nút, giúp giảm tình trạng quá tải cho một máy chủ duy nhất.

Quản lý tài nguyên (Resource Management): Mỗi nút có thể quản lý tài nguyên của chính nó, bao gồm băng thông, bộ nhớ và dữ liệu.

### 1.6.3. Các loại mạng ngang hàng

- Mạng P2P không có cấu trúc (Unstructured P2P):

Trong mạng không có cấu trúc, các nút được sắp xếp ngẫu nhiên và không có quy tắc cố định để kết nối các nút với nhau. Mạng P2P không có cấu trúc dễ dàng thiết lập, không cần kiến trúc phức tạp. tuy vậy việc tìm kiếm tài nguyên có thể không hiệu quả, đặc biệt là khi số lượng nút tăng lên. Ví dụ: Gnutella, Napster.

- Mạng P2P có cấu trúc (Structured P2P):

Trong mạng có cấu trúc, các nút được sắp xếp theo một quy tắc nhất định, thường sử dụng bảng băm phân tán (Distributed Hash Table - DHT) để quản lý tài nguyên. Loại mạng này dễ dàng tìm kiếm và truy cập dữ liệu nhanh chóng nhưng phức tạp hơn trong việc thiết lập và bảo trì so với mạng không cấu trúc. Ví dụ: Chord, Kademlia.

- Mạng P2P lai (Hybrid P2P):

Kết hợp giữa mạng P2P và mạng máy khách-máy chủ, nơi một số nút đặc biệt đóng vai trò như máy chủ trung gian giúp điều phối và quản lý kết nối giữa các nút. Loại mạng này tận dụng được cả lợi ích của mô hình P2P và máy khách-máy chủ.

Ví dụ: BitTorrent (với các trackers đóng vai trò máy chủ trung gian).

### 1.6.4. Cơ chế hoạt động của mạng P2P

Kết nối giữa các nút: Các nút trong mạng P2P kết nối trực tiếp với nhau, tạo thành một mạng phân tán. Khi một nút tham gia mạng, nó tìm kiếm các nút khác để kết nối, và sau đó có thể chia sẻ hoặc yêu cầu tài nguyên.

Trao đổi dữ liệu: Dữ liệu hoặc tài nguyên có thể được chia sẻ giữa các nút bằng cách gửi yêu cầu đến các nút khác để tìm kiếm tài nguyên mong muốn. Các yêu cầu này có thể được phát sóng (broadcast) hoặc gửi theo cách có cấu trúc (structured search) tùy thuộc vào loại mạng P2P.

Tìm kiếm và phân phối dữ liệu: Một trong những cơ chế quan trọng trong P2P là tìm kiếm và định vị dữ liệu. Trong mạng P2P không cấu trúc, các yêu cầu tìm kiếm thường được phát đi khắp mạng. Trong mạng có cấu trúc, bảng băm phân tán DHT được sử dụng để tìm kiếm dữ liệu một cách nhanh chóng.

Chia nhỏ và ghép nối dữ liệu (Swarming): Trong các mạng như BitTorrent, các tệp tin lớn được chia thành nhiều phần nhỏ và các phần này được tải xuống từ nhiều nguồn khác nhau, giúp tăng tốc độ truyền tải.

### 1.6.5. Công nghệ và giao thức trong mạng ngang hàng

Bảng băm phân tán (Distributed Hash Table - DHT): DHT là công nghệ chính để định vị và quản lý dữ liệu trong các mạng P2P có cấu trúc. Các nút và tài nguyên trong mạng đều được ánh xạ tới một không gian địa chỉ ảo. Mỗi nút chỉ chịu trách nhiệm quản lý một phần của không gian này. Ví dụ: Kademlia, Chord.

Giao thức BitTorrent: BitTorrent là một giao thức phổ biến cho việc chia sẻ tệp tin trong mạng P2P. Các tệp tin được chia nhỏ thành nhiều mảnh, và người dùng có thể tải xuống các mảnh từ nhiều người khác nhau cùng lúc.

Gnutella: Là một trong những giao thức P2P không cấu trúc đầu tiên. Gnutella sử dụng mô hình phát sóng (broadcast) để tìm kiếm và trao đổi dữ liệu giữa các nút.

FastTrack: Giao thức được sử dụng bởi các mạng chia sẻ tệp tin nổi tiếng như Kazaa, sử dụng một số nút siêu cấp (supernodes) để quản lý và điều phối lưu lượng mạng.

### 1.6.6. Ứng dụng của mạng ngang hàng

- Chia sẻ tệp (File Sharing): Ứng dụng phổ biến nhất của mạng P2P là chia sẻ tệp tin, ví dụ như BitTorrent. Người dùng có thể chia sẻ tệp âm nhạc, video, phần mềm hoặc các tài liệu khác một cách dễ dàng và nhanh chóng. Ví dụ của ứng dụng hướng này có thể kể đến là BitTorrent, eMule.

- Truyền tải video và nội dung đa phương tiện (Media Streaming): Mạng P2P cũng được sử dụng cho các ứng dụng truyền tải video và phát trực tuyến nội dung đa phương tiện. Các nội dung được truyền tải qua mạng P2P có thể giảm tải cho máy chủ trung tâm và tăng tốc độ truy cập của người dùng. Ví dụ: Joost, Popcorn Time.

- Tính toán phân tán (Distributed Computing): Mạng P2P cho phép tận dụng sức mạnh tính toán của nhiều máy tính phân tán để thực hiện các tác vụ phức tạp. Ví dụ: SETI@home (tìm kiếm sự sống ngoài hành tinh), Folding@home (nghiên cứu về protein).

- Hệ thống blockchain và tiền điện tử (Blockchain & Cryptocurrency): Blockchain là một ứng dụng của mạng P2P, trong đó các giao dịch được xác nhận và ghi nhận trên một sổ cái phân tán mà không cần sự can thiệp của trung gian. Ví dụ: Bitcoin, Ethereum.

- Mạng truyền thông (Communication Networks): Mạng P2P được sử dụng trong các ứng dụng nhắn tin và truyền thông ngang hàng, nơi các tin nhắn và dữ liệu được truyền trực tiếp giữa các thiết bị mà không cần máy chủ trung gian. Các hệ thống Skype (trước đây sử dụng P2P), Tox là ví dụ của loại ứng dụng này.

## CÂU HỎI VÀ BÀI TẬP

1. Sự khác biệt giữa mã hóa đối xứng và mã hóa bất đối xứng là gì?
2. Liệt kê các thuật toán mã hóa đối xứng và giải thích cách chúng hoạt động?
3. Kể tên các thuật toán mã hóa bất đối xứng và giải thích lý do tại sao chúng thường được sử dụng trong bảo mật thông tin?
4. Chế độ hoạt động của mã hóa khối (block cipher mode) là gì?
5. Khái niệm padding trong mã hóa là gì?
6. Vai trò của khóa phiên (session key) trong mã hóa đối xứng là gì?
7. Hàm băm là gì?
8. Tính chất nào của hàm băm là quan trọng nhất?
9. Hàm băm mật mã khác gì so với hàm băm thông thường?

10. Sự khác biệt giữa mã hóa và hàm băm là gì?

11. Tại sao va chạm hàm băm (hash collision) là một vấn đề trong bảo mật?

12. Chữ ký số hoạt động như thế nào?

13. Sự khác biệt giữa chữ ký số và chữ ký điện tử là gì?

14. Vai trò của hàm băm trong chữ ký số là gì?

15. Các thuật toán phổ biến sử dụng trong chữ ký số là gì?

16. Làm thế nào để xác minh tính hợp lệ của chữ ký số?

17. Chữ ký số giúp ngăn chặn các cuộc tấn công như thế nào?

18. Viết một chương trình để mã hóa và giải mã một thông điệp sử dụng thuật toán AES. Thử nghiệm với các chế độ hoạt động khác nhau như ECB, CBC, và GCM.

19. Viết một chương trình để tạo ra một hệ thống mã hóa lai, trong đó sử dụng mã hóa bất đối xứng để mã hóa khóa phiên (session key) và mã hóa đối xứng để mã hóa thông điệp.

20. Sử dụng thuật toán AES trong chế độ ECB để mã hóa một hình ảnh bitmap. Quan sát kết quả và giải thích lý do tại sao chế độ ECB không nên được sử dụng để mã hóa dữ liệu lớn có cấu trúc như hình ảnh.

21. Viết một chương trình đơn giản để tạo ra một hàm băm dựa trên chuỗi ký tự (ví dụ: sử dụng tổng của mã ASCII). Hãy thử nghiệm với một số chuỗi đầu vào khác nhau

22. Cho một chuỗi băm đã được cho trước (ví dụ: `d2d2d2...`), hãy tìm một chuỗi đầu vào có thể tạo ra băm này. Hãy thử với các hàm băm phổ biến như MD5, SHA-1, SHA-256 và ghi lại thời gian thực hiện.

23. Hãy thử tìm hai chuỗi khác nhau nhưng tạo ra cùng một giá trị băm sử dụng một hàm băm đơn giản tự viết. So sánh kết quả với các hàm băm mật mã phổ biến như MD5 hay SHA-1.

24. Viết một chương trình để kiểm tra tính toàn vẹn của tệp tin sử dụng hàm băm (ví dụ: SHA-256). Hãy thay đổi một số byte trong tệp tin và quan sát sự thay đổi của giá trị băm.

25. Giải thích tấn công Birthday và viết một chương trình mô phỏng nguyên lý tấn công này trên một hàm băm đơn giản, sau đó thực hiện với một hàm băm như MD5 hoặc SHA-1.

26. Viết một chương trình đơn giản sử dụng thuật toán RSA để tạo chữ ký số cho một chuỗi văn bản và sau đó xác minh chữ ký này. Thử nghiệm với các thư viện mật mã phổ biến như `PyCryptodome` hoặc `OpenSSL`.

27. Sử dụng thuật toán ECDSA để tạo và xác minh chữ ký số. So sánh hiệu suất của ECDSA với RSA khi ký cùng một thông điệp.

28. Tải về một chứng chỉ số từ một trang web (ví dụ: thông qua HTTPS) và viết một chương trình để trích xuất khóa công khai từ chứng chỉ này. Sử dụng khóa công khai để xác minh một chữ ký số.

29. Nghiên cứu cách chữ ký số được sử dụng trong các hợp đồng thông minh (smart contract) trên nền tảng blockchain. Viết một hợp đồng thông minh đơn giản có sử dụng chữ ký số để xác minh giao dịch.

30. Thực hiện bài tập thực hành để tạo và xác minh chữ ký số bằng cả RSA và ECDSA, sau đó so sánh về kích thước chữ ký, tốc độ ký, và tốc độ xác minh.

## TÀI LIỆU THAM KHẢO

Books:


1. Stallings, William. "Cryptography and Network Security: Principles and Practice" (7th Edition)
2. Schneier, Bruce. "Applied Cryptography: Protocols, Algorithms, and Source Code in C" (2nd Edition)
3. Stinson, Douglas R. "Cryptography: Theory and Practice"
4. Katz, Jonathan; Lindell, Yehuda. "Introduction to Modern Cryptography"
5. Menezes, Alfred; van Oorschot, Paul C.; Vanstone, Scott A. "Handbook of Applied Cryptography" (CRC Press, 1996)
6. Schneier, Bruce. "Applied Cryptography: Protocols, Algorithms, and Source Code in C" (John Wiley & Sons, 2015)
7. Coulouris, George; Dollimore, Jean; Kindberg, Tim; Blair, Gordon. "Distributed Systems: Concepts and Design" (5th Edition)
8. Tanenbaum, Andrew S.; Van Steen, Maarten. "Distributed Systems: Principles and Paradigms" (2nd Edition)
9. Tanenbaum, Andrew S.; Van Steen, Maarten. "Distributed Systems: Principles and Paradigms" (2nd Edition, Pearson Education, 2016)
10. Coulouris, George; Dollimore, Jean; Kindberg, Tim; Blair, Gordon. "Distributed Systems: Concepts and Design" (5th Edition, Addison-Wesley, 2011)
11. National Institute of Standards and Technology (NIST) Special Publications (SPs) & FIPS PUBs:
12. FIPS PUB 197: Advanced Encryption Standard (AES)

13. FIPS PUB 180-4: Secure Hash Standard (SHS)
14. FIPS PUB 186-4: Digital Signature Algorithm (DSA)
15. SP 800-38A: Recommendation for Block Cipher Modes of Operation
16. SP 800-56A: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography

Request for Comments (RFCs):

1. RFC 3447: Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1
2. RFC 1321: MD5 Message-Digest Algorithm

Online Resources:

1. "The Twofish Encryption Algorithm: A 128-Bit Block Cipher" by Bruce Schneier et al.
2. "Serpent: A Proposal for the Advanced Encryption Standard" by Ross Anderson et al.
3. "The RC4 Stream Cipher and Its Variants" by Ronald Rivest
4. J. Daemen and V. Rijmen, "The Design of Rijndael: AES - The Advanced Encryption Standard"
5. NIST Special Publication 800-38A, "Recommendation for Block Cipher Modes of Operation"
6. "Public Key Infrastructure (PKI) and Digital Signatures," Microsoft Documentation
7. "Digital Signatures," National Institute of Standards and Technology (NIST)
8. "The Salsa20 Family of Stream Ciphers," by Daniel J. Bernstein
9. The Elliptic Curve Cryptography (ECC) Primer, Certicom Research
10. ISO/IEC 10118-3: Information technology — Security techniques — Hash-functions
11. Public Key Infrastructure (PKI) Standards
12. Directive 1999/93/EC of the European Parliament and of the Council on a Community framework for electronic signatures
13. NIST Digital Signature Standard (DSS)
14. Decentralized Finance (DeFi) Pulse
15. The InterPlanetary File System (IPFS) Documentation
16. Chainlink Blog
17. Ethereum Official Website
18. Coursera: "Cloud Computing Specialization" - University of Illinois
19. MIT OpenCourseWare: "Distributed Systems"
20. "What is a Distributed System?" - Martin Kleppmann

21. Leslie Lamport (1978). Time, Clocks, and the Ordering of Events in a Distributed System
22. Eric A. Brewer (2000). Towards Robust Distributed Systems
23. Ian Clarke et al. (2001), Freenet: A distributed anonymous information storage and retrieval system
24. John Buford, Heather Yu, Eng Keong Lua (2009), P2P Networking and Applications
25. Stefan Saroiu, Steven D. Gribble, Henry M. Levy (2003), Measurement and Analysis of Spyware in a University Environment
26. Bram Cohen (2008), The BitTorrent Protocol Specification
27. Leonard Kleinrock (1976), Queueing Systems Volume 2: Computer Applications
28. David P. Anderson (2004), BOINC: A System for Public-Resource Computing and Storage
29. M. Castro, P. Druschel, A-M.