

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

---

**ĐỖ XUÂN CHỢ**

**BÀI GIẢNG**  
**MẬT MÃ HỌC CƠ SỞ**

*Hà Nội, tháng 12 năm 2016*

## MỞ ĐẦU

Với xu thế bùng nổ phát triển công nghệ và các ứng dụng của chúng ta trong cuộc sống hiện nay thì vấn đề an toàn và bảo mật hệ thống thông tin đang trở nên cấp thiết hơn bao giờ hết. Có rất nhiều phương pháp, kỹ thuật, cách thức để bảo vệ hệ thống công nghệ thông tin trước các cuộc tấn công. Mỗi phương pháp và kỹ thuật có ưu điểm và nhược điểm riêng. Một phương pháp cơ bản để đảm bảo an toàn cho thông tin hiện nay là ứng dụng các kỹ thuật mật mã để mã hóa thông tin. Hiện nay, các ứng dụng của mật mã học được triển khai và áp dụng ở hầu hết các lĩnh vực và công nghệ trong đời sống. Chính vì vậy, việc nghiên cứu và tìm hiểu về mật mã học cũng như các ứng dụng của mật mã trong thực tế là vấn đề rất cần quan tâm hiện nay.

Môn học “Mật mã học cơ sở” là môn cơ sở chuyên ngành thuộc chương trình đào tạo đại học ngành An toàn thông tin của Học Viện Công Nghệ Bưu Chính Viễn Thông. Môn học cung cấp cho sinh viên các kiến thức cơ bản về mật mã học bao gồm: vai trò và tầm quan trọng của mật mã, khái niệm toán học trong mật mã; các thuật toán mã hóa thông dụng; các hàm băm; vấn đề quản lý và phân phối khóa; một số ứng dụng của mật mã trong thực tế. Bài giảng “Mật mã học cơ sở” được biên soạn trên cơ sở đề cương chi tiết môn học đã được duyệt và tổng hợp tài liệu từ nhiều nguồn nhằm cung cấp tài liệu phục vụ cho sinh viên nghiên cứu và học tập. Bài giảng được cấu trúc thành các chương như sau:

*Chương 1: Tổng quan về mật mã học.* Chương này cung cấp các kiến thức cơ bản liên quan đến mật mã học bao gồm: khái niệm, các thuật ngữ, lịch sử phát triển và một số ứng dụng cơ bản. Bên cạnh đó, chương 1 còn trình bày về một số lý thuyết toán học được ứng dụng trong lĩnh vực mật mã như: lý thuyết số, lý thuyết thông tin, độ phức tạp của thuật toán...

*Chương 2: Mã hóa khóa đối xứng.* Chương 2 cung cấp các kiến thức về hệ mật khóa đối xứng cũng như nguyên tắc mã hóa, giải mã của một số thuật toán mã hóa khóa đối xứng. Ngoài ra, chương 2 trình bày một số ưu và nhược điểm của các kỹ thuật mã hóa khóa đối xứng.

*Chương 3: Mã hóa khóa bất đối xứng.* Chương 3 trình bày một số kiến thức liên quan đến kỹ thuật mã hóa khóa bất đối xứng bao gồm: khái niệm, nguyên tắc mã hóa, giải mã, ưu điểm và nhược điểm. Bên cạnh đó, chương 3 đề cập một số ứng dụng của mã hóa khóa bất đối xứng trong thực tế.

*Chương 4: Các hàm băm.* Chương này cung cấp các kiến thức liên quan đến hàm băm bao gồm: định nghĩa, khái niệm, tính chất, vai trò của hàm băm. Ngoài ra, chương 4 còn trình bày về một số hàm băm phổ biến hiện nay cũng như nguyên tắc làm việc của một số hàm băm đó.

*Chương 5: Quản lý khóa.* Chương này cung cấp các kiến thức liên quan đến vấn đề quản lý và phân phối khóa bao gồm: các khái niệm, phân loại, vai trò... Ngoài ra, chương 5 còn mô tả quy trình quản lý và phân phối khóa của một số giao thức phổ biến hiện nay.

## MỤC LỤC

### DANH MỤC CÁC BẢNG BIỂU VÀ HÌNH VẼ

### DANH MỤC CÁC TỪ VIẾT TẮT

CHƯƠNG 1: TỔNG QUAN VỀ MẬT MÃ HỌC.....	11
1.1. CÁC THUẬT NGỮ VÀ CÁC KHÁI NIỆM CƠ BẢN.....	11
1.2. LỊCH SỬ PHÁT TRIỂN.....	13
1.3. PHÂN LOẠI .....	15
1.4. VAI TRÒ .....	17
1.5. MỘT SỐ ỨNG DỤNG CỦA MẬT MÃ HỌC.....	18
1.5.1. Kerberos .....	18
1.5.2. Krypto Knight .....	19
1.5.3. PGP.....	19
1.5.4. Thẻ thông minh .....	20
1.6. MỘT SỐ KHÁI NIỆM TOÁN HỌC TRONG MẬT MÃ.....	21
1.6.1. Lý thuyết thông tin .....	21
1.6.2. Lý thuyết số.....	34
1.6.3. Độ phức tạp thuật toán .....	37
CHƯƠNG 2: MÃ HÓA KHÓA ĐỐI XỨNG.....	42
2.1. GIỚI THIỆU VỀ MÃ HÓA KHÓA ĐỐI XỨNG .....	42
2.2. CÁC KỸ THUẬT MÃ HÓA KHÓA ĐỐI XỨNG CỔ ĐIỂN.....	43
2.2.1. Phương pháp thay thế.....	43
2.2.2. Phương pháp mã hóa hoán vị .....	53
2.2.3. Phương pháp mã hóa XOR .....	53
2.2.4. Phương pháp sách hoặc khóa chạy.....	54
2.3. CÁC KỸ THUẬT MÃ HÓA KHÓA ĐỐI XỨNG HIỆN ĐẠI .....	54
2.3.1. Thuật toán mã hóa DES.....	54
2.3.2. Thuật toán mã hóa AES .....	69
2.3.3. Thuật toán mã hóa A5/1 .....	83
2.3.4. Thuật toán mã hóa RC4.....	87
2.4. ƯU ĐIỂM VÀ NHƯỢC ĐIỂM CỦA MÃ HÓA KHÓA ĐỐI XỨNG .....	91

CHƯƠNG 3: MÃ HÓA KHÓA BẤT ĐỐI XỨNG.....	94
3.1. GIỚI THIỆU VỀ MÃ HÓA BẤT ĐỐI XỨNG.....	94
3.1.1. Giới thiệu chung.....	94
3.1.2. Một số thuật toán mã hóa bất đối xứng.....	96
3.2. THUẬT TOÁN MÃ HÓA RSA.....	97
3.3. ƯU ĐIỂM, NHƯỢC ĐIỂM CỦA MÃ HÓA BẤT ĐỐI XỨNG .....	100
3.4. ỨNG DỤNG CỦA MÃ HÓA BẤT ĐỐI XỨNG.....	102
CHƯƠNG 4: CÁC HÀM BĂM.....	104
4.1. GIỚI THIỆU VỀ HÀM BĂM .....	104
4.1.1. Định nghĩa.....	104
4.1.2. Phân loại.....	105
4.1.3. Các tính chất cơ bản.....	104
4.1.4. Vai trò.....	107
4.2. CÁC HÀM BĂM KHÔNG KHÓA.....	108
4.2.1. MDC độ dài đơn.....	109
4.2.2. MDC độ dài kép: MDC-2 và MDC- 4 .....	110
4.3. CÁC HÀM BĂM CÓ KHÓA.....	112
4.4. MỘT SỐ HÀM BĂM THÔNG DỤNG .....	113
4.4.1. Hàm băm họ MD - Message Digest (MD5).....	114
4.4.2. Hàm băm họ SHA - Secure Hash Algorithm (SHA-1) .....	117
CHƯƠNG 5: QUẢN LÝ KHÓA.....	121
5.1. GIỚI THIỆU .....	121
5.2. PHÂN PHỐI VÀ THỎA THUẬN KHÓA BÍ MẬT.....	124
5.2.1. Các kỹ thuật phân phối và thỏa thuận khóa bí mật .....	124
5.2.2. Một số giao thức phân phối và thỏa thuận khóa bí mật .....	129
5.3. PHÂN PHỐI VÀ THỎA THUẬN KHÓA CÔNG KHAI .....	137
5.3.1. Các kỹ thuật phân phối và thỏa thuận khóa công khai.....	137
5.3.2. Chứng chỉ xác thực khóa công khai X.509 .....	140
5.4. PHÂN PHỐI VÀ THỎA THUẬN KHÓA KẾT HỢP .....	143
5.4.1. Giới thiệu chung.....	143

5.4.2. Giao thức Diffie-Hellman .....	144
TÀI LIỆU THAM KHẢO.....	151

## DANH MỤC CÁC HÌNH VẼ

Hình 1. 1. Mô hình mã hóa và giải mã đơn giản .....	12
Hình 1. 2. Sơ đồ mã hóa của mã dòng .....	16
Hình 1. 3. Sơ đồ giải mã của mã dòng .....	16
Hình 2. 1: Mô hình mã hóa đối xứng .....	42
Hình 2. 2. Mô hình Feistel .....	55
Hình 2. 3. Sơ đồ tổng quát quá trình mã hóa DES .....	57
Hình 2. 4. Mô hình vòng lặp <i>DES</i> .....	59
Hình 2. 5. Mô tả hàm <i>f</i> trong <i>DES</i> .....	60
Hình 2. 6. Mô hình các bước tạo khóa của <i>DES</i> .....	65
Hình 2. 7. Sơ đồ mã hóa và giải mã <i>Triple DES</i> .....	69
Hình 2. 8. Sơ đồ SPN .....	70
Hình 2. 9. Sơ đồ mã hóa giải mã <i>AES</i> .....	74
Hình 2. 10. Phép biến đổi SubByte .....	76
Hình 2. 11. Phép biến đổi ShiftRows .....	78
Hình 2. 12. Phép biến đổi MixColumn .....	79
Hình 2. 13. Ví dụ về phép MixColumn .....	80
Hình 2. 14. Biểu diễn sinh $N_k$ khóa đầu tiên đối với khóa mã 128 bit .....	81
Hình 2. 15. Sơ đồ mã hóa giải mã <i>A5/1</i> .....	87
Hình 2. 16. Khởi tạo <i>S</i> và <i>T</i> .....	88
Hình 2. 17. Quá trình hoán vị <i>S</i> .....	89
Hình 2. 18. Quá trình sinh số mã hóa thông điệp .....	90
Hình 3. 1. Mô hình mã hóa bất đối xứng .....	95
Hình 3. 2. Mô hình bảo mật .....	101
Hình 3. 3. Mô hình xác thực .....	101
Hình 3. 4. Mô hình chứng thực và không chối bỏ .....	102
Hình 4. 1. Ví dụ tính chất hàm băm .....	104
Hình 4. 2. Phân loại hàm băm .....	105
Hình 4. 3. Sơ đồ kiểm tra toàn vẹn chỉ dùng <i>MAC</i> .....	107
Hình 4. 4. Sơ đồ kiểm tra toàn vẹn dùng <i>MDC</i> và mã hóa .....	108
Hình 4. 5. Sơ đồ kiểm tra tính toàn vẹn dùng <i>MDC</i> và kênh an toàn .....	108
Hình 4. 6. Các sơ đồ hàm băm <i>MDC</i> đơn .....	109
Hình 4. 7. Sơ đồ thuật toán <i>MDC-2</i> .....	111
Hình 4. 8. Sơ đồ thuật toán <i>MDC-4</i> .....	112
Hình 4. 9. Sơ đồ hàm băm <i>MAC</i> .....	113
Hình 4. 10. Sơ đồ hàm băm <i>MD5</i> .....	115
Hình 4. 11. Sơ đồ cấu trúc hàm <i>F</i> .....	116

Hình 4. 12. Cấu trúc biến đổi 1 vòng MD5 .....	117
Hình 4. 13. Sơ đồ thuật toán băm SHA-1 .....	118
Hình 4. 14. Cấu trúc hàm F của thuật toán băm SHA-1 .....	119
Hình 5. 1. Sơ đồ trao đổi khóa trong một hệ thống 5 người .....	124
Hình 5. 2. Phân phối khóa điểm – điểm .....	124
Hình 5. 3. Mô hình tạo và sử dụng khóa của hệ mã hóa khóa đối xứng .....	125
Hình 5. 4. Mô hình trao đổi khóa tập trung .....	126
Hình 5. 5. Mô hình trao đổi khóa đơn giản sử dụng KDC .....	126
Hình 5. 6. Sơ đồ mô tả quá trình trao đổi khóa của giao thức Needham-Schroeder .....	130
Hình 5. 7. Hoạt động của <i>Kerberos</i> .....	133
Hình 5. 8. Mô hình tạo và sử dụng khóa của hệ mã hóa khóa bất đối xứng .....	138
Hình 5. 9. Mô hình trao đổi khóa công khai thông qua sử dụng một máy chủ không trực tuyến và chứng chỉ .....	139
Hình 5. 10. Cơ chế xác thực một chiều .....	142
Hình 5. 11. Cơ chế xác thực hai chiều .....	142
Hình 5. 12. Cơ chế xác thực ba chiều .....	143
Hình 5. 13. Quá trình thiết lập khóa phiên giữa hai thực thể .....	144
Hình 5. 14. Sơ đồ ví dụ về việc trao đổi khóa .....	145



## DANH MỤC CÁC BẢNG BIỂU

Bảng 1. 1. Độ phức tạp để phân tích số nguyên ra thừa số nguyên tố .....	40
Bảng 2. 1: Bảng thống kê kết quả tấn công vét cạn .....	44
Bảng 2. 2: Ma trận mã hóa Palayfair .....	46
Bảng 2. 3. Bảng thế A .....	50
Bảng 2. 4. Bảng chân trị phép toán XOR .....	53
Bảng 2. 5. Bảng hoán vị khởi tạo <i>DES</i> .....	58
Bảng 2. 6. Bảng hoán vị kết thúc <i>DES</i> .....	58
Bảng 2. 7. Bảng <i>Expand DES</i> .....	60
Bảng 2. 8. Bảng <i>S-box1</i> .....	61
Bảng 2. 9. Bảng <i>S-box2</i> .....	61
Bảng 2. 10. Bảng <i>S-box3</i> .....	62
Bảng 2. 11. Bảng <i>S-box4</i> .....	62
Bảng 2. 12. Bảng <i>S-box5</i> .....	62
Bảng 2. 13. Bảng <i>S-box6</i> .....	63
Bảng 2. 14. Bảng <i>S-box7</i> .....	63
Bảng 2. 15. Bảng <i>S-box8</i> .....	63
Bảng 2. 16. Bảng <i>P-box</i> .....	64
Bảng 2. 17. Bảng nén khóa 64 xuống 56bit <i>DES</i> .....	65
Bảng 2. 18. Bảng nén khóa 56 bit xuống 48 bit <i>DES</i> .....	65
Bảng 2. 19. Ví dụ mã hóa <i>DES</i> .....	66
Bảng 2. 20. Bảng <i>S-box AES</i> .....	75
Bảng 2. 21. Hộp thay thế ngược <i>IS</i> .....	77
Bảng 2. 22. Bảng kết quả của quá trình sinh khóa .....	83
Bảng 4. 1. Các hàm băm thông dụng .....	113
Bảng 4. 2. Bảng tính giá trị $ROTL(t, s)$ .....	117
Bảng 5. 2. Quá trình trao đổi khóa giữa hai thực thể trong giao thức Shamir .....	132
Bảng 5. 3. Bảng diễn giải giao thức Diffie-Hellman .....	146

## DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Thuật ngữ Tiếng Anh	Thuật ngữ Tiếng Việt
AES	Advanced Encryption Standard	Chuẩn mã hóa nâng cao
CBC	Cipher Block Chaining	Chế độ mã móc xích
CFB	Cipher Feedback Mode	Chế độ mã phản hồi
CRHF	Collision resistant hash functions	Hàm băm chống đụng độ
DES	Data Encryption Standard	Chuẩn mã hóa dữ liệu
ECB	Electronic code book	Chế độ bảng tra mã điện tử
H	Hash function	Hàm băm
KDC	Key distribution center	Trung tâm phân phối khóa
KTC	Key translation center	Trung tâm dịch khóa
MAC	Message authentication codes	Mã xác thực thông điệp
MD	Message Digest	Tóm tắt thông điệp (có thể xem như một loại hàm băm)
MDC	Modification detection codes	Mã phát hiện sửa đổi
OWHF	One-way hash functions	Hàm băm một chiều
PKC	Public-key certificate	Chứng chỉ khóa công khai
SHA	Secure Hash Algorithm	Thuật toán băm an toàn
SPN	Substitution-Permutation Network	Mạng thay thế hoán vị

## CHƯƠNG 1: TỔNG QUAN VỀ MẬT MÃ HỌC

*Chương này cung cấp các kiến thức cơ bản liên quan đến mật mã học bao gồm: khái niệm, các thuật ngữ, lịch sử phát triển và một số ứng dụng cơ bản. Bên cạnh đó, chương 1 còn trình bày về một số lý thuyết toán học được ứng dụng trong lĩnh vực mật mã như: lý thuyết số, lý thuyết thông tin, độ phức tạp của thuật toán...*

### 1.1. CÁC THUẬT NGỮ VÀ CÁC KHÁI NIỆM CƠ BẢN

Một trong những khái niệm căn bản trong lĩnh vực mật mã là khái niệm *mật mã học*. Có nhiều định nghĩa khác nhau về mật mã học như: *Mật mã học là kỹ thuật, nghệ thuật viết các ký tự bí mật* (theo Webster's Revised Unabridged Dictionary). Hoặc *mật mã học là việc mã hóa dữ liệu mà nó chỉ có thể được giải mã bởi một số người chỉ định* (Free Online Dictionary of Computing)... Tuy nhiên, theo quan điểm của tác giả thì có thể hiểu một cách đơn giản về mật mã học như sau: *Mật mã học là một ngành khoa học chuyên nghiên cứu, áp dụng các phương pháp, kỹ thuật, thuật toán nhằm che giấu, xử lý hoặc biến đổi thông tin thông thường thành dạng không đọc trực tiếp được là văn bản mã hóa*. Để thực hiện được quy trình biến đổi từ thông tin thông thường thành văn bản mã hóa bao gồm nhiều công đoạn với nhiều thuật ngữ và các khái niệm liên quan. Dưới đây sẽ trình bày một số thuật ngữ và khái niệm cơ bản được sử dụng phổ biến nhất hiện nay.

#### - **Người gửi, người nhận**

*Người gửi (sender)*: là người chịu trách nhiệm gửi thông điệp được mã hóa cho các bên liên quan.

*Người nhận (Receiver)*: là người có quyền hợp pháp nhận được thông điệp từ người gửi.

#### - **Thông điệp và Mã hóa**

*Thông điệp (plaintext)*: hay còn gọi là bản rõ là một dạng văn bản có nội dung mà Người gửi muốn gửi tới Người nhận. Thông điệp thường được ký hiệu là  $M$  (message) hoặc  $P$  (plaintext).  $P$  hoặc  $M$  có thể là chuỗi bit, một file văn bản, tập tin ảnh bitmap, một dòng âm thanh hay ảnh số...

*Mã hóa (encryption)*: quá trình cải trang một thông điệp, che giấu nội dung thật sự của thông điệp để nội dung của thông điệp không bị lộ.

*Bản mã (ciphertext)*: là kết quả của một thông điệp được mã hóa. Bản mã được ký hiệu là  $C$  (*ciphertext*).

*Giải mã (decryption)*: là quá trình chuyển đổi từ bản mã trở lại thông điệp ban đầu.

Giả sử  $E$  là hàm mã hóa (*encrypt*), áp dụng trên  $P$  để tạo ra  $C$ , có biểu diễn toán học dạng:

$$E(P) = C \quad (1.1)$$

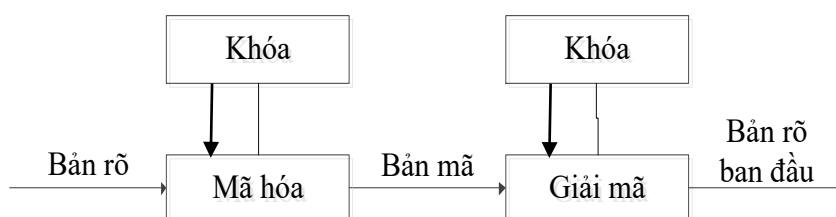
Trong quy trình ngược lại,  $D$  là hàm giải mã (*decrypt*) áp dụng trên  $C$  để có  $P$ :

$$D(C) = P \quad (1.2)$$

Để đảm bảo quá trình từ mã hóa đến giải mã là đúng, cần đảm bảo:

$$D(E(P)) = P \quad (1.3)$$

#### - Thuật toán và khóa



**Hình 1. 1. Mô hình mã hóa và giải mã đơn giản**

Hình 1.1. thể hiện một mô hình mã hóa và giải mã đơn giản. Từ mô hình này có thể thấy một số khái niệm cụ thể như sau :

*Thuật toán mã hóa* (hay còn gọi là *giải thuật mã hóa*): là các phương pháp, mô hình toán học được ứng dụng trong quá trình mã hóa thông tin.

*Thuật toán giải mã* (hay còn gọi là *giải thuật giải mã*): là kỹ thuật, thuật toán, mô hình toán học được ứng dụng trong quá trình giải mã thông tin.

*Khóa (key)* : bản chất của Key là chìa. Tuy nhiên, thuật ngữ chìa không được sử dụng phổ biến mà thường sử dụng thuật ngữ Key là khóa. Chính vì vậy, trong bài giảng sẽ thống nhất sử dụng Key – khóa. Khóa là công cụ được ứng dụng trong quá trình mã hóa và giải mã. Khóa có thể là một phần tử trong một tập giá trị nào đó, phạm vi các giá trị

của khóa được gọi là không gian khóa. Khóa dùng để mã hóa và giải mã có thể giống hoặc khác nhau, tùy theo từng phương pháp, kỹ thuật mã hóa sử dụng. Các chương sau sẽ phân tích chi tiết hơn về vấn đề này.

- **Thăm mã:** là ngành khoa học khôi phục lại thông điệp mà không cần biết khóa. Thăm mã thành công có thể thu được thông điệp hoặc khóa hoặc cả hai. Nó cũng có thể tìm thấy điểm yếu của hệ mật mà đã để lộ ra các kết quả trước đó. Thăm mã có thể chia thành 4 loại chính:

- Tấn công khi chỉ biết bản mã (ciphertext-only attack) ;
- Tấn công đã biết thông điệp (known-plaintext attack):
- Tấn công chọn thông điệp rõ (chosen plaintext attack) ;
- Tấn công chọn thông điệp thích hợp (adaptive-chosen-ciphertext attack).

- **Độ an toàn thuật toán:** Trong các nghiên cứu của Lars Knudsen đã trình bày chi tiết và cụ thể về độ an toàn của thuật toán mã hóa cũng như các tiêu chí đánh giá mức độ an toàn của một thuật toán. Dưới đây là một số tiêu chí cơ bản để đánh giá độ an toàn của một thuật toán mã hóa:

- Chi phí yêu cầu cho việc phá mã lớn hơn so với việc mã hóa thì thuật toán đó tương đối an toàn;
- Thời gian yêu cầu để phá mã dài hơn so với thời gian mà dữ liệu mã hóa cần giữ bí mật thì thuật toán đó là an toàn;
- Tổng lượng dữ liệu mã hóa bằng 1 khóa ít hơn lượng dữ liệu đưa vào để phá mã thì nó được coi là an toàn.

## 1.2. LỊCH SỬ PHÁT TRIỂN

Mật mã học là một ngành khoa học có lịch sử từ hàng nghìn năm nay. Trong phần lớn thời gian phát triển của mình, lịch sử mật mã học chính là lịch sử của những phương pháp mật mã học cổ điển. Vào đầu thế kỷ 20, sự xuất hiện của các cơ cấu cơ khí và điện cơ, chẳng hạn như máy Enigma, đã cung cấp những cơ chế phức tạp và hiệu quả hơn cho việc mật mã hóa. Bài giảng này sẽ trình bày một số cột mốc quan trọng trong lịch sử phát triển của mật mã học.

*Mật mã học cổ điển:* cách đây khoảng 4500 năm người Hy Lạp cổ đại đã sử dụng các kỹ thuật mật mã ví dụ như gậy mật mã. Cũng có những bằng chứng cho thấy người La Mã nắm được các kỹ thuật mật mã (mật mã *Caesar* và các biến thể).

*Thời trung cổ:* Nguyên do xuất phát có thể là từ việc phân tích bản kinh Koran, do nhu cầu tôn giáo mà kỹ thuật phân tích tần suất đã được phát minh để phá vỡ các hệ thống mật mã đơn ký tự vào khoảng năm 1000. Đây chính là kỹ thuật phá mã cơ bản nhất được sử dụng, mãi cho tới tận thời điểm thế chiến thứ II. Về nguyên tắc, mọi kỹ thuật mật mã đều không chống lại được kỹ thuật mã này cho tới khi kỹ thuật mật mã dùng nhiều bảng chữ cái được Alberti sáng tạo (năm 1465).

*Mật mã học từ 1800 tới Thế chiến II:* Tới thế kỷ 19, mật mã học mới được phát triển một cách có hệ thống. Trong thập niên 1840, Edgar Allan Poe đã xây dựng một số phương pháp có hệ thống để giải mật mã. Sau này ông có viết một đề tài về các phương pháp mã hóa và chúng trở thành những công cụ rất có lợi, được áp dụng vào việc giải mã của Đức trong Thế chiến II.

*Mật mã học trong Thế chiến II:* Trong thế chiến II, các hệ thống mật mã cơ khí và cơ điện tử được sử dụng rộng rãi. Người Đức đã sử dụng rộng rãi một hệ thống máy rôto cơ điện tử, dưới nhiều hình thức khác nhau, có tên gọi là máy Enigma. Quân đội Đức cũng cho triển khai một số thử nghiệm cơ học sử dụng thuật toán mật mã dùng một lần (one-time pad). Bộ ngoại giao của Nhật cũng cục bộ xây dựng một hệ thống dựa trên nguyên lý của "bộ điện cơ chuyển mạch dịch bước" (được Mỹ gọi là Purple), và đồng thời cũng sử dụng một số máy tương tự để trang bị cho một số tòa đại sứ Nhật Bản. Các máy mật mã mà phe Đồng Minh sử dụng trong thế chiến II, bao gồm cả máy TypeX của Anh và máy SIGABA của Mỹ, đều là những thiết kế cơ điện dùng rôto trên tinh thần tương tự như máy Enigma, song có nhiều nâng cấp lớn.

*Mật mã học hiện đại:* Năm 1949 Claude Shannon công bố bài: Lý thuyết về truyền thông trong các hệ thống bảo mật (Communication Theory of Secrecy Systems) trên tập san Bell System Technical Journal - Tập san kỹ thuật của hệ thống Bell - và một thời gian ngắn sau đó, trong cuốn Mathematical Theory of Communication - Lý thuyết toán học trong truyền thông - cùng với tác giả Warren Weaver. Ngày 17 tháng 3 năm 1975, IBM

(International Business Machines) đề xuất Tiêu chuẩn mật mã hóa dữ liệu DES (Data Encryption Standard). Vào năm 1976, Whitfield Diffie và Martin Hellman giới thiệu một phương pháp hoàn toàn mới về cách thức phân phối các khóa mật mã. Năm 2001, DES đã chính thức được thay thế bởi AES (Advanced Encryption Standard - Tiêu chuẩn mã hóa tiên tiến) khi NIST công bố phiên bản FIPS 197.

### 1.3. PHÂN LOẠI

Có nhiều phương pháp và tiêu chí để phân loại các phương pháp mã hóa. Dưới đây sẽ trình bày một số phương pháp phổ biến dùng để phân loại mã hóa.

*Cách 1: Phân loại mã hóa theo đặc trưng của khóa:*

Mã hóa khóa đối xứng (Hay còn gọi là mã hóa khóa bí mật): là phương pháp mã hóa mà quá trình mã hóa và giải mã dùng chung một khóa. Khóa được dùng trong mã hóa đối xứng gọi là khóa bí mật. Các thuật toán mã hóa đối xứng thường gặp: DES, AES...

Mã hóa khóa bất đối xứng (Hay còn gọi là mã hóa khóa công khai): là phương pháp mã hóa mà khóa sử dụng để mã hóa sẽ khác với khóa dùng để giải mã. Khóa dùng để mã hóa gọi là khóa công khai và khóa dùng để giải mã gọi là khóa bí mật. Tất cả mọi người đều có thể biết được khóa công khai (kể cả hacker), và có thể dùng khóa công khai để mã hóa thông tin. Nhưng chỉ có người nhận mới nắm giữ khóa bí mật, nên chỉ có người nhận mới có thể giải mã được thông tin.

*Cách 2: Phân loại mã hóa theo đặc trưng xử lý thông điệp*

Mã hóa dòng: là kiểu mã hóa mà từng bit (hoặc từng ký tự) của thông điệp được kết hợp với từng bit (hoặc từng ký tự) tương ứng của khóa để tạo thành bản mã.

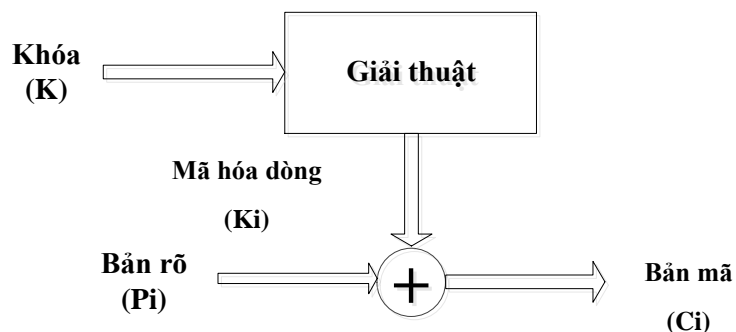
Đặc trưng cơ bản của mã dòng như sau:

- Kích thước một đơn vị mã hóa là :  $k$  bit.
- Thông điệp được chia thành các đơn vị mã hóa  $P \rightarrow P_0P_1P_2...P_{n-1}$   $\langle \text{length}(P_i) = k \text{ bit} \rangle$
- Một bộ sinh số ngẫu nhiên: Dùng khóa  $K$  ban đầu để sinh ra các số ngẫu nhiên có kích thước bằng kích thước của đơn vị mã hóa :

$$\text{StreamCipher}(K) \rightarrow S = S_0S_1S_2...S_{n-1} \langle \text{length}(S_i) = k \text{ bit} \rangle$$

- Mỗi đơn vị mã hóa được XOR tương ứng với số ngẫu nhiên để cho ra bản mã.

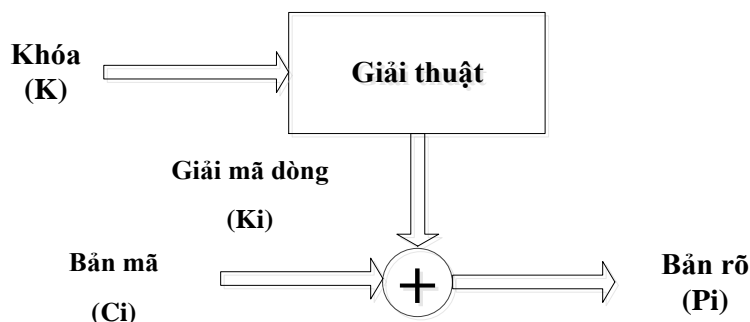
$$C_0 = P_0 \oplus S_0, C_1 = P_1 \oplus S_1, \dots, C_{n-1} = P_{n-1} \oplus S_{n-1} \rightarrow C = C_0 C_1 C_2 \dots C_{n-1};$$



Hình 1. 2. Sơ đồ mã hóa của mã dòng

- Quá trình giải mã là một quá trình ngược lại, bản mã được XOR với bộ sinh số ngẫu nhiên cho ra thông điệp.

$$P_0 = C_0 \oplus S_0, P_1 = C_1 \oplus S_1, \dots, P_{n-1} = C_{n-1} \oplus S_{n-1} \rightarrow P = P_0 P_1 P_2 \dots P_{n-1};$$



Hình 1. 3. Sơ đồ giải mã của mã dòng

Một số thuật toán mã hóa dòng phổ biến hiện nay: A5/1; RC4. Trong chương 2 của bài giảng sẽ trình bày chi tiết hơn về các thuật toán này.

Mã hóa khối: là kiểu mã hóa mà dữ liệu được chia ra thành từng khối có kích thước cố định để mã hóa.

Để xây dựng thuật toán mã hóa khối an toàn bằng cách sử dụng kết hợp các thao tác mã hóa tạo ra tính hỗn loạn và tính khuếch tán thông tin:

- Tính hỗn loạn (*diffusion*) giúp phá vỡ mối quan hệ giữa thông điệp và bản mã, tạo ra mối quan hệ phức tạp và chặt chẽ giữa khóa và bản mã.



- Sự khuếch tán (*confusion*) giúp phá vỡ và phân tán các phần tử trong các mã xuất hiện trong thông điệp để không thể phát hiện ra các mẫu này trong bản mã.

**Các chế độ hoạt động của mã hóa khối:** Trong mã hóa khối có 4 chế độ hoạt động cơ bản như sau:

- Chế độ ECB (Electronic Codebook): cùng khối thông điệp đầu vào, khối bản mã giống nhau. Các khối bản mã hoàn toàn độc lập nhau.
- Chế độ CBC (Cipher-Block Chaining): cùng khối thông điệp đầu vào, khối bản mã giống nhau với cùng khóa và phần nối đuôi. Khối mã  $c_j$  phụ thuộc vào khối rõ  $x_j$  và các khối rõ trước đó ( $x_1-x_{j-1}$ ).
- Chế độ CFB (Cipher Feedback): cùng khối thông điệp đầu vào, khối bản mã khác nhau. Khối mã  $c_j$  phụ thuộc vào khối rõ  $x_j$  và các khối rõ trước đó ( $x_1-x_{j-1}$ ).
- Chế độ OFB (Output Feedback): cùng khối thông điệp đầu vào, khối bản mã khác nhau. Luồng khóa độc lập với thông điệp.

**Các mô hình mã hóa khối:** Về mô hình mã khối, các nhà mật mã chia làm 2 mô hình chính: mạng thay thế hoán vị (*Substitution-Permutation Network - SPN*) và Mô hình Feistel. Trong chương 2 của bài giảng sẽ trình bày chi tiết hơn về 2 mô hình này.

## 1.4. VAI TRÒ

Mật mã học có vai trò rất lớn trong việc đảm bảo an toàn cho hệ thống thông tin cũng như các ứng dụng hiện nay. Việc ứng dụng mật mã học nhằm đảm bảo các tính chất:

- Tính bí mật (*confidentiality*): thông tin chỉ được phép truy cập bởi những đối tượng hợp lệ, những đối tượng được cấp phép.
- Tính toàn vẹn thông tin (*integrity*): đảm bảo thông tin không bị thay đổi trong quá trình truyền tin; hoặc nếu có thay đổi thì sẽ bị phát hiện.
- Tính sẵn sàng (*availability*): đảm bảo thông tin phải luôn luôn sẵn sàng khi cần thiết. Điều đó có nghĩa rằng mật mã học cung cấp một cơ chế bảo mật tốt, có thể tránh được những rủi ro cả về phần cứng, phần mềm và tránh được các cuộc tấn công của tin tặc.

- Tính xác thực (authentication): đảm bảo các bên liên quan nhận biết và tin tưởng nhau trong một hệ truyền tin, đồng thời đảm bảo thông tin trao đổi là thông tin thật.
- Tính chống chối bỏ (non-repudiation): đảm bảo rằng các bên liên quan không thể chối bỏ các hành động đã thực hiện trước đó.

## 1.5. MỘT SỐ ỨNG DỤNG CỦA MẬT MÃ HỌC

Một số ứng dụng của mật mã phải kể đến như: Dịch vụ xác thực; Điều khiển truy nhập; Các công cụ cho đảm bảo an toàn cho truyền thông không dây; Các nền tảng bảo mật như PKI, PGP; Các giao thức bảo mật như SSL/TLS, SSH, SET, IPSec; Các hệ thống như VPN. Trong phần tiếp theo, bài giảng sẽ mô tả tổng quát một số ứng dụng cụ thể của mật mã đang được triển khai trong thực tế hiện nay.

### 1.5.1. Kerberos

*Kerberos* là một giao thức xác thực cho các máy đáng tin cậy trong một mạng không tin cậy. Nó cũng đảm bảo tính toàn vẹn và tính bí mật cho thông tin truyền đi. *Kerberos* sử dụng mã hóa khóa đối xứng như *DES*, *triple DES* để đảm bảo an toàn thông tin. Tuy nhiên, *Kerberos* sẽ mất tác dụng nếu một người nào đó có được quyền truy cập vào máy chủ và sao chép tập tin chứa khóa.

Một số nhận xét, đánh giá về ưu điểm và nhược điểm của giao thức Kerberos:

*Ưu điểm:* *Kerberos* được đánh giá là giao thức xác thực an toàn nhờ các đặc điểm sau:

- Khi sử dụng *Kerberos*, mật khẩu không bao giờ truyền dưới dạng rõ mà luôn được mã hóa.
- *Kerberos* không yêu cầu người dùng lặp đi lặp lại thao tác nhập mật khẩu trước khi truy cập vào các dịch vụ điều đó hạn chế được nguy cơ tấn công ăn cắp dữ liệu.
- Giao thức được mã hóa theo các chuẩn mã hóa cao cấp như *Triple DES*, *RC4*, *AES* nên rất an toàn.
- Tất cả các trao đổi giữa các máy đều chứa timestamp nên vé bị đánh cắp không thể tái sử dụng, chống được tấn công dùng lại (*replay attack*).

*Hạn chế của Kerberos:* Bên cạnh các ưu điểm kể trên, hệ thống *Kerberos* cũng tồn tại một số hạn chế nhất định:

- Độ bảo mật của hệ thống phụ thuộc vào sự an toàn của hệ thống *KDC* (Key Distribution Center). Nếu *KDC* bị tấn công thì toàn bộ các thành phần trong hệ thống cũng bị tê liệt.
- Do tất cả các trao đổi đều gắn *timestamp* nên đòi hỏi các máy tính trong hệ thống phải đồng bộ về thời gian (không chênh lệch nhau quá 5 phút). Nếu không đảm bảo điều này, cơ chế xác thực dựa trên thời hạn sử dụng sẽ không hoạt động.
- Với cơ chế đăng nhập một lần trên một máy tính, nếu máy tính đó rơi vào tay những kẻ tấn công mạng thì toàn bộ dữ liệu người dùng sẽ bị đánh cắp và gây nguy cơ cho toàn bộ hệ thống.

### 1.5.2. Krypto Knight

*KryptoKnight* được phát triển bởi Phòng thí nghiệm nghiên cứu *IBM Zurich* và *Yorktown*. *KryptoKnight* cung cấp dịch vụ xác thực và phân phối khóa cho các ứng dụng và đối tượng giao tiếp trong môi trường mạng.

Mục tiêu của *KryptoKnight* là cung cấp dịch vụ bảo mật mạng với tính bền và độ linh hoạt cao. Tính bền vững trong các thông điệp của *KryptoKnight* cho phép nó trở thành giao thức bảo mật truyền thông ở mọi tầng, có thể đảm bảo tính bí mật thông tin mà không cần sử dụng thêm bất kì giao thức nào khác. Mặt khác, *KryptoKnight* tránh việc sử dụng mã hóa hàng loạt, khiến cho kiến trúc của nó trở nên linh hoạt. *KryptoKnight* cung cấp các dịch vụ nhằm:

- Hỗ trợ người sử dụng ủy thác danh tính của mình cho đối tượng khác.
- Hỗ trợ các thực thể tham gia giao tiếp xác thực lẫn nhau bằng cách cung cấp ủy thác từ người sử dụng hợp pháp.
- Cho phép các thực thể xác thực nguồn gốc và nội dung dữ liệu được trao đổi.

Các dịch vụ mà *Kryptoknight* cung cấp bao gồm: Single Sign-On (SSO); Xác thực hai bên; Phân phối khóa; Xác thực nguồn gốc và nội dung dữ liệu.

### 1.5.3. PGP

*PGP (Pretty Good Privacy)*: khi mới ra đời *PGP* là công cụ bảo mật file/email, sau đó được phát triển thành một chuẩn bảo mật và có nhiều bản cài đặt của *PGP*. Chức năng bảo mật *PGP* bao gồm:

- Đảm bảo tính bí mật thông qua mã hóa
- Xác thực thông qua chữ ký số

Mô hình hoạt động của *PGP* được chia làm 3 mô hình chính như sau:

- *Mô hình Direct Trust*: Là mô hình đơn giản nhất. Trong mô hình này, người dùng tin tưởng rằng khóa là hợp lệ vì họ biết nó đến từ đâu.
- *Mô hình Hierarchical Trust*: Trong một hệ thống phân cấp, có một số các 'root CA' gọi là chứng chỉ gốc. Các root CA này có thể tự xác thực bản thân, hoặc có thể xác thực tới các chứng chỉ khác.
- *Mô hình A Web of Trust*: Một mô hình *web of trust* bao gồm cả các mô hình an toàn khác. Do đó nó là một mô hình tin cậy phức tạp nhất. Chứng chỉ có thể được tin cậy trực tiếp, hoặc được tin cậy trong chuỗi (*meta-introducers*), hoặc bởi một số nhóm người giới thiệu (*users*).

Trong thực tế, mô hình *PGP* được ứng dụng rất rộng với nhiều phiên bản khác nhau. Tuy nhiên, có thể khái quát ứng dụng của *PGP* như sau: email, chữ ký số, mật mã hóa ổ đĩa cứng máy tính xách tay, bảo mật tệp và thư mục, bảo mật các phiên trao đổi *IM*, mật mã hóa luồng chuyển tệp, bảo vệ các tệp và thư mục lưu trữ trên máy chủ mạng.

#### 1.5.4. Thẻ thông minh

Thẻ thông minh (Smart Card), thẻ gắn chip, hay thẻ tích hợp vi mạch (*integrated circuit card -ICC*) là loại thẻ bỏ túi thường có kích thước của thẻ tín dụng, bên trong chứa một mạch tích hợp có khả năng lưu trữ và xử lý thông tin. Nó có thể đóng vai trò như thẻ căn cước, thực hiện việc xác thực thông tin, lưu trữ dữ liệu hay dùng trong các ứng dụng thẻ. Thẻ thông minh được chia làm 2 loại chính như sau:

- Thẻ thông minh có tiếp xúc: Loại thẻ thông minh có tiếp xúc có một diện tích tiếp xúc, bao gồm một số tiếp điểm mạ vàng, và có diện tích khoảng 1 cm vuông. Khi được

đưa vào máy đọc, con chip trên thẻ sẽ giao tiếp với các tiếp điểm điện tử và cho phép máy đọc thông tin từ chip và viết thông tin lên nó.

- Thẻ thông minh không tiếp xúc: Đây là loại thẻ mà chip trên nó liên lạc với máy đọc thẻ thông qua công nghệ cảm ứng *RFID* (với tốc độ dữ liệu từ 106 đến 848 kbit/s).

Một số chức năng và vai trò của thẻ thông minh trong vấn đề xác thực và bảo mật thông tin:

- Cho phép thực hiện các giao dịch kinh doanh một cách hiệu quả theo một cách chuẩn mực, linh hoạt và an ninh mà trong đó con người ít phải can thiệp vào.
- Giúp thực hiện việc kiểm tra và xác nhận chặt chẽ mà không phải dùng thêm các công cụ khác như mật khẩu...Chính vì thế, có thể thực hiện hệ thống dùng cho việc đăng nhập sử dụng máy tính, máy tính xách tay, dữ liệu bảo mật hoặc các môi trường kế hoạch sử dụng tài nguyên của công ty như *SAP*, v.v..với thẻ thông minh là phương tiện kiểm tra và xác nhận duy nhất.

Ngoài ra, thẻ thông minh cung cấp một số tính năng có thể được sử dụng để cung cấp hoặc tăng cường bảo vệ quyền riêng tư trong hệ thống. Trong bài giảng chỉ liệt kê một số tính năng quan trọng của thẻ thông minh. Chi tiết về những tính năng này và làm thế nào thực hiện được các chức năng này, có thể tham khảo trong các tài liệu tham khảo khác. Một số tính năng cụ thể như sau: Chứng thực; An toàn dữ liệu lưu trữ; mã hóa; Bảo mật thiết bị; Thông tin liên lạc an toàn; Sinh trắc học; Chứng chỉ....

## **1.6. MỘT SỐ KHÁI NIỆM TOÁN HỌC TRONG MẬT MÃ**

Có thể thấy rằng mật mã học là một lĩnh vực khoa học rộng lớn có liên quan rất nhiều ngành toán học như: Đại số tuyến tính, Lý thuyết thông tin, Lý thuyết độ phức tạp tính toán. Bởi vậy việc trình bày đầy đủ mọi khía cạnh của toán học trong mật mã học trong khuôn khổ bài giảng là một điều khó có thể làm được. Chính vì lý do đó, bài giảng chỉ dừng ở mức mô tả ngắn gọn một số khái niệm toán học chủ yếu và thường được áp dụng để xây dựng các phương pháp mã hóa. Chi tiết hơn về cơ sở toán học của mật mã có thể tham khảo tại các tài liệu tham khảo trong bài giảng.

### **1.6.1. Lý thuyết thông tin**

### 1.6.1.1. Độ an toàn của hệ mật

Có 2 quan điểm cơ bản về độ an toàn của một hệ mật.

**1. Độ an toàn tính toán:** Độ đo này liên quan đến những nỗ lực tính toán cần thiết để phá một hệ mật. Một hệ mật là an toàn về mặt tính toán nếu một thuật toán tốt nhất để phá nó cần ít nhất  $N$  phép toán,  $N$  là số rất lớn nào đó. Vấn đề ở chỗ, không có một hệ mật thực tế đã biết nào có thể được chứng tỏ là an toàn theo định nghĩa này. Trên thực tế, gọi một hệ mật là "an toàn về mặt tính toán" nếu có một phương pháp tốt nhất phá hệ này nhưng yêu cầu thời gian lớn đến mức không chấp nhận được.

Một quan điểm chứng minh về độ an toàn tính toán là quy độ an toàn của một hệ mật về một bài toán đã được nghiên cứu và bài toán này được coi là khó. Ví dụ, có thể chứng minh một khẳng định: "Một hệ mật đã cho là an toàn nếu không thể phân tích ra thừa số một số nguyên  $n$  cho trước". Các hệ mật loại này đôi khi gọi là "An toàn chứng minh được". Tuy nhiên cần phải hiểu rằng, quan điểm này chỉ cung cấp một chứng minh về độ an toàn có liên quan để một bài toán khác chứ không phải là một chứng minh hoàn chỉnh về độ an toàn.

**2. Độ an toàn không điều kiện:** Độ đo này liên quan đến độ an toàn của các hệ mật khi không có một hạn chế nào được đặt ra về khối lượng tính toán mà kẻ tấn công được phép thực hiện. Một hệ mật được gọi là an toàn không điều kiện nếu nó không thể bị phá trong trường hợp lý tưởng (với khả năng tính toán không hạn chế).

Mặt khác độ an toàn không điều kiện của một hệ mật không thể được đánh giá theo quan điểm độ phức tạp tính toán vì thời gian tính toán cho phép không hạn chế. Ở đây lý thuyết xác suất là nền tảng thích hợp để nghiên cứu về độ an toàn không điều kiện.

#### Định nghĩa 1.1:

*Giả sử  $X$  và  $Y$  là các biến ngẫu nhiên. Ký hiệu xác suất để  $X$  nhận giá trị  $x$  là  $p(x)$  và để  $Y$  nhận giá trị  $y$  là  $p(y)$ . Xác suất đồng thời  $p(x,y)$  là xác suất để  $X$  nhận giá trị  $x$  và  $Y$  nhận giá trị  $y$ . Xác suất có điều kiện  $p(x|y)$  là xác suất để  $X$  nhận giá trị  $x$  với điều kiện  $Y$  nhận giá trị  $y$ . Các biến ngẫu nhiên  $X$  và  $Y$  được gọi là độc lập nếu  $p(x,y) = p(x)p(y)$  với mọi giá trị có thể  $x$  của  $X$  và  $y$  của  $Y$ .*

Quan hệ giữa xác suất đồng thời và xác suất có điều kiện được biểu thị theo công thức:

$$p(x,y)= p(x/y) p(y) \quad (1.8)$$

Đổi chỗ  $x$  và  $y$  có:

$$p(x,y)= p(y/x) p(x) \quad (1.9)$$

Từ hai biểu thức trên có thể rút ra kết quả sau:(được gọi là định lý Bayes)

### Định lý 1.1: (Định lý Bayes)

Nếu  $p(y) > 0$  thì:

$$p\left(\frac{x}{y}\right)=\frac{p(x)p(y/x)}{p(y)} \quad (1.10)$$

**Hệ quả 1.1.**  $x$  và  $y$  là các biến độc lập khi và chỉ khi:  $p(x/y)= p(x) , \forall x,y$ .

Giả sử rằng, một khoá cụ thể chỉ dùng cho một bản mã. Giả sử có một phân bố xác suất trên không gian thông điệp  $P$ . Ký hiệu xác suất tiên nghiệm về thông điệp xuất hiện là  $p_P(x)$ . Cũng giả sử rằng, khóa  $K$  được chọn (bởi Alice và Bob) theo một phân bố xác suất xác định nào đó. Thông thường khoá được chọn ngẫu nhiên, bởi vậy tất cả các khoá sẽ đồng khả năng, tuy nhiên đây không phải là điều bắt buộc. Ký hiệu xác suất để khóa  $K$  được chọn là  $p_K(K)$ . Cần nhớ rằng khóa được chọn trước khi Alice biết thông điệp. Bởi vậy có thể giả định rằng khoá  $K$  và thông điệp  $x$  là các sự kiện độc lập.

Hai phân bố xác suất trên  $P$  và  $K$  sẽ tạo ra một phân bố xác suất trên  $C$ . Thật vậy, có thể dễ dàng tính được xác suất  $p_C(y)$  với  $y$  là bản mã được gửi đi. Với một khoá  $K \in K$ , xác định:

$$C(K)=\{e_K(x): x \in P\} \quad (1.11)$$

Ở đây  $C(K)$  biểu thị tập các bản mã có thể nếu  $K$  là khóa. Khi đó với mỗi  $y \in C$ , có:

$$p_C(y)= \sum_{\{K: y \in C(K)\}} p_K(K) p_P(d_K(y)) \quad (1.12)$$

Nhận thấy rằng, với bất kì  $y \in C$  và  $x \in P$ , có thể tính được xác suất có điều kiện  $p_C(y/x)$ . (Tức là xác suất để  $y$  là bản mã với điều kiện thông điệp là  $x$ ):

$$p_C(y|x) = \sum_{\{K: x=d_K(y)\}} p_K(K) \quad (1.13)$$

Từ đó có thể tính được xác suất có điều kiện  $p_P(x/y)$  (tức xác suất để  $x$  là thông điệp với điều kiện  $y$  là bản mã) bằng cách dùng định lý Bayes. Công thức thu được như sau:

$$p_P(x|y) = \frac{p_P(x) \sum_{\{K: x=d_K(y)\}} p_K(K)}{\sum_{\{K: y \in C(K)\}} p_K(K) p_P(d_K(y))} \quad (1.14)$$

Các phép tính này có thể thực hiện được nếu biết được các phân bố xác suất. Sau đây sẽ trình bày một ví dụ đơn giản để minh họa việc tính toán các phân bố xác suất này.

*Ví dụ:*

Giả sử  $P = \{a, b\}$  với  $p_P(a) = 1/4$ ,  $p_P(b) = 3/4$

Cho  $K = \{K_1, K_2, K_3\}$  với  $p_K(K_1) = 1/2$ ,  $p_K(K_2) = p_K(K_3) = 1/4$

Giả sử  $C = \{1, 2, 3, 4\}$  và các hàm mã được xác định là:

$e_{K_1}(a) = 1, e_{K_1}(b) = 2, e_{K_2}(a) = 2, e_{K_2}(b) = 3, e_{K_3}(a) = 3, e_{K_3}(b) = 4$

Hệ mật này được biểu thị bằng ma trận mã hóa sau:

	$a$	$b$
$K_1$	1	2
$K_2$	2	3
$K_3$	2	4

Tính phân bố xác suất  $p_C$  có:

$$p_C(1) = 1/8$$

$$p_C(2) = 3/8 + 1/16 = 7/16$$

$$p_C(3) = 3/16 + 1/16 = 1/4$$

$$p_C(4) = 3/16$$



Bây giờ đã có thể các phân bố xác suất có điều kiện trên thông điệp với điều kiện đã biết bản mã. Từ đó có:

$$p_P(a/1)=1 \quad p_P(b/1)=0 \quad p_P(a/2)=1/7 \quad p_P(b/2)=6/7$$

$$p_P(a/3)=1/4 \quad p_P(b/3)=3/4 \quad p_P(a/4)=0 \quad p_P(b/4)=1$$

Bây giờ đã có đủ điều kiện để xác định khái niệm về độ mật hoàn thiện. Một cách không hình thức, độ mật hoàn thiện có nghĩa là kẻ tấn công với bản mã trong tay không thể thu được thông tin gì về thông điệp. Ý tưởng này sẽ được làm chính xác bằng cách phát biểu nó theo các thuật ngữ của các phân bố xác suất định nghĩa ở trên.

### Định nghĩa 1.2.

*Một hệ mật có độ mật hoàn thiện nếu  $p_P(x/y) = p_P(x)$  với mọi  $x \in P, y \in C$ . Tức xác suất hậu nghiệm để thông điệp là  $x$  với điều kiện đã thu được bản mã  $y$  là đồng nhất với xác suất tiên nghiệm để thông điệp là  $x$ .*

Trong ví dụ trên chỉ có bản mã 3 mới thoả mãn tính chất độ mật hoàn thiện, các bản mã khác không có tính chất này.

Sau đây sẽ nghiên cứu độ mật hoàn thiện trong trường hợp chung. Trước tiên thấy rằng, (sử dụng định lý Bayes) điều kiện để  $p_P(x/y) = p_P(x)$  với mọi  $x \in P, y \in P$  là tương đương với  $p_C(y/x) = p_C(y)$  với mọi  $x \in P, y \in P$ .

Giả sử rằng  $p_C(y) > 0$  với mọi  $y \in C$  ( $p_C(y) = 0$ ) thì bản mã sẽ không được dùng và có thể loại khỏi  $C$ . Cố định một giá trị nào đó  $x \in P$ , với mỗi  $y \in C$  có  $p_C(y/x) = p_C(y) > 0$ . Bởi vậy, với mỗi  $y \in C$  phải có ít nhất một khoá  $K$  và một  $x$  sao cho  $e_K(x) = y$ . Điều này dẫn đến  $|K| \geq |C|$ . Trong một hệ mật bất kỳ phải có  $|C| \geq |P|$  vì mỗi quy tắc mã hóa là một đơn ánh. Trong trường hợp giới hạn,  $|K| = |C| = |P|$ , có định lý sau (Theo Shannon).

### Định lý 1.2.

*Giả sử  $(P, C, K, E, D)$  là một hệ mật, trong đó  $|K| = |C| = |P|$ . Khi đó, hệ mật có độ mật hoàn thiện khi và chỉ khi khoá  $K$  được dùng với xác suất như nhau bằng  $1/|K|$ , và với mỗi  $x \in P$ , mỗi  $y \in C$  có một khoá duy nhất  $K$  sao cho  $e_K(x) = y$*

Chứng minh: Giả sử hệ mật đã cho có độ mật hoàn thiện. Như đã thấy ở trên: với mỗi  $x \in P$ , mỗi  $y \in C$ , phải có ít nhất một khoá  $K$  sao cho  $e_K(x)=y$ . Bởi vậy có bất đẳng thức:

$$|C| = |\{e_K(x): K \in K\}| = |K|$$

Tuy nhiên, giả sử rằng  $|C| = |K|$ , bởi vậy phải có:

$$|\{e_K(x): K \in C\}| = |K|$$

Tức là ở đây không tồn tại hai khoá  $K_1$  và  $K_2$  khác nhau để  $e_{K_1}(x)=e_{K_2}(x)=y$ . Như vậy đã chứng minh được rằng, với bất kỳ  $x \in P$  và  $y \in C$  có đúng một khoá  $K$   $e_K(x)=y$ .

Ký hiệu  $n=|K|$ . Giả sử  $P=\{x_i: 1 \leq i \leq n\}$  và cố định một giá trị  $y \in C$ . Có thể ký hiệu các khoá  $K_1, K_2, \dots, K_n$  sao cho  $e_{K_i}(x_i)=y_i, 1 \leq i \leq n$ . Sử dụng định lý Bayes có:

$$p_P(x_i | y) = \frac{p_C(y | x_i) p_P(x_i)}{p_C(y)} = \frac{p_K(K_i) \cdot (p_P(x_i))}{p_C(y)} \quad (1.15)$$

Xét điều kiện độ mật hoàn thiện  $p_P(x_i|y)=p_P(x_i)$ . Điều kiện này kéo theo  $p_K(K_i)=p_C(y)$  với  $1 \leq i \leq n$ . Tức là khoá được dùng với xác suất như nhau (chính bằng  $p_C(y)$ ). Tuy nhiên vì số khoá là  $n=|K|$  nên có  $p_K(K)=1/|K|$  với mỗi  $K \in K$

Mật mã khoá sử dụng một lần của Vernam (One-Time-Pad: OTP) là một ví dụ quen thuộc về hệ mật có độ mật hoàn thiện. Gillbert Vernam lần đầu tiên mô tả hệ mật này vào năm 1917. Hệ OTP dùng để mã hóa và giải mã tự động các thông điệp điện báo. Điều thú vị là trong nhiều năm OTP được coi là một hệ mật không thể bị phá nhưng không thể chứng minh cho tới khi Shannon xây dựng được khái niệm về độ mật hoàn thiện hơn 30 năm sau đó.

Giả sử  $n$  ( $n \geq 1$ ) là số nguyên và  $P=C=K=(\mathbb{Z}_2)^n$ . Với  $K \in (\mathbb{Z}_2)^n$ , xác định  $e_K(x)$  là tổng vector theo modulo 2 của  $K$  và  $x$  (hay tương đương với phép hoặc loại trừ của hai dãy bit tương ứng). Như vậy, nếu  $x=(x_1, \dots, x_n)$  và  $K=(K_1, \dots, K_n)$  thì:

$$e_K(x) = (x_1 + K_1, \dots, x_n + K_n) \bmod 2$$

Phép mã hóa là đồng nhất với phép giải mã. Nếu  $y=(y_1, \dots, y_n)$  thì:

$$d_K(x) = (y_1 + K_1, \dots, y_n + K_n) \bmod 2$$

### 1.6.1.2. Entropy

Phần trước đã trình bày về khái niệm độ mật hoàn thiện và thể hiện mối quan tâm vào một trường hợp đặc biệt, khi một khoá chỉ được dùng cho một lần mã. Trong phần này, bài giảng sẽ nghiên cứu khi có nhiều thông điệp được mã hóa bằng cùng một khoá và bằng cách nào mà thám mã có thể thực hiện có kết quả phép tấn công chỉ với bản mã trong thời gian đủ lớn. Công cụ cơ bản trong nghiên cứu bài toán này là khái niệm Entropy. Đây là khái niệm trong lý thuyết thông tin do Shannon đưa ra vào năm 1948. Có thể coi Entropy là đại lượng đo thông tin hay còn gọi là độ bất định. Nó được tính như một hàm của phân bố xác suất.

Giả sử có một biến ngẫu nhiên  $X$  nhận các giá trị trên một tập hữu hạn theo một phân bố xác suất  $p(X)$ . Thông tin thu nhận được bởi một sự kiện xảy ra tuân theo một phân bố  $p(X)$  là gì?. Tương tự, nếu sự kiện còn chưa xảy ra thì cái gì là độ bất định và kết quả bằng bao nhiêu?. Đại lượng này được gọi là Entropy của  $X$  và được ký hiệu là  $H(X)$ .

Các ý tưởng này có vẻ như khá trừu tượng, bởi vậy sẽ xét một ví dụ cụ thể hơn. Giả sử biến ngẫu nhiên  $X$  biểu thị phép tung đồng xu. Phân bố xác suất là:  $p(\text{mặt sấp}) = p(\text{mặt ngửa}) = 1/2$ . Có thể nói rằng, thông tin (hay Entropy) của phép tung đồng xu là một bit vì có thể mã hóa mặt sấp bằng 1 và mặt ngửa bằng 0. Tương tự Entropy của  $n$  phép tung đồng tiền có thể mã hóa bằng một xâu bit có độ dài  $n$ .

### Định nghĩa 1.3

*Giả sử  $X$  là một biến ngẫu nhiên lấy các giá trị trên một tập hữu hạn theo phân bố xác suất  $p(X)$ . Khi đó Entropy của phân bố xác suất này được định nghĩa là lượng:*

$$H(X) = -\sum_{i=1}^n p_i \log_2 P_i \quad (1.16)$$

Nếu các giá trị có thể của  $X$  là  $x_i$ ,  $1 \leq i \leq n$  thì có:

$$H(X) = - \sum_{i=1}^n p(X = x_i) \log_2 P(X = x_i) \quad (1.17)$$

Nhận xét: Nhận thấy rằng  $\log_2 p_i$  không xác định nếu  $p_i=0$ . Bởi vậy đôi khi Entropy được định nghĩa là tổng tương ứng trên tất cả các xác suất khác 0. Vì  $\lim_{x \rightarrow 0} x \log_2 x = 0$  nên trên thực tế cũng không có trở ngại gì nếu cho  $p_i=0$  với giá trị  $i$  nào đó. Tuy nhiên tuân theo giả định là khi tính Entropy của một phân bố xác suất  $p_i$ , tổng trên sẽ được lấy trên các chỉ số  $i$  sao cho  $p_i \neq 0$  cũng thấy rằng việc chọn cơ sở của logarit là tùy ý; cơ sở này không nhất thiết phải là 2. Với một cơ sở khác sẽ chỉ làm thay đổi giá trị của Entropy đi một hằng số.

Chú ý rằng, nếu  $p_i = 1/n$  với  $1 \leq i \leq n$  thì  $H(X) = \log_2 n$  thì dễ dàng thấy rằng  $H(X) \geq 0$  và  $H(X) = 0$  khi và chỉ khi  $p_i = 1$  với một giá trị  $i$  nào đó và  $p_j = 0$  với  $j \neq i$ .

Xét Entropy của các thành phần khác nhau của một hệ mật. Có thể coi khoá là một biến ngẫu nhiên  $K$  nhận các giá trị tuân theo phân bố xác suất  $p_K$  và bởi vậy có thể tính được  $H(K)$ . Tương tự có thể tính các Entropy  $H(P)$  và  $H(C)$  theo các phân bố xác suất tương ứng của bản mã và thông điệp.

Các tính chất của Entropy: Trong phần này sẽ chứng minh một số kết quả quan trọng liên quan đến Entropy. Trước tiên sẽ nghiên cứu về bất đẳng thức Jensen. Bất đẳng thức Jensen có liên quan đến hàm lồi có định nghĩa như sau:

#### **Định nghĩa 1.4**

Một hàm có giá trị thực  $f$  là lồi trên khoảng  $I$  nếu:

$$f\left(\frac{x+y}{2}\right) \geq \frac{f(x)+f(y)}{2} \quad (1.18)$$

với mọi  $x, y \in I$ ,  $f$  là hàm lồi thực sự trên khoảng  $I$  nếu:

$$f\left(\frac{x+y}{2}\right) > \frac{f(x)+f(y)}{2} \quad (1.19)$$

với mọi  $x, y \in I$ ,  $x \neq y$ .

**Định lý 1.3: (Bất đẳng thức Jensen không chứng minh)**

Giả sử  $h$  là một hàm lồi thực sự và liên tục trên khoảng  $I$ ,  $\sum_{i=1}^n a_i = 1$  và  $a_i > 0$ ;  $1 \leq i \leq n$ .

Khi đó:

$$\sum_{i=1}^n a_i f(x_i) \leq f\left(\sum_{i=1}^n a_i x_i\right) \quad (1.20)$$

trong đó  $x_i \in I$ ;  $1 \leq i \leq n$ . Ngoài ra dấu “=” chỉ xảy ra khi và chỉ khi  $x_1 = x_2 = \dots = x_n$

Tiếp theo sẽ đưa ra một số kết quả về Entropy. Trong định lý 1.4 sử dụng khẳng định: hàm  $\log_2 x$  là một hàm lồi thực sự trong khoảng  $(0, \infty)$ .

#### **Định lý 1.4**

Giả sử  $X$  là biến ngẫu nhiên có phân bố xác suất  $p_1, p_2, \dots, p_n$  trong đó  $p_i > 0$ ;  $1 \leq i \leq n$ . Khi đó  $H(X) < \log_2 n$ . Dấu “=” xảy ra khi và chỉ khi  $p_i = 1/n$ ,  $1 \leq i \leq n$

Chứng minh:

Áp dụng bất đẳng thức Jensen có:

$$\begin{aligned} H(X) &= -\sum_{i=1}^n p_i \log_2 p_i = \sum_{i=1}^n p_i \log_2 (1/p_i) \\ &\leq \log_2 \sum_{i=1}^n (p_i \times 1/p_i) = \log_2 n \end{aligned}$$

Ngoài ra, dấu “=” chỉ xảy ra khi và chỉ khi  $p_i = 1/n$ ,  $1 \leq i \leq n$ .

#### **Định lý 1.5**

$$H(X, Y) \leq H(X) + H(Y) \quad (1.21)$$

Đẳng thức xảy ra khi và chỉ khi  $X$  và  $Y$  là các biến cố độc lập

Chứng minh:

Giả sử  $X$  nhận các giá trị  $x_i$ ,  $1 \leq i \leq m$ ;  $Y$  nhận các giá trị  $y_j$ ,  $1 \leq j \leq n$ .

Ký hiệu:

$$p_i = p(X = x_i), 1 \leq i \leq m$$

$$q_j = p(Y = y_j), 1 \leq j \leq n$$

$$r_{ij} = p(X = x_i, Y = y_j), 1 \leq i \leq m, 1 \leq j \leq n \text{ (Đây là phân bố xác suất).}$$

Nhận thấy rằng:

$$p_i = \sum_{j=1}^n r_{ij}; (1 \leq i \leq m)$$

$$q_j = \sum_{i=1}^m r_{ij}; (1 \leq j \leq n)$$

Có:

$$\begin{aligned} H(X) + H(Y) &= -\left(\sum_{i=1}^m p_i \log_2 p_i + \sum_{j=1}^n q_j \log_2 q_j\right) \\ &= -\left(\sum_{i=1}^m \sum_{j=1}^n r_{ij} \log_2 p_i + \sum_{j=1}^n \sum_{i=1}^m r_{ij} \log_2 q_j\right) \\ &= -\sum_{i=1}^m \sum_{j=1}^n r_{ij} \log_2 p_i q_j \end{aligned}$$

$$\text{Mặt khác: } H(X, Y) = -\sum_{i=1}^m \sum_{j=1}^n r_{ij} \log_2 r_{ij}$$

Kết hợp lại được kết quả sau:

$$H(X, Y) - H(X) - H(Y) = \sum_{i=1}^m \sum_{j=1}^n r_{ij} \log_2 (1/r_{ij}) - \sum_{i=1}^m \sum_{j=1}^n r_{ij} \log_2 p_i q_j \leq \log_2 \sum_{i=1}^m \sum_{j=1}^n p_i q_j = \log_2 1 = 0$$

(Ở đây đã áp dụng bất đẳng thức Jensen khi biết rằng các  $r_{ij}$  tạo nên một phân bố xác suất)

Khi đẳng thức xảy ra, có thể thấy rằng phải có một hằng số  $c$  sao cho  $p_i q_j / r_{ij} = c$  với mọi  $i, j$ . Áp dụng đẳng thức sau:

$$\sum_{i=1}^m \sum_{j=1}^n r_{ij} = \sum_{i=1}^m \sum_{j=1}^n p_i q_j = 1$$

Điều này dẫn đến  $c=1$ . Bởi vậy đẳng thức xảy ra khi và chỉ khi  $r_{ij} = p_i q_j$ , nghĩa là:

$$P(X=x_i, Y=y_j) = p(X=x_i)p(Y=y_j) \text{ với } 1 \leq i \leq m, 1 \leq j \leq n.$$

Điều này có nghĩa là  $X$  và  $Y$  độc lập.

Tiếp theo sẽ đưa ra khái niệm Entropy có điều kiện:

**Định nghĩa 1.5:**

Giả sử  $X$  và  $Y$  là hai biến ngẫu nhiên. Khi đó với giá trị xác định bất kỳ  $y$  của  $Y$  có một phân bố xác suất có điều kiện  $p(X|y)$ . Rõ ràng là:

$$H(X|y) = -\sum_x p(x|y) \log_2 p(x|y) \quad (1.22)$$

**Định nghĩa 1.6:**

Entropy có điều kiện  $H(X|Y)$  là trung bình có trọng số (ứng với các xác suất  $p(y)$  của Entropy  $H(X|y)$  trên mọi giá trị có thể  $y$ .  $H(X|y)$  được tính bằng:

$$H(X|Y) = -\sum_y \sum_x p(y) p(x|y) \log_2 p(x|y) \quad (1.23)$$

Entropy có điều kiện đo lượng thông tin trung bình về  $X$  do  $Y$  mang lại.

Sau đây là 2 kết quả trực tiếp.

**Định lý 1.6:**

$$H(X,Y) = H(Y) + H(X|Y) \quad (1.24)$$

**Hệ quả 1.2:**

$$H(X|Y) \leq H(X)$$

Dấu “=” xảy ra khi và chỉ khi  $X$  và  $Y$  độc lập.

Các khóa giả và khoảng duy nhất: Trong phần này sẽ áp dụng các kết quả về Entropy ở trên cho các hệ mật. Trước tiên sẽ chỉ ra một quan hệ cơ bản giữa các Entropy của các thành phần trong hệ mật. Entropy có điều kiện  $H(K|C)$  được gọi là độ bất định về khoá. Nó cho biết về lượng thông tin về khoá thu được từ bản mã.

**Định lý 1.7:**

Giả sử  $(P, C, K, E, D)$  là một hệ mật. Khi đó:

$$H(K|C) = H(K) + H(P) - H(C) \quad (1.25)$$

Giả sử  $(P, C, K, E, D)$  là hệ mật đang được sử dụng. Một xâu của thông điệp  $x_1, x_2, \dots, x_n$  sẽ được mã hóa bằng một khoá để tạo ra bản mã  $y_1, y_2, \dots, y_n$ . Nhớ lại rằng, mục

đích cơ bản của thám mã là phải xác định được khoá. Xem xét các phương pháp tấn công chỉ với bản mã và coi Oscar có khả năng tính toán vô hạn. Cũng giả sử Oscar biết thông điệp là một văn bản theo ngôn ngữ tự nhiên (chẳng hạn văn bản tiếng Anh). Nói chung Oscar có khả năng rút ra một số khoá nhất định (các khoá có thể hay các khoá chấp nhận được) nhưng trong đó chỉ có một khoá đúng, các khoá có thể còn lại (các khoá không đúng) được gọi là các khoá giả.

Ví dụ, giả sử Oscar thu được một xâu bản mã **WNAJW** mã bằng phương pháp mã dịch vòng (**MDV**). Dễ dàng thấy rằng, chỉ có hai xâu thông điệp có ý nghĩa là *river* và *arena* tương ứng với các khoá  $F(= 5)$  và  $W(= 22)$ . Trong hai khoá này chỉ có một khoá đúng, khoá còn lại là khoá giả. Trên thực tế, việc tìm một bản mã của **MDV** có độ dài 5 và 2 bản giải mã có nghĩa không phải quá khó khăn, bạn đọc có thể tìm ra nhiều ví dụ khác. Mục đích của là phải tìm ra giới hạn cho số trung bình các khoá giả. Trước tiên, phải xác định giá trị này theo Entropy (cho một ký tự) của một ngôn ngữ tự nhiên  $L$  (ký hiệu là  $H_L$ ).  $H_L$  là lượng thông tin trung bình trên một ký tự trong một xâu có nghĩa của thông điệp. Chú ý rằng, một xâu ngẫu nhiên các ký tự của bảng chữ cái sẽ có Entropy trên một ký tự bằng  $\log_2 26 \approx 4,76$ . Có thể lấy  $H(P)$  là xấp xỉ bậc nhất cho  $H_L$ . Trong trường hợp  $L$  là Anh ngữ, tính được  $H(P) \approx 4,19$ . Dĩ nhiên các ký tự liên tiếp trong một ngôn ngữ không độc lập với nhau và sự tương quan giữa các ký tự liên tiếp sẽ làm giảm Entropy. Ví dụ, trong Anh ngữ, chữ Q luôn kéo theo sau là chữ U. Để làm xấp xỉ bậc hai, tính Entropy của phân bố xác suất của tất cả các bộ đôi rồi chia cho 2. Một cách tổng quát, định nghĩa  $P^n$  là biến ngẫu nhiên có phân bố xác suất của tất cả các bộ  $n$  của thông điệp. Sẽ sử dụng tất cả các định nghĩa sau:

**Định nghĩa 1.7:**

*Giả sử  $L$  là một ngôn ngữ tự nhiên. Entropy của  $L$  được xác định là lượng sau:*

$$H_L = \lim_{n \rightarrow \infty} \frac{H(P^n)}{n} \quad (1.26)$$

Độ dư của  $L$  là:  $R_L = 1 - (H_L / \log_2 |P|)$  (1.27)



**Nhận xét:**  $H_L$  đo Entropy trên mỗi ký tự của ngôn ngữ  $L$ . Một ngôn ngữ tự nhiên sẽ có Entropy là  $\log_2/P$ . Bởi vậy đại lượng  $R_L$  đo phần "ký tự vượt trội" là phần dư. Trong trường hợp Anh ngữ, dựa trên bảng chứa một số lớn các bộ đôi và các tần số, có thể tính được  $H(P^2)$ . Ước lượng theo cách này, tính được  $H(P^2) \approx 3,90$ . Cứ tiếp tục như vậy bằng cách lập bảng các bộ ba v.v... thu được ước lượng cho  $H_L$ . Trên thực tế, bằng nhiều thực nghiệm khác nhau, có thể đi tới kết quả sau  $1,0 \leq H_L \leq 1,5$ . Tức là lượng thông tin trung bình trong tiếng Anh vào khoảng 1 bit tới 1,5 bit trên mỗi ký tự.

Giả sử lấy 1,25 là giá trị ước lượng của giá trị của  $H_L$ . Khi đó độ dư vào khoảng 0,75. Tức là tiếng Anh có độ dư vào khoảng 75%! (Điều này không có nghĩa loại bỏ tùy ý 3 trên 4 ký tự của một văn bản tiếng Anh mà vẫn có khả năng đọc được nó. Nó chỉ có nghĩa là tìm được một phép mã Huffman cho các bộ  $n$  với  $n$  đủ lớn, phép mã này sẽ nén văn bản tiếng Anh xuống còn 1/4 độ dài của bản gốc). Với các phân bố xác suất  $C^n$  đã cho trên  $K$  và  $P^n$ . Có thể xác định phân bố xác suất trên là tập các bộ  $n$  của bản mã. Đã xác định  $P^n$  là biến ngẫu nhiên biểu diễn bộ  $n$  của thông điệp. Tương tự  $C^n$  là biến ngẫu nhiên biểu thị bộ  $n$  của bản mã.

### **Định lý 1.8:**

*Giả sử  $(P, C, K, E, D)$  là một hệ mật trong đó  $|C|=|P|$  và các khóa được chọn đồng xác suất. Giả sử  $R_L$  là độ dư của ngôn ngữ gốc. Khi đó với một xâu bản mã độ dài  $n$  cho trước ( $n$  đủ lớn), số trung bình các khóa giả  $s_n$  thỏa mãn bất đẳng thức sau:*

$$\bar{s} \geq \{ |K| / (|P| n R_L) \} - 1$$

Lượng  $|K| / (|P| n R_L) - 1$  tiến tới 0 theo hàm mũ khi  $n$  tăng. Ước lượng này có thể không chính xác với các giá trị  $n$  nhỏ. Đó là do  $H(P^n)/n$  không phải là một ước lượng tốt cho  $H_L$  nếu  $n$  nhỏ.

### **Định nghĩa 1.8:**

*Khoảng duy nhất của một hệ mật được định nghĩa là giá trị của  $n$  mà ứng với giá trị này, số khoá giả trung bình bằng 0 (ký hiệu giá trị này là  $n_0$ ). Điều đó có nghĩa là  $n_0$  là độ dài trung bình cần thiết của bản mã để thám mã có thể tính toán khoá một cách duy nhất với thời gian đủ lớn.*

## 1.6.2. Lý thuyết số

### 1.6.2.1. Số nguyên

Tập các số nguyên:  $\{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\} = \mathbb{Z}$

#### **Định nghĩa 1.9:**

Cho  $a, b \in \mathbb{Z}$ ,  $a$  là ước nguyên của  $b$ , nếu  $\exists c \in \mathbb{Z} : b = a.c$ . Ký hiệu  $a|b$

Các tính chất chia hết:  $\forall a, b, c \in \mathbb{Z}$  có:

- ✓  $a|a$
- ✓ Nếu  $a|b$  và  $b|c$  thì  $a|c$
- ✓ Nếu  $a|b$  và  $a|c$  thì  $a|(bx+cy) \quad \forall x, y \in \mathbb{Z}$
- ✓ Nếu  $a|b$  và  $b|a$  thì  $a = \pm b$

#### **Định nghĩa 1.10:** Thuật toán chia đối với các số nguyên:

Nếu  $a$  và  $b$  là các số nguyên với  $b \geq 1$  thì  $a = qb + r$ ,  $0 \leq r < b$  và  $r$  là duy nhất.

Phần dư của phép chia  $a$  và  $b$  được ký hiệu  $a \bmod b = r$ .

Thương của phép chia  $a$  và  $b$  được ký hiệu  $a \div b = q$ .

$$\text{Có } a \div b = \left\lfloor \frac{a}{b} \right\rfloor, \quad a \bmod b = a - b \left\lfloor \frac{a}{b} \right\rfloor$$

**Ví dụ:**  $a=73, b=17 \Rightarrow 73 \div 17=4, 73 \bmod 17=5$

#### **Định nghĩa 1.11:** Ước chung $c$ là ước chung của $a$ và $b$ nếu $c|a$ và $c|b$

#### **Định nghĩa 1.12:** Ước chung lớn nhất (ƯCLN)

Số nguyên dương  $d$  là ƯCLN của các số nguyên  $a$  và  $b$ , ký hiệu  $d=(a,b)$  nếu:

- ✓  $d$  là ước chung của  $a$  và  $b$ .
- ✓ Nếu có  $c|a$  và  $c|b$  thì  $c|d$

Như vậy  $(a,b)$  là số nguyên dương lớn nhất là ước của cả  $a$  và  $b$  không kễ  $(0,0)=0$ .

**Ví dụ:** Các ước chung của 12 và 18 là  $\{\pm 1, \pm 2, \pm 3, \pm 6\} \rightarrow (12,18)=6$

#### **Định nghĩa 1.13:** Bội chung nhỏ nhất (BCNN)

Số nguyên dương  $d$  là BCNN của các số nguyên  $a$  và  $b$ , ký hiệu  $d=BCNN(a,b)$  nếu:

✓  $a/d, b/d$ .

✓ Nếu có  $a|c, b|c$  thì  $d|c$

Như vậy  $d$  là số nguyên dương nhỏ nhất là bội của cả  $a$  và  $b$ .

**Tính chất:**

$$BCNN(a,b) = \frac{a.b}{(a,b)}$$

**Ví dụ:**  $(12,18)=6 \Rightarrow BCNN(12,18)=\frac{12.18}{6}=36$

**Định nghĩa 1.14:** Hai số nguyên dương  $a$  và  $b$  được gọi là nguyên tố cùng nhau nếu  $(a,b)=1$ .

**Định nghĩa 1.15:** Số nguyên  $p \geq 2$  được gọi là số nguyên tố nếu các ước dương của nó chỉ là 1 và  $p$ . Ngược lại  $p$  được gọi là hợp số.

**Định lý 1.9: (Định lý cơ bản của số học)** Với mỗi số nguyên  $n \geq 2$  luôn phân tích được dưới dạng tích của lũy thừa các số nguyên tố.

$$n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k} \quad (1.28)$$

Trong đó  $p_i$  là các số nguyên tố khác nhau và  $e_i$  là các số nguyên dương. Hơn nữa phân tích trên là duy nhất.

**Định nghĩa 1.16:** Với  $n \geq 2$ , hàm  $\varphi(n)$  được xác định là các số nguyên trong khoảng  $[1,n]$  nguyên tố cùng nhau với  $n$ .

**Các tính chất của hàm  $\varphi(n)$**

- Nếu  $p$  là số nguyên tố thì  $\varphi(p) = p - 1$
- Nếu  $(m,n)=1$  thì  $\varphi(m.n) = \varphi(m).\varphi(n)$
- Nếu  $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$  là phân tích ra thừa số nguyên tố của  $n$  thì:

$$\varphi(n) = n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \dots (1 - \frac{1}{p_k}) \quad (1.29)$$

### 1.6.2.2. Khái niệm đồng dư modulo

Giả sử  $a$  và  $b$  là các số nguyên và  $m$  là một số nguyên dương. Khi đó viết  $a \equiv b \pmod{m}$  nếu  $a$  và  $b$  khi chia cho  $m$  có cùng số dư, tức là  $(a-b)$  chia hết cho  $m$ . Mệnh đề  $a \equiv b \pmod{m}$  được gọi là “ $a$  đồng dư với  $b$  theo modun  $m$ ”.

Chứng minh: Giả sử chia  $a$  và  $b$  cho  $m$  và thu được thương nguyên và phần dư, các phần dư nằm giữa 0 và  $m-1$ . Nghĩa là:

$$a = q_1 m + r_1 \text{ và } b = q_2 m + r_2$$

Trong đó :  $0 \leq r_1 \leq m-1$  và  $0 \leq r_2 \leq m-1$ . Khi đó có thể dễ dàng thấy rằng:  $a \equiv b \pmod{m}$  khi và chỉ khi  $r_1 = r_2$

Tiếp theo sẽ dùng ký hiệu  $a \bmod m$  để xác định phần dư khi  $a$  được chia cho  $m$  (chính là giá trị  $r_1$  ở trên). Như vậy:  $a \equiv b \pmod{m}$  khi và chỉ khi:  $(a \bmod m) = (b \bmod m)$

Nếu thay giá trị của  $a$  bằng giá trị  $(a \bmod m)$  thì nói  $a$  được rút gọn theo modulo  $m$

### 1.6.2.3. Số học modulo

**Định nghĩa số học modun  $m$ :**

$Z_m$  được coi là tập hợp  $\{0, 1, \dots, m-1\}$  có trang bị hai phép toán cộng và nhân. Việc cộng và nhân trong  $Z_m$  được thực hiện giống như cộng và nhân các số thực ngoại trừ một điểm là các kết quả được rút gọn theo modun  $m$ .

**Ví dụ:**  $Z_{26}$  được coi là tập hợp  $\{0, 1, 2, \dots, 25\}$  có trang bị hai phép toán cộng và nhân.

$$25 + 7 = 32 \bmod 26 = 6 \quad (\text{ví dụ K=7, 'Z' } \rightarrow \text{'H'})$$

$$5 * 9 = 45 \bmod 26 = 19$$

**Định lý về đồng dư thức**

Đồng dư thức  $ax \equiv b \pmod{m}$  chỉ có một nghiệm duy nhất  $x \in Z_m$  với mọi  $b \in Z_m$  khi và chỉ khi  $\text{UCLN}(a, m) = 1$ .

**Chứng minh:**

- Giả sử rằng,  $\text{UCLN}(a, m) = d > 1$ .

- Với  $b = 0$  thì đồng dư thức  $ax \equiv 0 \pmod{m}$  sẽ có ít nhất hai nghiệm phân biệt trong  $Z_m$  là  $x = 0$  và  $x = m/d$ .

### Khái niệm phần tử nghịch đảo:

- Giả sử  $a \in Z_m$ .
- Phần tử nghịch đảo (theo phép nhân) của  $a$  là phần tử  $a^{-1} \in Z_m$  sao cho:

$$aa^{-1} \equiv a^{-1}a \equiv 1 \pmod{m}$$

### Tính chất

- $a$  có nghịch đảo theo modun  $m$  khi và chỉ khi  $\text{UCLN}(a, m) = 1$ ,
- Nếu nghịch đảo tồn tại thì phải là duy nhất.
- Nếu  $b = a^{-1}$  thì  $a = b^{-1}$ .
- Nếu  $m$  là số nguyên tố thì mọi phần tử khác không của  $Z_m$  đều có nghịch đảo.

### 1.6.3. Độ phức tạp thuật toán

#### 1.6.3.1. Khái niệm về thuật toán

Có thể định nghĩa thuật toán theo nhiều cách khác nhau. Ở đây sẽ hiểu khái niệm thuật toán theo một cách thông thường nhất.

**Định nghĩa:** Thuật toán là một quy tắc để với những dữ liệu ban đầu đã cho, tìm được lời giải của bài toán được xét sau một khoảng thời gian hữu hạn.

Để minh họa cách ghi một thuật toán cũng như tìm hiểu các yêu cầu đề ra cho thuật toán, xét trên các ví dụ cụ thể sau đây:

Cho  $n$  số  $X[1], X[2], \dots, X[n]$ , cần tìm  $m$  và  $j$  sao cho:  $m = X[j] = \max_{1 \leq k \leq n} X[k]$  và  $j$  là lớn nhất có thể. Điều đó có nghĩa là cần tìm cực đại của các số đã cho và chỉ số lớn nhất trong các số cực đại. Với mục tiêu tìm số cực đại với chỉ số lớn nhất, xuất phát từ giá trị  $X[n]$ . Bước thứ nhất, vì mới chỉ có một số có thể tạm thời xem  $m = X[n]$  và  $j = n$ . Tiếp theo so sánh  $X[n]$  với  $X[n-1]$ . Nếu  $X[n]$  không nhỏ hơn  $X[n-1]$  thì giữ nguyên, trong trường hợp ngược lại,  $X[n-1]$  chính là số cực đại trong hai số đã xét và phải thay đổi  $m$  và  $j$ . Đặt  $m = X[n-1]$ ,  $j = n-1$ . Với cách làm như trên, ở mỗi bước luôn nhận được số cực

đại trong số những số đã xét. Bước tiếp theo là so sánh nó với những số đứng trước hoặc kết thúc thuật toán trong trường hợp không còn số nào đứng trước nó.

### 1.6.3.2. Độ phức tạp của thuật toán

Thời gian làm việc của máy tính khi chạy một thuật toán nào đó không chỉ phụ thuộc vào thuật toán mà còn phụ thuộc vào máy tính được sử dụng. Vì thế, để có một tiêu chuẩn chung, sẽ đo độ phức tạp của một thuật toán bằng số các phép tính phải làm khi thực hiện thuật toán. Khi tiến hành cùng một thuật toán, số các phép tính phải thực hiện còn phụ thuộc vào cỡ của bài toán, tức là độ lớn của đầu vào. Vì thế độ phức tạp của thuật toán sẽ là một hàm số của độ lớn đầu vào. Trong những ứng dụng thực tiễn, không cần biết chính xác hàm này mà chỉ cần biết “cỡ” của chúng, tức là cần có một ước lượng đủ tốt của chúng. Trong khi làm việc, máy tính thường ghi các chữ số bằng bóng đèn “sáng, tắt”, bóng đèn sáng chỉ số 1, bóng đèn tắt chỉ số 0. Vì thế để thuận tiện nhất là dùng hệ đếm cơ số 2, trong đó để biểu diễn một số, chỉ cần dùng hai ký hiệu 0 và 1. Một ký hiệu 0 hoặc 1 được gọi là 1bit “viết tắt của binary digit”. Một số nguyên  $n$  biểu diễn bởi  $k$  chữ số 1 và 0 được gọi là một số  $k$  bit.

Độ phức tạp của một thuật toán được đo bằng số các phép tính bit. Phép tính bit là một phép tính logic hay số học thực hiện trên các số một bit 0 và 1. Để ước lượng độ phức tạp của thuật toán dùng khái niệm bậc  $O$  lớn.

**Định nghĩa:** Giả sử  $f[n]$  và  $g[n]$  là hai hàm xác định trên tập hợp các số nguyên dương. Nói  $f[n]$  có bậc  $O$  lớn của  $g[n]$  và viết  $f[n] = O(g[n])$ , nếu tồn tại 1 số  $C > 0$  sao cho với  $n$  đủ lớn, các hàm  $f[n]$  và  $g[n]$  đều dương thì  $f[n] < C g[n]$ .

**Ví dụ:**

- Giả sử  $f[n]$  là đa thức có công thức:  $f[n] = a_d n^d + a_{d-1} n^{d-1} + \dots + a_1 n + a_0$  trong đó  $a_d > 0$ . Dễ dàng chứng minh được  $f[n] = O(n^d)$ .
- Nếu  $f_1[n] = O(g[n])$ ,  $f_2[n] = O(g[n])$  thì  $f_1 + f_2 = O(g)$ .
- Nếu  $f_1 = O(g_1)$ ,  $f_2 = O(g_2)$  thì  $f_1 f_2 = O(g_1 g_2)$ .
- Nếu tồn tại giới hạn hữu hạn:

$$\lim_{n \rightarrow \infty} \frac{f[n]}{g[n]}$$

thì  $f=O(g)$

- Với mọi số  $\varepsilon > 0$ ,  $\log n = O(n^\varepsilon)$

**Định nghĩa:**

Một thuật toán được gọi là có độ phức tạp đa thức hoặc có thời gian đa thức, nếu số các phép tính cần thiết để thực hiện thuật toán không vượt quá  $O(\log^d n)$ , trong đó  $n$  là độ lớn của đầu vào và  $d$  là số nguyên dương nào đó.

Nói cách khác nếu đầu vào là các số  $k$  bit thì thời gian thực hiện thuật toán là  $O(k^d)$ , tức là tương đương với một đa thức của  $k$ .

Các thuật toán với thời gian  $O(n^\alpha)$ ,  $\alpha > 0$  được gọi là thuật toán với độ phức tạp mũ hoặc thời gian mũ.

Chú ý rằng nếu một thuật toán nào đó có độ phức tạp  $O(g)$  thì cũng có thể nói nó có độ phức tạp  $O(h)$  với mọi hàm  $h > g$ . Tuy nhiên luôn luôn cố gắng tìm ước lượng tốt nhất có thể để tránh hiểu sai về độ phức tạp thực sự của thuật toán.

Cũng có những thuật toán có độ phức tạp trung gian giữa đa thức và mũ. Thường gọi đó là thuật toán dưới mũ. Chẳng hạn thuật toán nhanh nhất được biết hiện nay để phân tích một số nguyên  $n$  ra thừa số là thuật toán có độ phức tạp:

$$\exp = (\sqrt{\log n \log \log n})$$

Khi giải một bài toán không những chỉ cố gắng tìm ra một thuật toán nào đó, mà còn muốn tìm ra thuật toán “tốt nhất”. Đánh giá độ phức tạp là một trong những cách để phân tích, so sánh và tìm ra thuật toán tối ưu. Tuy nhiên độ phức tạp không phải là tiêu chuẩn duy nhất để đánh giá thuật toán. Có những thuật toán về lý thuyết thì có độ phức tạp cao hơn một thuật toán khác, nhưng khi sử dụng lại có kết quả (gần đúng) nhanh hơn nhiều. Điều này còn tùy thuộc vào những bài toán cụ thể, những mục tiêu cụ thể và cả kinh nghiệm của người sử dụng.

Cần lưu ý thêm một số điểm sau đây: Mặc dù định nghĩa thuật toán đưa ra chưa phải là chặt chẽ, nó vẫn quá “*cứng nhắc*” trong những ứng dụng thực tế. Bởi vậy cần đến các thuật toán “*xác suất*”, tức là các thuật toán phụ thuộc vào một hay nhiều tham số ngẫu nhiên. Những “*thuật toán*” này về nguyên tắc không được gọi là thuật toán vì chúng có thể không bao giờ kết thúc với xác suất rất bé. Tuy nhiên thực nghiệm chỉ ra rằng, các thuật toán xác suất thường hữu hiệu hơn các thuật toán không xác suất. Thậm chí trong rất nhiều trường hợp, chỉ có các thuật toán như thế là sử dụng được.

Khi làm việc với các thuật toán xác suất, thường hay phải sử dụng các số “*ngẫu nhiên*”. Khái niệm chọn số ngẫu nhiên cũng cần được chính xác hóa. Thường thì người ta sử dụng một “*máy*” sản xuất số giả ngẫu nhiên nào đó. Tuy nhiên ở đây khi nói đến việc chọn số ngẫu nhiên, có thể hiểu đó là việc được thực hiện trên máy.

Cần chú ý ngay rằng, đối với các thuật toán xác suất, không thể nói đến thời gian tuyệt đối, mà chỉ có thể nói đến thời gian hy vọng (*expected*).

Để hình dung được phần nào “*độ phức tạp*” của các thuật toán khi làm việc với những số lớn, xem Bảng 1.1 dưới đây cho khoảng thời gian cần thiết để phân tích một số nguyên  $n$  ra thừa số nguyên tố bằng thuật toán nhanh nhất được biết hiện nay.

**Bảng 1.1. Độ phức tạp để phân tích số nguyên ra thừa số nguyên tố**

Số chữ số thập phân	Số phép tính bit	Thời gian
50	$1,4 \cdot 10^{10}$	3,9 giờ
75	$9 \cdot 10^{12}$	104 ngày
100	$2,3 \cdot 10^{15}$	74 năm
200	$1,2 \cdot 10^{23}$	$3,8 \cdot 10^9$ năm
300	$1,5 \cdot 10^{29}$	$4,9 \cdot 10^{15}$ năm
500	$1,3 \cdot 10^{39}$	$4,2 \cdot 10^{25}$ năm

Từ Bảng 1.1, thấy rằng ngay với một thuật toán dưới mũ, thời gian làm việc với các số nguyên lớn là quá lâu. Vì thế nói chung con người luôn cố gắng tìm những thuật toán đa thức.



## 1.7. CÂU HỎI VÀ BÀI TẬP

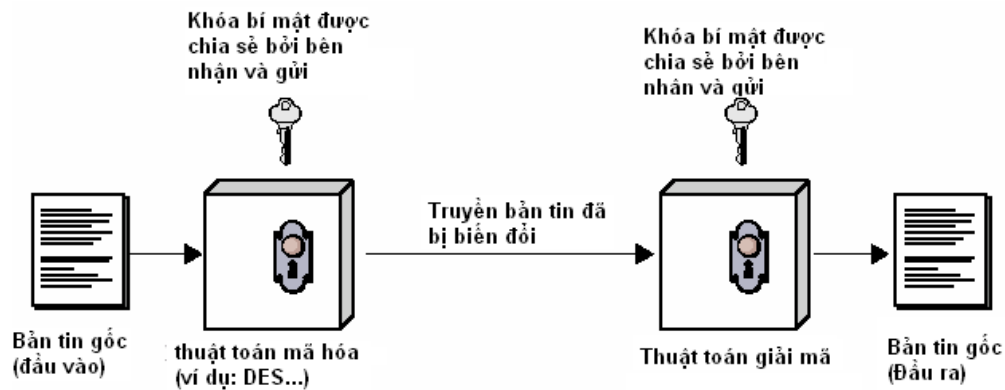
- 1) Độ an toàn thuật toán là gì?. Hãy trình bày một số tiêu chí cơ bản để đánh giá độ an toàn của một thuật toán?.
- 2) Hãy trình bày về các phương pháp phân loại mã hóa?.
- 3) Hãy nêu vai trò của mật mã học?.
- 4) Trình bày ưu nhược điểm của giao thức Kerberos?.
- 5) Hãy trình bày khái quát về Krypto Knight?.
- 6) Hãy trình bày về các mô hình hoạt động của PGP?.
- 7) Thẻ thông minh được chia thành mấy loại?. Hãy nêu chức năng và vai trò của thẻ thông minh?.
- 8) Hãy nêu định nghĩa về độ an toàn tính toán của hệ mật trong lý thuyết xác suất?.
- 9) Hãy nêu định nghĩa về độ phức tạp tính toán?.

## CHƯƠNG 2: MÃ HÓA KHÓA ĐỐI XỨNG

Chương 2 cung cấp các kiến thức về hệ mật khóa đối xứng cũng như nguyên tắc mã hóa, giải mã của một số thuật toán mã hóa khóa đối xứng. Ngoài ra, chương 2 trình bày một số ưu và nhược điểm của các kỹ thuật mã hóa khóa đối xứng.

### 2.1. GIỚI THIỆU VỀ MÃ HÓA KHÓA ĐỐI XỨNG

Trong phần giới thiệu về mật mã học ở chương 1, bài giảng đã giới thiệu về một số khái niệm trong mật mã. Trong phần này, bài giảng sẽ giới thiệu về đặc điểm của mã hóa khóa đối xứng. Hình 2.1 mô tả quy trình mã hóa và giải mã của mã hóa khóa đối xứng.



Hình 2. 1: Mô hình mã hóa đối xứng

Một hệ mật là một bộ 5  $(P, C, K, E, D)$  thỏa mãn các điều kiện sau:

- $P$  là một tập hữu hạn các bản rõ có thể.
- $C$  là một tập hữu hạn các bản mã có thể.
- $K$  là một tập hữu hạn các khóa có thể (không gian khóa).
- Đối mỗi  $k \in K$  có một quy tắc mã  $e_k \in E$

$$e_k : P \rightarrow C$$

và quy tắc giải mã tương ứng  $d_k \in D$

$$d_k : C \rightarrow P$$

sao cho  $d_k(e_k(x)) = x$  với  $\forall x \in P$ .

Lịch sử ra đời mã hóa cách đây hàng ngàn năm. Trong quá trình phát triển thì các kỹ thuật mã hóa trước kia đều sử dụng 1 khóa cho cả quá trình mã hóa và giải mã. Chính vì vậy, trong mã hóa khóa đối xứng chia thành 2 giai đoạn phát triển chính đó là: các kỹ thuật mã hóa cổ điển và các kỹ thuật mã hóa hiện đại. Phần tiếp theo, bài giảng trình bày một số thuật toán thuộc cả 2 giai đoạn phát triển này.

## 2.2. CÁC KỸ THUẬT MÃ HÓA ĐỐI XỨNG CỔ ĐIỂN

### 2.2.1. Phương pháp thay thế

Mã hóa thay thế là phương pháp thay thế mỗi ký tự thông điệp thành ký tự bản mã. Từ đó, người nhận có thể thay thế ngược lại những ký tự của bản mã để được thông điệp.

Trong mật mã học cổ điển, có một số phương pháp mã hóa thay thế :

- Mã hóa thay thế đơn bảng.
- Mã hóa thay thế đa ký tự.
- Mã hóa thay thế đa bảng.

#### 2.2.1.1. Phương pháp mã hóa thay thế đơn bảng

Trước khi tìm hiểu phương pháp thay thế đơn bảng, hãy đi tìm hiểu mã hóa *Caesar* (*Caesar Cipher*).

##### a. Phương pháp Caesar

Thế kỷ thứ 3 trước công nguyên, nhà quân sự người La Mã *Julius Caesar* đã nghĩ ra phương pháp mã hóa một thông điệp như sau: thay thế mỗi chữ trong thông điệp bằng chữ đứng sau nó  $k$  vị trí trong bảng chữ cái. Giả sử chọn  $k = 3$ , có bảng chuyển đổi như sau:

Chữ ban đầu	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Chữ thay thế	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

**Chú ý:** sau Z sẽ vòng lại là A, do đó  $x \rightarrow A$ ,  $y \rightarrow B$  và  $z \rightarrow C$

Giả sử có *thông điệp* gốc: meet me after the toga party

Như vậy *bản mã* sẽ là: PHHW PH DIWHU WKH WRJD SDUWB

Thay vì gửi trực tiếp thông điệp cho các cấp dưới, Ceasar gửi bản mã. Khi cấp dưới nhận được bản mã, tiến hành giải mã theo quy trình ngược lại để có được thông điệp. Như vậy nếu đối thủ của *Ceasar* có lấy được bản mã, thì cũng không hiểu được ý nghĩa của bản mã.

Gán cho mỗi chữ cái theo thứ tự từ A đến Z một con số nguyên theo thứ tự từ 0 đến 25. Phương pháp *Ceasar* được biểu diễn như sau: với mỗi chữ cái  $p$  thay bằng chữ mã hóa  $C$ , trong đó:

$$C = (p + k) \bmod 26 \text{ (trong đó } \bmod \text{ là phép chia lấy số dư)}$$

Và quá trình giải mã đơn giản là:  $p = (C - k) \bmod 26$ .

Trong đó  $k$  được gọi là khóa. Dĩ nhiên là *Ceasar* và cấp dưới phải cùng dùng chung một giá trị khóa  $k$ , nếu không thông điệp giải mã sẽ không giống thông điệp ban đầu.

Ngày nay phương pháp mã hóa của Ceasar không được xem là an toàn. Giả sử đối thủ của **Ceasar** có được bản mã **PHHW PH DIWHU WKH WRJD SDUWB** và biết được phương pháp mã hóa và giải mã là phép cộng trừ *modulo* 26. “*Kẻ tấn công*” có thể thử tất cả 25 trường hợp của  $k$  như bảng 2.1:

**Bảng 2. 1: Bảng thống kê kết quả tấn công vét cạn**

KEY	PHHW	PH	DIWHU	WKH	WRJD	SDUWB
1	oggv	og	chvgt	vjg	vqic	rctva
2	nffu	nf	bgufs	uif	uphb	qbsuz
<b>3</b>	<b>meet</b>	<b>me</b>	<b>after</b>	<b>the</b>	<b>toga</b>	<b>party</b>
4	ldds	ld	zesdq	sgd	snfz	ozqsx
5	kccr	kc	ydrp	rfc	rmey	nyprw
6	jbbq	jb	xcqbo	qeb	qldx	mxoqv
7	iaap	ia	wbpan	pda	pkcw	lwnpu
8	hzzo	hz	vaozm	ocz	objv	kvmot

9	gyyn	gy	uznyl	nby	niau	julns
10	fxxm	fx	tymxk	max	mhzt	itkmr
11	ewwl	ew	sxlwj	lzw	lgys	hsjlq
12	dvvk	dv	rwkvi	kyv	kfxr	grikp
13	cuuj	cu	qvjuh	jxu	jewq	fqhjo
14	btti	bt	puitg	iwt	idvp	epgin
15	assh	as	othsf	hvs	hcuo	dofhm
16	zrrg	zr	nsGRE	gur	gbtn	cnegl
17	yqqf	yq	mrfqd	ftq	fasm	bmdfk
18	xppe	xp	lqepc	esp	ezrl	alcej
19	wood	wo	kpdob	dro	dyqk	zkbdi
20	vnnc	vn	jocna	cqn	cxpj	yjach
21	ummb	um	inbmz	bpm	bwoi	xizbg
22	tlla	tl	hmaly	aol	avnh	whyaf
23	skkz	sk	glzkx	znk	zumg	vgxze
24	rjyy	rj	fkyjw	ymj	ytlf	ufwyd
25	qiix	qi	ejxiv	xli	xske	tevxc

Trong 25 trường hợp trên, chỉ có trường hợp  $k = 3$  thì bản giải mã tương ứng là có ý nghĩa. Do đó “*Kẻ tấn công*” có thể chắc chắn rằng “*meet me after the toga party*” là thông điệp ban đầu.

## b. Phương pháp thay thế đơn bằng (Monoalphabetic Substitution Cipher)

Xét lại phương pháp *Caesar* với  $k = 3$ :

Chữ ban đầu	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Chữ thay thế	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Phương pháp đơn bằng tổng quát hóa phương pháp Caesar bằng cách dòng mã hóa không phải là một dịch chuyển  $k$  vị trí của các chữ cái A, B, C, ... nữa mà là một hoán vị

của 26 chữ cái này. Lúc này mỗi hoán vị được xem như là một khóa. Giả sử có hoán vị sau:

Chữ ban đầu	a	b	c	d	e	f	g	h	i	j	K	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Khóa	Z	P	B	Y	J	R	S	K	F	L	X	Q	N	W	V	D	H	M	G	U	T	O	I	A	E	C

Như vậy thông điệp **meet me after the toga party**

được mã hóa thành: **NJJU NJ ZRUJM UKJ UVSZ DZMUE**

Quá trình giải mã được tiến hành ngược lại để cho ra thông điệp ban đầu.

Việc mã hóa được tiến hành bằng cách thay thế một chữ cái trong thông điệp thành một chữ cái trong bản mã, nên phương pháp này được gọi là phương pháp thay thế. Số lượng hoán vị của 26 chữ cái là  $26!$ , đây cũng chính là số lượng khóa của phương pháp này. Vì  $26!$  là một con số khá lớn nên việc tấn công phá mã vét cạn khóa là bất khả thi (6400 thiên niên kỷ với tốc độ thử khóa là  $109$  khóa/giây). Vì vậy mã hóa đơn bảng đã được xem là một phương pháp mã hóa an toàn trong suốt 1000 năm sau công nguyên.

### 2.2.1.2. Phương pháp thay thế đa ký tự

#### a. Mã hóa Playfair

**Bảng 2. 2: Ma trận mã hóa Playfair**

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

Mã hóa *Playfair* sử dụng đơn vị mã hóa là 2 ký tự đứng sát nhau. Như vậy hai ký tự này được thay thế cùng một lúc bởi hai ký tự khác. Mã hóa *Playfair* dùng một ma trận  $5 \times 5$  (*Matrix State*).

Trong bảng trên, khóa *MONARCHY* được điền vào dòng đầu tiên và 3 ký tự dòng thứ 2 của bảng. Các ô còn lại được điền các chữ cái theo thứ tự của bảng chữ cái Tiếng Anh. Riêng 2 chữ *I* và *J* được điền vào chung 1 ô vì trong Tiếng Anh hai chữ cái này ít bị nhầm lẫn. Ví dụ: khi gặp chữ *SM\_LE* thì sẽ đoán được chữ đó là *SMILE* chứ không thể nào là *SMJLE* được.

Trước khi tiến hành mã hóa, thông điệp được tách ra thành các cặp ký tự. Nếu 2 ký tự trong một cặp giống nhau thì sẽ được thay chữ cái đứng sau trong cặp đó bằng chữ *X* (vì trong Tiếng Anh thì thường 2 chữ *X* rất hiếm khi đứng cạnh nhau). Ví dụ: từ *football* sẽ được tách thành 4 cặp *fo ot ba lx*.

Quy tắc mã hóa như sau :

- Nếu 2 ký tự được mã hóa thuộc cùng 1 hàng, thì được thay thế bằng 2 ký tự tiếp theo trong hàng. Nếu đến cuối hàng thì quay về đầu hàng. Ví dụ: cặp ***LP*** thay bằng ***PQ***, cặp ***ST*** thay bằng ***TU***.
- Nếu 2 ký tự trong cặp thuộc cùng 1 cột, thì được thay bằng 2 ký tự tiếp theo trong cột. Nếu đến cuối cột thì quay về đầu cột. Ví dụ cặp *OV* được mã hóa thành *HO*.
- Trong các trường hợp còn lại, 2 ký tự được mã hóa sẽ tạo thành đường chéo của 1 hình chữ nhật và được thay bằng 2 ký tự trên đường chéo kia. Ví dụ: *HS* trở thành *BP* (*B* cùng dòng với *H* và *P* cùng dòng với *S*); *EA* trở thành *IM* (hoặc *JM*).

Như vậy nếu chỉ xét trên 26 chữ cái thì mã hóa *Playfair* có  $26 \times 26 = 676$  cặp chữ cái, do đó các cặp chữ cái này ít bị chênh lệch về tần suất hơn so với sự chênh lệch tần suất của từng chữ cái. Ngoài ra số lượng các cặp chữ cái nhiều hơn cũng làm cho việc phá mã tần suất khó khăn hơn. Đây chính là lý do mà người ta tin rằng mã hóa *Playfair* không thể bị phá và được quân đội Anh sử dụng trong chiến tranh thế giới lần thứ nhất.

## b. Mã hóa Hill

Trong mã hóa Hill, mỗi chữ cái (A – Z) được gán tương ứng với 1 số nguyên từ 0 đến 25. Đơn vị mã hóa của mã hóa Hill là 1 khối (*block*)  $m$  ký tự thông điệp (ký hiệu  $p_1p_2...p_m$ ) thành 1 khối (*block*)  $m$  ký tự bản mã (ký hiệu  $c_1c_2...c_m$ ). Việc thay thế này được thực hiện bằng  $m$  phương trình tuyến tính. Giả sử  $m = 3$ , minh họa  $m$  phương trình đó như sau:

Dạng phương trình :

$$c_1 = k_{11}p_1 + k_{12}p_2 + k_{13}p_3 \mod 26$$

$$c_2 = k_{21}p_1 + k_{22}p_2 + k_{23}p_3 \mod 26$$

$$c_3 = k_{31}p_1 + k_{32}p_2 + k_{33}p_3 \mod 26$$

Dạng ma trận :

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \mod 26$$

Hay:  $C = KP \mod 26$  với  $P$  và  $C$  là vector đại diện cho thông điệp và bản mã, còn  $K$  là ma trận dùng làm khóa.

Xét ví dụ thông điệp là *paymoremoney* cùng với khóa  $K$  là :

$$K = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

Ba chữ cái đầu tiên của thông điệp tương ứng với vector (15, 0, 24). Vậy

$$\begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \begin{bmatrix} 15 \\ 0 \\ 24 \end{bmatrix} \mod 26 = \begin{bmatrix} 11 \\ 23 \\ 18 \end{bmatrix} = LNS$$

Thực hiện tương tự có bản mã đầy đủ là *LNSHDLEWMTRW*.



Để giải mã cần sử dụng một ma trận nghịch đảo của  $K$  là  $K^{-1}$ , tức là  $K^{-1}K \bmod 26 = 1$  là ma trận đơn vị (không phải mọi ma trận  $K$  đều tồn tại ma trận nghịch đảo, tuy nhiên nếu tồn tại thì có thể tìm được ma trận đơn vị bằng cách tính hạng *det* của ma trận).

Ví dụ ma trận nghịch đảo của ma trận trên là :

$$K^{-1} = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix}$$

Vì:

$$\begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix} \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} = \begin{bmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{bmatrix} \bmod 26 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Khi đó bảng giải mã là  $K^{-1}C \bmod 26 = K^{-1}KP \bmod 26 = P$ .

Có thể thấy mã hóa *Hill* ẩn giấu các thông tin về tần suất nhiều hơn mã hóa Playfair do có thể mã hóa 3 hoặc nhiều hơn nữa các ký tự cùng lúc.

### 2.2.1.3. Phương pháp mã hóa thay thế đa bảng

Với sự phát hiện ra quy luật phân bố tần suất, các nhà phá mã đang tạm thời chiếm ưu thế trong cuộc chiến mã hóa-thám mã. Cho đến thế kỷ thứ 15, một nhà ngoại giao người Pháp tên là Vigenere đã tìm ra phương án mã hóa thay thế đa bảng.

Trong Vigenere Cipher, người ta dùng tất cả 26 bảng thế thu được từ bảng gốc chữ cái tiếng Anh mà dịch đi từ 0-25 vị trí. Sự hoà trộn này có quy luật hoàn toàn xác định bởi khoá. Mỗi chữ của khoá sẽ xác định mỗi bảng thế được dùng. Bảng 2.3 (bảng thế A – dịch đi 0 vị trí từ bảng gốc chữ cái tiếng Anh) là một trong 26 bảng được sử dụng trong phương pháp Vigenere.

**Bảng 2. 3. Bảng thế A**

KEY	a b c d e f g h i j k l m n o p q r s t u v w x y z
a	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
b	B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
c	C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
d	D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
e	E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
f	F G H I J K L M N O P Q R S T U V W X Y Z A B C D E
g	G H I J K L M N O P Q R S T U V W X Y Z A B C D E F
h	H I J K L M N O P Q R S T U V W X Y Z A B C D E F G
i	I J K L M N O P Q R S T U V W X Y Z A B C D E F G H
j	J K L M N O P Q R S T U V W X Y Z A B C D E F G H I
k	K L M N O P Q R S T U V W X Y Z A B C D E F G H I J
l	L M N O P Q R S T U V W X Y Z A B C D E F G H I J K
m	M N O P Q R S T U V W X Y Z A B C D E F G H I J K L
n	N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
o	O P Q R S T U V W X Y Z A B C D E F G H I J K L M N
p	P Q R S T U V W X Y Z A B C D E F G H I J K L M N O
q	Q R S T U V W X Y Z A B C D E F G H I J K L M N O P
r	R S T U V W X Y Z A B C D E F G H I J K L M N O P Q
s	S T U V W X Y Z A B C D E F G H I J K L M N O P Q R
t	T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
u	U V W X Y Z A B C D E F G H I J K L M N O P Q R S T
v	V W X Y Z A B C D E F G H I J K L M N O P Q R S T U
w	W X Y Z A B C D E F G H I J K L M N O P Q R S T U V
x	X Y Z A B C D E F G H I J K L M N O P Q R S T U V W
y	Y Z A B C D E F G H I J K L M N O P Q R S T U V W X
z	Z A B C D E F G H I J K L M N O P Q R S T U V W X Y

Dòng thứ  $t$  chính là một mã hóa *Cesar*  $t-1$  vị trí. Ví dụ: Ở dòng thứ 9 ứng với từ khóa  $I$  là một mã hóa *Cesar* 8 vị trí. Nếu tổng quát hóa, mỗi dòng của mã hóa đa bảng chính là một mã hóa đơn bảng.

Ví dụ:

Khóa	r	a	d	i	o	r	a	d	i	o	r	a
Bản rõ	c	o	d	e	b	r	e	a	k	i	n	g
Bản mã	t	o	g	m	p	i	e	d	s	w	e	g

Theo ví dụ trên, khóa có độ dài là 5 (radio), nên các ký tự trong bản rõ sẽ được mã hóa theo modulo của 5. Cụ thể, các ký tự có vị trí mod 5 bằng 0 (c, r, n) sẽ mã hóa bởi bảng thế R (A thành R), bằng 1 (o, e, g) sẽ mã hóa bởi bảng thế A, bằng 2 (d, a) sẽ mã hóa bởi bảng thế D, bằng 3 (e, k) sẽ mã hóa bởi bảng thế I và bằng 4 (b, i) sẽ mã hóa bởi bảng thế O.

#### 2.2.1.4. One Time Pad

Để khắc phục điểm yếu của việc lặp lại khóa trong mã hóa thay thế đa bảng, bỏ sự liên quan giữa thông điệp và bản mã, *Joseph Mauborgne*, giám đốc viện nghiên cứu mật mã của quân đội Mỹ, vào cuối cuộc chiến tranh thế giới lần thứ nhất đã đề xuất phương án dùng khóa ngẫu nhiên. Khóa ngẫu nhiên có chiều dài bằng chiều dài của thông điệp, mỗi khóa chỉ sử dụng một lần.

Ví dụ mã hóa:

Thông điệp *P*: **wearediscoveredsaveyourself**

Khóa *K1*: **FHWYKLV MKVKXCVDJSFSAPXZCVP**

Bản mã *C*: **BLWPOODEMJFBTZN VJNJQOJORG GU**

Nếu dùng khóa *K1* để giải mã thì sẽ được lại thông điệp:

**“wearediscoveredsaveyourself”**

Tuy nhiên xét hai trường hợp giải mã bản mã trên với 2 khóa khác như sau:

Trường hợp 1: Bản mã *C*: **BLWPOODEMJFBTZN VJNJQOJORG GU**

Khóa *K2*: **IESRLKBWJFCIFZUCJLZXAXAAPSY**

Bản giải mã: **theydecidedtoattacktomorrow**

*(they decided to attack tomorrow)*

Trường hợp 2:      Bản mã C: **BLWPOODEMJFBTZNJVJNJQOJORGGU**

Khóa K3: **FHAHDDRAIQFIASJGJWQSVVBIAZB**

Bản giải mã: **wewillmeetatthepartytonight**

*(we will meet at the party tonight)*

Trong cả hai trường hợp trên thì bản giải mã đều có ý nghĩa. Điều này có nghĩa là nếu người phá mã thực hiện phá mã vét cạn thì sẽ tìm được nhiều khóa ứng với nhiều bản 24 tin có ý nghĩa, do đó sẽ không biết được thông điệp nào là thông điệp chính. Điều này chứng minh phương pháp OTP là phương pháp mã hóa an toàn tuyệt đối, và được xem là “*chén thánh*” của khóa mật mã cổ điển. Một điều cần chú ý là để phương pháp OTP là an toàn tuyệt đối thì mỗi khóa chỉ được sử dụng một lần. Nếu một khóa được sử dụng nhiều lần thì cũng không khác gì việc lặp lại một từ trong khóa (ví dụ khóa có từ *DECEPTIVE* được lặp lại). Ngoài ra các khóa phải thật sự ngẫu nhiên với nhau. Nếu các điều này bị vi phạm thì sẽ có một mối liên hệ giữa thông điệp và bản mã, và người phá mã sẽ tận dụng mối quan hệ này.

Tuy nhiên, phương pháp *OTP* không có ý nghĩa sử dụng thực tế. Vì chiều dài khóa bằng chiều dài thông điệp, mỗi khóa chỉ sử dụng một lần, nên thay vì truyền khóa trên kênh an toàn thì có thể truyền trực tiếp thông điệp mà không cần quan tâm đến vấn đề mã hóa. Vì vậy sau chiến tranh thế giới thứ nhất, người ta vẫn chưa thể tìm ra loại mật mã nào khác mà không bị phá mã. Mọi cố gắng vẫn là tìm cách thực hiện một mã thay thế đa bảng dùng một khóa dài, ít lặp lại, để hạn chế phá mã. Máy *ENIGMA* được quân đội Đức sử dụng trong chiến tranh thế giới lần thứ 2 là một máy như vậy. Sử dụng máy *ENIGMA*, Đức đã chiếm ưu thế trong giai đoạn đầu của cuộc chiến. Tuy nhiên trong giai đoạn sau, các nhà phá mã người Ba Lan và Anh (trong đó có *Alan Turing*, người phá minh ra máy tính có thể lập trình được) đã tìm ra cách phá mã máy *ENIGMA*. Việc phá mã thực hiện được dựa vào một số điểm yếu trong khâu phân phối khóa của quân Đức. Điều này đóng vai trò quan trọng vào chiến thắng của quân đồng minh trong cuộc chiến.

### 2.2.2. Phương pháp mã hóa hoán vị

Trong mã hóa hoán vị (đổi chỗ), các ký tự trong thông điệp được sắp xếp, đổi chỗ lại để tạo thành bản mã. Do thứ tự các ký tự đã bị thay đổi nên bản mã sẽ trở thành thông điệp vô nghĩa, không thể đọc được thông tin chứa trong nó.

Một ví dụ đơn giản, dịch vị trí các ký tự thông điệp sang phải 2 đơn vị. Khi đó thông điệp “hello world” sẽ thành “ldhello wor” hoàn toàn vô nghĩa.

Có thể áp dụng cho từng khối  $n$  ký tự và thực hiện hoán vị theo một khóa  $K$  định sẵn.

Ví dụ thực hiện đổi chỗ khối 8 ký tự :

Cho khóa  $K$  như sau :  $1 \rightarrow 4, 2 \rightarrow 8, 3 \rightarrow 1, 4 \rightarrow 5, 5 \rightarrow 7, 6 \rightarrow 2, 7 \rightarrow 6, 8 \rightarrow 3$

Thông điệp(*Plaintext*): SACKGAUL | SPARENOO | NE |

Bản mã(*Ciphertext*) : UKAGLSCA | ORPEOSAN | E N |

Số thứ tự : 8 6 7 5 4 3 2 1 | 8 6 7 5 4 3 2 1 | 8 6 7 5 4 3 2 1 |

### 2.2.3. Phương pháp mã hóa XOR

Phương pháp mã hóa *XOR* sử dụng phép toán logic *XOR* để tạo ra bản mã. Từng bit của thông điệp được *XOR* tương ứng với từng bit của khóa để cho ra bản mã.

**Bảng 2. 4. Bảng chân trị phép toán XOR**

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Ví dụ: Mã hóa từ LOVE với khóa  $K$  là từ HATE sử dụng phương pháp *XOR*.

Chuyển các từ và khóa về dạng nhị phân như sau:

<b>Plainttext</b>	L	O	V	E
<b>Plaintext_bin</b>	0100 1100	0100 1111	0101 0110	0100 0101
<b>Key</b>	H	A	T	E
<b>Key_bin</b>	0100 1000	0100 0001	0101 0100	0100 0101
<b>Ciphertext_bin</b>	0000 0100	0000	1110	0000 0010 0000 0000

#### 2.2.4. Phương pháp sách hoặc khóa chạy

Trong các tiểu thuyết trinh thám, phim trinh thám thường bắt gặp phương pháp này trong việc mã hóa và giải mã sử dụng khóa chứa trong các trang sách hoặc chứa trong các dòng của một thông điệp .v.v.v..

Ví dụ: với bản mã là 259, 19, 8; 22, 3, 8; 375, 7, 4; 394, 17, 2 và cuốn sách được dùng là "A Fire Up on the Deep":

- Trang 259, dòng 19, từ thứ 8  $\rightarrow$  *sack*
- Trang 22, dòng 3, từ thứ 8  $\rightarrow$  *island*
- Trang 375, dòng 7, từ thứ 4  $\rightarrow$  *sharp*
- Trang 394, dòng 17, từ thứ 2  $\rightarrow$  *path*
- Thông điệp tương ứng của bản mã "259,19,8;22,3,8;375,7,4;394,17,2 " là "sack island sharp path".

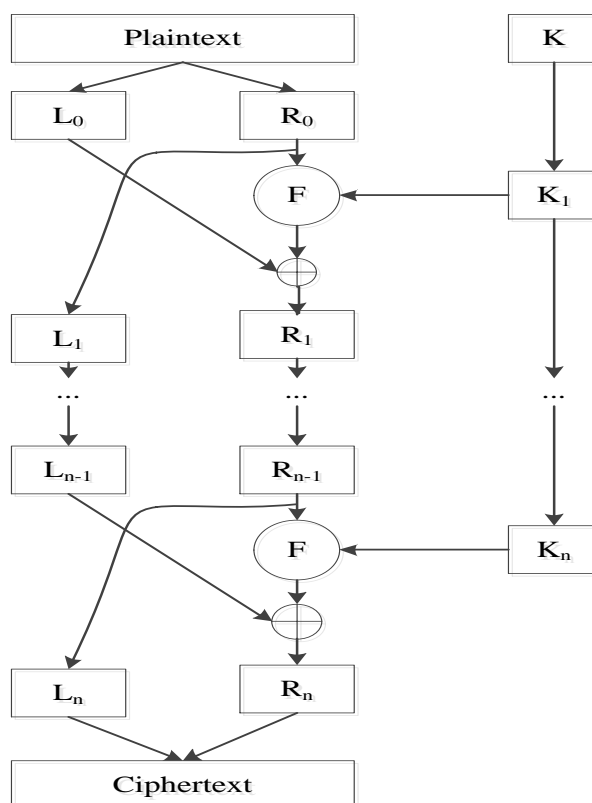
### 2.3. CÁC KỸ THUẬT MÃ HÓA ĐỐI XỨNG HIỆN ĐẠI

Như trình bày ở trên, việc phân loại mã hóa có thể phân loại theo đặc trưng xử lý thông điệp. Trong phần mã hóa khóa đối xứng hiện đại, bài giảng sẽ tập trung vào việc tìm hiểu các kỹ thuật mã hóa đối xứng thuộc mã hóa khối và mã hóa dòng. Trong kỹ thuật mã hóa đối xứng thuộc mã khối, sẽ trình bày các thuật toán: *DES*, *3DES*; *AES*. Trong kỹ thuật mã hóa đối xứng thuộc mã dòng, sẽ trình bày các thuật toán: *RC4* và *A5/I*.

#### 2.3.1. Thuật toán mã hóa DES/3DES

##### 2.3.1.1. Giới thiệu về mô hình Feistel

Hình 2.2 mô tả cấu trúc đề xuất bởi Feistel. Đầu vào của thuật toán mã hóa là khối văn bản gốc có chiều dài  $2w$  bit và một khóa  $K$ . Khối thông điệp gốc được chia thành hai nửa,  $L_0$  và  $R_0$ . Hai nửa này đi qua  $n$  vòng xử lý rồi tổ hợp để tạo ra khối bản mã hóa. Mỗi vòng  $i$  có đầu vào  $L_{i-1}$  và  $R_{i-1}$ , lấy từ vòng trước đó, giống như 1 khóa con  $K_i$ , lấy từ tổng thể  $K$ . Nói chung, những khóa con  $K_i$  khác với khóa  $K$  và các khóa khác.



Hình 2. 2. Mô hình Feistel

Tất cả các vòng có cấu trúc tương tự nhau. Một thay thế được thực hiện trên một nửa trái của dữ liệu. Điều này được thực hiện bằng cách áp dụng một hàm vòng  $F$  cho nửa bên phải của dữ liệu và sau đó lấy *exclusive-OR* của các đầu ra của hàm đó và một nửa còn lại của dữ liệu. Tất cả các vòng có cùng một cấu trúc chung cho mỗi vòng, nhưng là tham số của mỗi vòng là khóa con  $K_i$ . Sau sự thay thế này, một hoán vị được thực hiện mà bao gồm việc trao đổi hai nửa của dữ liệu. Việc thực hiện chính xác của một mạng *Feistel* phụ thuộc vào sự lựa chọn của các tham số và đặc điểm thiết kế sau:

- Kích cỡ khối: khối kích thước lớn hơn có nghĩa là bảo mật cao hơn (tất cả những thứ khác bằng nhau), nhưng giảm đi tốc độ mã hóa/giải mã của một thuật toán cho trước. Theo truyền thống, một kích thước khối 64 bit đã được coi là một sự cân bằng hợp lý và đã được phổ cập trong thiết kế gần mã hóa khối.
- Kích thước khóa: kích thước lớn hơn có nghĩa là an toàn hơn, nhưng có thể làm giảm tốc độ mã hóa/giải mã. Việc bảo mật cao hơn đạt được bằng cách chống tấn công *brute-force* (tấn công vét cạn) tốt hơn. Kích thước khóa của 64 bit hoặc ít hơn bây giờ

nhiều người xem là không đủ an toàn, và 128 bit đã trở thành một kích thước khóa thông thường.

- Số vòng: Bản chất của mã hóa *Feistel* là một vòng duy nhất cung cấp bảo mật không đầy đủ nhưng nhiều vòng cung cấp sẽ tăng cường bảo mật hơn. Một kích thước điển hình là 16 vòng.
- Thuật toán sinh khóa phụ: phức tạp lớn hơn trong thuật toán này nên dẫn đến khó khăn lớn hơn của phân tích mật mã.
- Hàm vòng: Một lần nữa, phức tạp hơn thường có nghĩa chống phân tích mật mã tốt hơn. Có hai quan tâm khác trong thiết kế của một mã hóa *Feistel*:
- Tốc độ phần mềm mã hóa/giải mã: Trong nhiều trường hợp, mã hóa được nhúng trong các ứng dụng hoặc các chức năng tiện ích. Vì vậy, tốc độ thực hiện của thuật toán sẽ trở thành một mối quan tâm.
- Dễ dàng trong việc phân tích: Mặc dù muốn làm cho thuật toán khó khăn nhất có thể để chống lại việc phân tích mã. Vì vậy, nếu thuật toán có thể được giải thích ngắn gọn và rõ ràng, nó được dễ dàng hơn trong việc phân tích thuật toán để giảm rủi ro và do đó chống lại các phân tích mã phát triển ở một mức độ cao hơn.

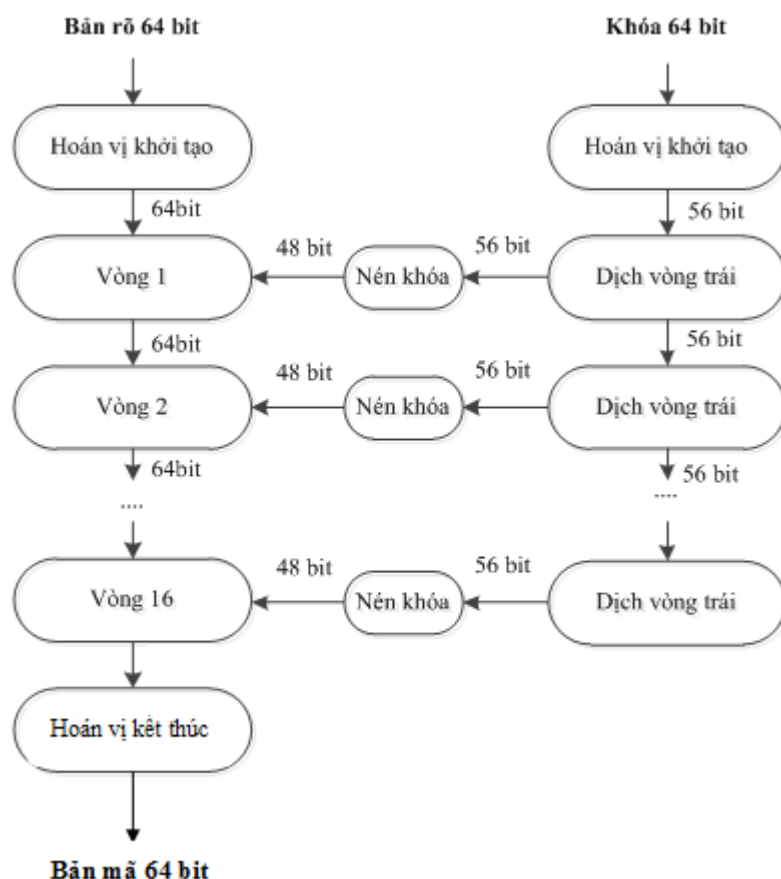
### 2.3.1.2. Thuật toán mã hóa DES

*DES* được phát triển tại *IBM* vào đầu những năm 1970 và được thừa nhận là chuẩn mã hóa tại Mỹ (*NSA*) vào năm 1976. Sau đó *DES* được sử dụng rộng rãi trong những năm 70 và 80.

Mã hóa *DES* có các tính chất sau:

- Là mã thuộc hệ mã *Feistel* gồm 16 vòng, ngoài ra *DES* có thêm một hoán vị khởi tạo trước khi vào vòng 1 và một hoán vị kết thúc sau vòng 16.
  - Kích thước của khối là 64 bit: ví dụ thông điệp “*meetmeafterthetogaparty*” biểu diễn theo mã *ASCII* thì mã *DES* sẽ mã hóa làm 3 lần, mỗi lần 8 chữ cái (64 bit): *meetmeaf*  
- *tertheto* - *gaparty*.
  - Kích thước khóa là 64 bit (thực chất chỉ có 56 bit được sử dụng để mã hóa);
  - Mỗi vòng của *DES* dùng khóa con có kích thước 48 bit được trích ra từ khóa chính.
- Hình 2.3 trình bày tổng quát về quy trình mã hóa của thuật toán mã hóa DES.





**Hình 2. 3. Sơ đồ tổng quát quá trình mã hóa DES**

Sơ đồ mã hóa *DES* gồm 3 phần, phần thứ nhất là các hoán vị khởi tạo và hoán vị kết thúc. Phần thứ hai là các vòng Feistel, phần thứ ba là thuật toán sinh khóa con. Sau đây sẽ trình bày lần lượt tìm hiểu chi tiết từng phần.

#### **a) Hoán vị khởi tạo và hoán vị kết thúc**

Giá trị đầu vào là khối  $M$  có kích thước 64 bit  $m_1, m_2, m_3, \dots, m_{64}$ . Hoán vị khởi tạo sẽ thực hiện biến đổi giá trị các bit theo nguyên tắc sau:  $(m_1, m_2, m_3, \dots, m_{64} \rightarrow m_{58}, m_{50}, m_{42}, \dots, m_7)$ . Bảng 2.5 thể hiện bảng hoán vị khởi tạo của DES

**Bảng 2. 5. Bảng hoán vị khởi tạo DES**

IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Hoán vị kết thúc hoán đổi các bit theo quy tắc sau:  $(c_1, c_2, c_3, \dots, c_{64} \rightarrow c_{40}, c_8, c_{48}, \dots, c_{25})$

**Bảng 2. 6. Bảng hoán vị kết thúc DES**

$IP^{-1}$							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

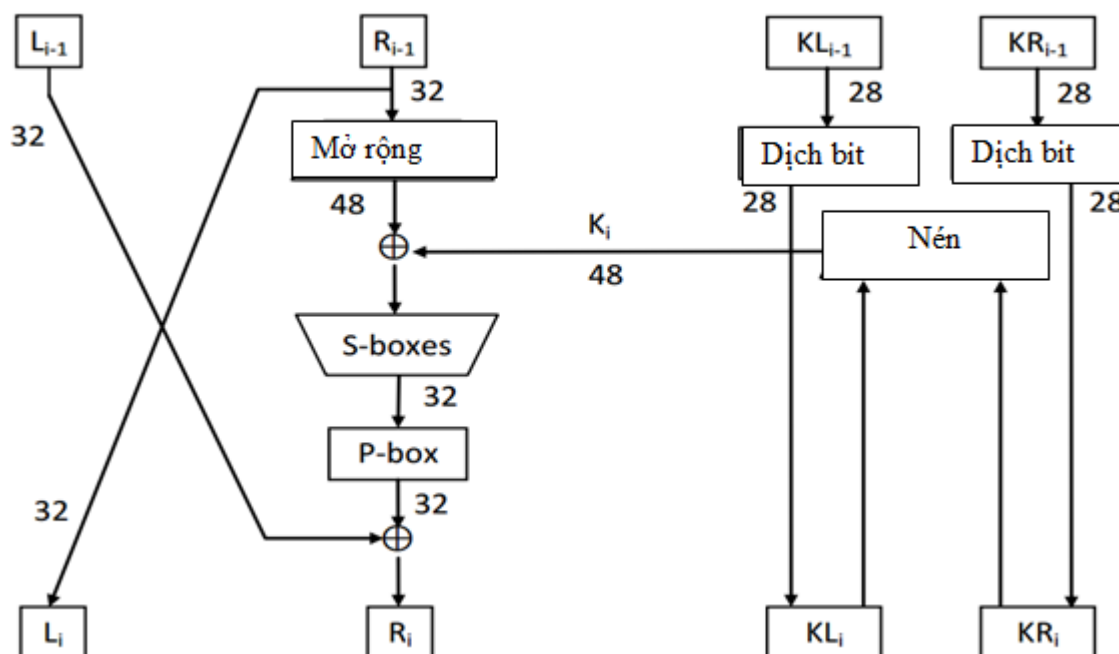
Ví dụ: Cho giá trị đầu vào thông điệp dưới dạng thập lục phân (*hexa*) như sau :  
 $0x0002\ 0000\ 0000\ 0001_{16}$ .

→ Phân tích thấy chỉ có bit 15 và bit 64 mang giá trị sử dụng bảng *IP* thấy giá trị tại bit 15 chuyển thành bit 63, bit 64 trở thành bit 25 trong output.

→ Sau hoán vị khởi tạo, giá trị sẽ là  $0x0000\ 0080\ 0000\ 0002$ .

Hoán vị kết thúc chính là hoán vị nghịch đảo của hoán vị khởi tạo. Đối với *knownplaintext* hay *chosen-plaintext attack*, hoán vị khởi tạo và hoán vị kết thúc không có ý nghĩa bảo mật, sự tồn tại của hai hoán vị trên được nhận định là do yếu tố lịch sử.

b) Các vòng của DES



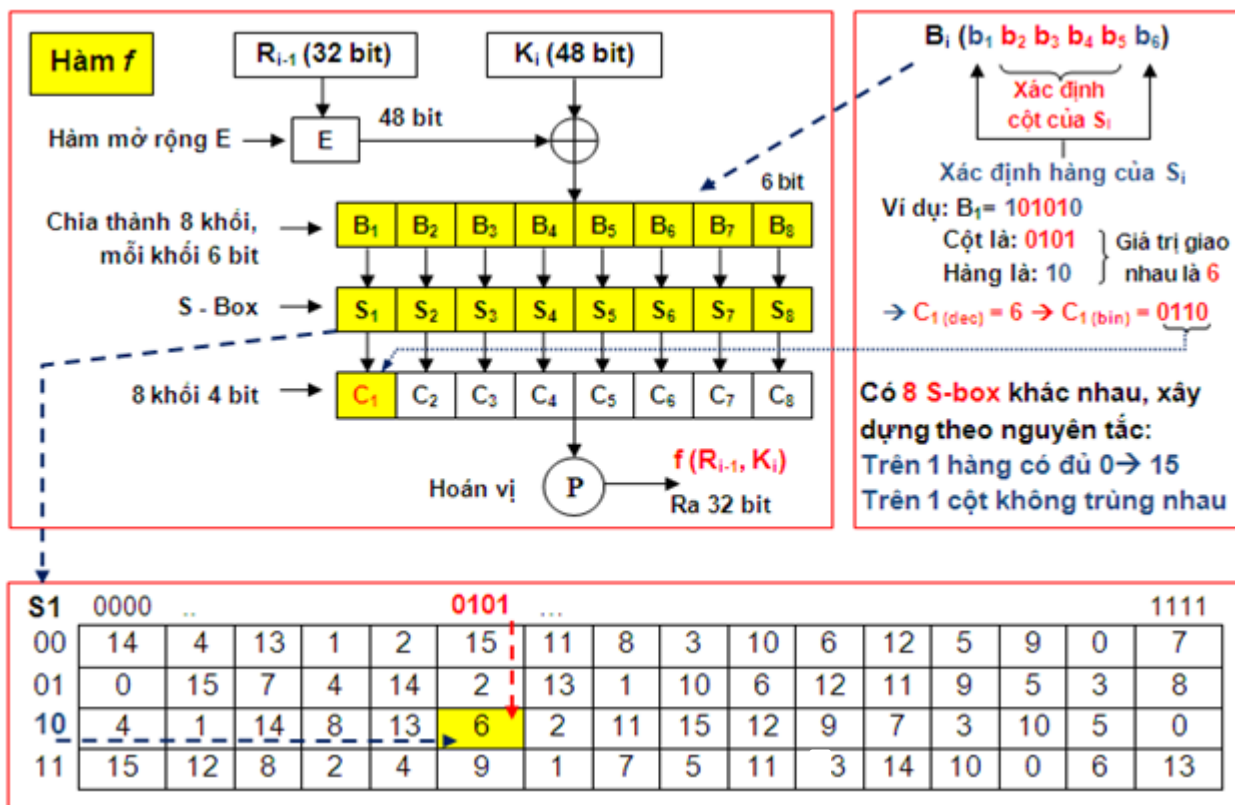
Hình 2. 4. Mô hình vòng lặp DES

Trong DES biểu thức hàm  $F$  là :

$$F(R_{i-1}, K_i) = P\text{-box}(s\text{-boxes}(\text{Expand}(R_{i-1}) \oplus K_i))$$

Trong đó hàm *Expand* (*Mở rộng*) vừa mở rộng vừa mở rộng  $R_{i-1}$  từ 32 bit lên 48 bit. Hàm *S-boxes* lại nén 48bit xuống còn 32 bit. Hàm *P-box* là một hoán vị 32bit.

Hình 2.5 mô tả 1 hàm  $f$  trong 1 vòng lặp của thuật toán DES.



Hình 2. 5. Mô tả hàm  $F$  trong DES

Mô tả của các hàm trên như sau:

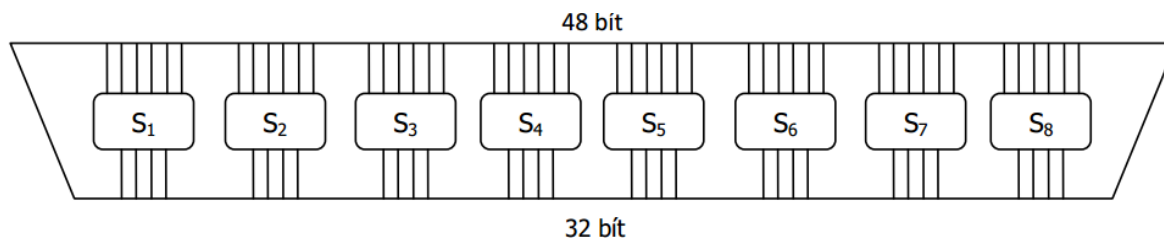
- Hàm *Expand*: hàm thực hiện hoán vị và mở rộng từ 32 bit lên 48 bit theo nguyên tắc sau :  $\underbrace{r_1, r_2, r_3, \dots, r_{32}}_{32bit} \rightarrow \underbrace{r_{32}, r_1, r_2, \dots, r_1}_{48bit}$

Bảng 2. 7. Bảng *Expand* DES

$E$					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

48 bit

- Hàm *S-boxes*: Hàm *S-boxes* của *DES* biến đổi một số 48 bit thành một số 32 bit. Hàm *S-boxes* được chia thành 8 hàm *S-box* con, mỗi hàm biến đổi số 6 bit thành số 4 bit.



Hàm *S-box* đầu tiên,  $S_1$  có quy tắc như bảng sau:

**Bảng 2. 8. Bảng *S-box1***

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
$b_0b_5$	00	1110 0100 1101 0001 0010 1111 1011 1000 0011 1010 0110 1100 0101 1001 0000 0111
	01	0000 1111 0111 0100 1110 0010 1101 0001 1010 0110 1100 1011 1001 0101 0011 1000
	10	0100 0001 1110 1000 1101 0110 0010 1011 1111 1100 1001 0111 0011 1010 0101 0000
	11	1111 1100 1000 0010 0100 1001 0001 0111 0101 1011 0011 1110 1010 0000 0110 1101

Ví dụ :  $\underbrace{000000}_{b_0b_1b_2b_3b_4b_5} \rightarrow 1110$  ,  $\underbrace{100001}_{b_0b_1b_2b_3b_4b_5} \rightarrow 1111$   
 $\text{row}[1]\text{col}[1]$        $\text{row}[4]\text{col}[1]$

Tương tự các bảng  $S_2 \rightarrow S_8$  sẽ biến đổi theo nguyên tắc của các bảng sau :

➤ Bảng  $S_2$

**Bảng 2. 9. Bảng *S-box2***

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
$b_0b_5$	00	1111 0001 1000 1110 0110 1011 0011 0100 1001 0111 0010 1101 1100 0000 0101 1010
	01	0011 1101 0100 0111 1111 0010 1000 1110 1100 0000 0001 1010 0110 1001 1011 0101
	10	0000 1110 0111 1011 1010 0100 1101 0001 0101 1000 1100 0110 1001 0011 0010 1111
	11	1101 0 1010 0001 0011 1111 0100 0010 1011 0110 0111 1100 0000 0101 1110 1001

➤ Bảng  $S_3$

**Bảng 2. 10. Bảng  $S\text{-}box3$**

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
$b_0b_5$	00	1010 0000 1001 1110 0110 0011 1111 0101 0001 1101 1100 0111 1011 0100 0010 1000
	01	1101 0111 0000 1001 0011 0100 0110 1010 0010 1000 0101 1110 1100 1011 1111 0001
	10	1101 0110 0100 1001 1000 1111 0011 0000 1011 0001 0010 1100 0101 1010 1110 0111
	11	0001 1010 1101 0000 0110 1001 1000 0111 0100 1111 1110 0011 1011 0101 0010 1100

➤ Bảng  $S_4$

**Bảng 2. 11. Bảng  $S\text{-}box4$**

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
$b_0b_5$	00	0111 1101 1110 0011 0000 0110 1001 1010 0001 0010 1000 0101 1011 1100 0100 1111
	01	1101 1000 1011 0101 0110 1111 0000 0011 0100 0111 0010 1100 0001 1010 1110 1001
	10	1010 0110 1001 0000 1100 1011 0111 1101 1111 0001 0011 1110 0101 0010 1000 0100
	11	0011 1111 0000 0110 1010 0001 1101 1000 1001 0100 0101 1011 1100 0111 0010 1110

➤ Bảng  $S_5$

**Bảng 2. 12. Bảng  $S\text{-}box5$**

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
$b_0b_5$	00	0010 1100 0100 0001 0111 1010 1011 0110 1000 0101 0011 1111 1101 0000 1110 1001
	01	1110 1011 0010 1100 0100 0111 1101 0001 0101 0000 1111 1010 0011 1001 1000 0110
	10	0100 0010 0001 1011 1010 1101 0111 1000 1111 1001 1100 0101 0110 0011 0000 1110
	11	1011 1000 1100 0111 0001 1110 0010 1101 0110 1111 0000 1001 1010 0100 0101 0011

➤ Bảng  $S_6$

**Bảng 2. 13. Bảng  $S\text{-box}6$**

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
$b_0b_5$	00	1100 0001 1010 1111 1001 0010 0110 1000 0000 1101 0011 0100 1110 0111 0101 1011
	01	1010 1111 0100 0010 0111 1100 1001 0101 0110 0001 1101 1110 0000 1011 0011 1000
	10	1001 1110 1111 0101 0010 1000 1100 0011 0111 0000 0100 1010 0001 1101 1011 0110
	11	0100 0011 0010 1100 1001 0101 1111 1010 1011 1110 0001 0111 0110 0000 1000 1101

➤ Bảng  $S_7$

**Bảng 2. 14. Bảng  $S\text{-box}7$**

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
$b_0b_5$	00	0100 1011 0010 1110 1111 0000 1000 1101 0011 1100 1001 0111 0101 1010 0110 0001
	01	1101 0000 1011 0111 0100 1001 0001 1010 1110 0011 0101 1100 0010 1111 1000 0110
	10	0001 0100 1011 1101 1100 0011 0111 1110 1010 1111 0110 1000 0000 0101 1001 0010
	11	0110 1011 1101 1000 0001 0100 1010 0111 1001 0101 0000 1111 1110 0010 0011 1100

➤ Bảng  $S_8$

**Bảng 2. 15. Bảng  $S\text{-box}8$**

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
$b_0b_5$	00	1101 0010 1000 0100 0110 1111 1011 0001 1010 1001 0011 1110 0101 0000 1100 0111
	01	0001 1111 1101 1000 1010 0011 0111 0100 1100 0101 0110 1011 0000 1110 1001 0010
	10	0111 1011 0100 0001 1001 1100 1110 0010 0000 0110 1010 1101 1111 0011 0101 1000
	11	0010 0001 1110 0111 0100 1010 1000 1101 1111 1100 1001 0000 0011 0101 0110 1011

Ví dụ: Input  $S\text{-box}1$  là  $\underline{100011}_2$ . Bit số  $b_0=1$  và bit  $b_5=1 \rightarrow b_0b_5=11_2$  tương ứng với  $3_{10}$ . Tương tự giá trị  $b_1b_2b_3b_4=0001_2=1_{10}$ . Đối chiếu với bảng  $S\text{-box}1$  ở hàng 4, cột 2 nhận được giá trị  $S\text{-box}1(100011)=1100_2$ .

Có thể thấy, mỗi hàm  $S\text{-box}$  con là một phép thay thế *Substitution*. Các hàm  $S\text{-box}$  con không khả nghịch, do đó hàm  $S\text{-boxes}$  cũng không khả nghịch. Sự phức tạp này của  $S\text{-boxes}$  là yếu tố chính làm cho  $DES$  có độ an toàn cao.

- $P\text{-box}$ : hàm  $P\text{-box}$  cũng thực hiện hoán vị 32 bit đầu vào theo quy tắc:

**Bảng 2. 16. Bảng  $P\text{-box}$**

P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

$$\underbrace{p_1, p_2, p_3, \dots, p_{32}}_{32\text{bit}} \rightarrow \underbrace{p_{16}, p_7, p_{20}, \dots, p_{25}}_{32\text{bit}}$$

### c) Thuật toán sinh khóa con

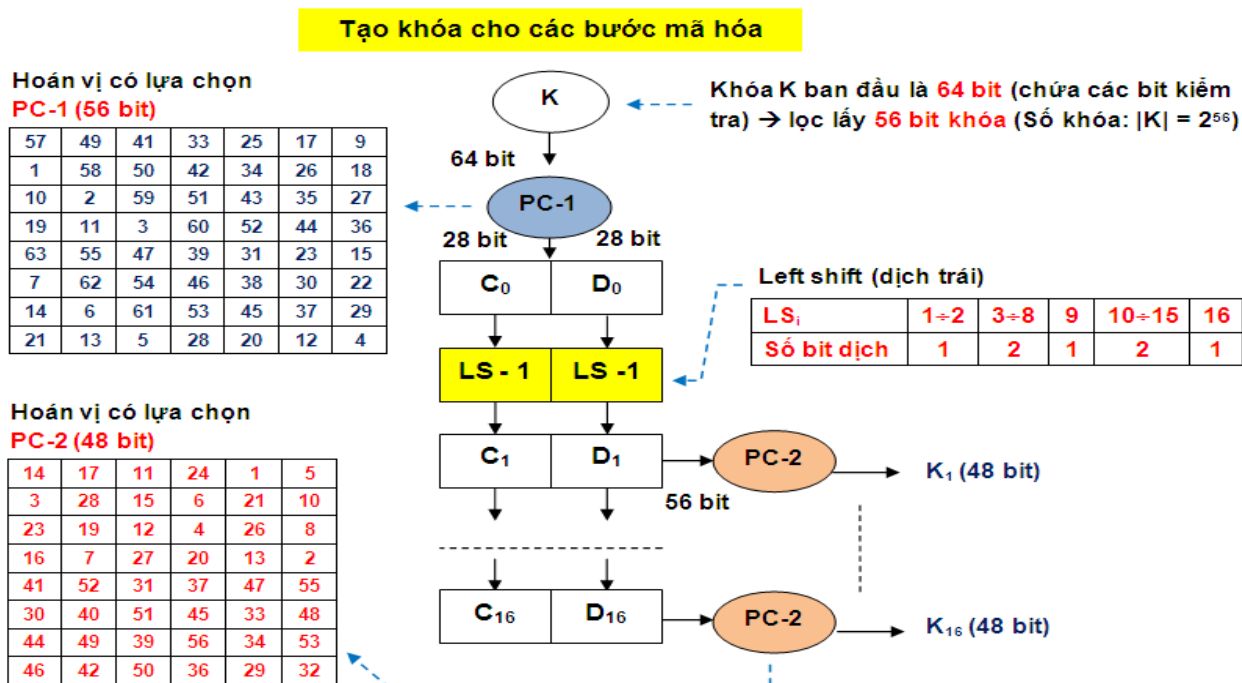
Thuật toán sinh khóa con cho từng vòng mã hóa của  $DES$  được thể hiện trong hình 2.6. Khóa  $K$  64 bit ban đầu được rút trích và hoán vị thành một khóa 56 bit (tức chỉ sử dụng 56 bit) theo quy tắc nén khóa (xem bảng 2.17).

Khóa 56 bit này được chia thành 2 nửa trái phải  $KL_0$  và  $KR_0$ , mỗi nửa có kích thước 28 bit. Tại vòng thứ  $i$  ( $i = 1, 2, 3, \dots, 16$ ),  $KL_{i-1}$  và  $KR_{i-1}$  được dịch vòng trái  $r_i$  bit để có được  $KL_i$  và  $KR_i$ , với  $r_i$  được định nghĩa như sau:

$$r_i = \begin{cases} 1 & \text{if } i \in (1, 2, 9, 16) \\ 2 & \text{if } i \notin (1, 2, 9, 16) \end{cases} \quad (2.1)$$

Cuối cùng khóa  $K_i$  của mỗi vòng được tạo ra bằng cách hoán vị và nén 56 bit của  $KL_i$  và  $KR_i$  thành 48 bit (xem bảng 2.18)





Hình 2. 6. Mô hình các bước tạo khóa của DES

Bảng 2. 17. Bảng nén khóa 64bit xuống 56bit DES

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

Nửa trên

63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Nửa dưới

Bảng 2. 18. Bảng nén khóa 56 bit xuống 48 bit DES

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Ví dụ mã hóa DES:

Cho Thông điệp  $P : 123456ABCD132536_{16}$

Khóa  $K : AAB09182736CCDD_{16}$

Bảng 2.19 trình bày kết quả qua các vòng lặp của DES

**Bảng 2. 19. Ví dụ mã hóa DES**

Thông điệp : $123456ABCD132536$			
Sau hoán vị khởi tạo : $14A7D6818CA18AD$			
Chia hai nửa trái phải $L_0=14A7D681$ , $R_0= 18CA18AD$			
Vòng lặp	Nửa trái	Nửa phải	Khóa sinh ra
Vòng lặp	<i>Nửa trái</i>	<i>Nửa phải</i>	<i>Khóa sinh ra</i>
Round1	$18CA18AD$	$5A78E394$	$194CD072DE8C$
Round2	$5A78E394$	$4A1210F6$	$4568581ABCCE$
Round3	$4A1210F6$	$B8089591$	$06EDA4ACF5B5$
Round4	$B8089591$	$236889C2$	$DA2D032B6EE3$
Round5	$236779C2$	$2E8F9C65$	$69A629FEC913$
Round6	$A15A4B87$	$2E8F9C65$	$C1948E87485E$
Round7	$2E8F9C65$	$A15A4B87$	$708AD2DDB3C0$
Round8	$A9FC20A3$	$308BEE97$	$34F822F0C66D$
Round9	$308BEE97$	$10AF9D37$	$84BB4473DCCC$
Round10	$10AF9D37$	$6CA6CB20$	$02765708B5BF$
Round11	$6CA6CB20$	$FF3C485F$	$6D5560AF7CA5$
Round12	$FF3C485F$	$22A5963B$	$C2C1E96A4BF3$
Round13	$22A5963B$	$387CCDAA$	$99C31397C91F$
Round14	$387CCDAA$	$BD2DD2AB$	$251B8BC717D0$
Round15	$BD2DD2AB$	$CF26B472$	$3330C5D9A36D$

Round16	CF26B472	19BA9212	181C5D75C66D
Giá trị thu được sau khi lặp 16 vòng : CF26B47219BA9212			
Bản mã (Sau khi thực hiện hoán vị kết thúc): C0B7A8D05F3A829C			

### **Độ an toàn của mã hóa DES**

Độ an toàn của thuật toán DES được quan tâm và đánh giá ngay từ lúc DES ra đời. Có nhiều ý kiến khác nhau, trong đó một số nhà thám mã đã chỉ ra được rằng, thuật toán DES không thực sự an toàn. Hãy xem xét tính an toàn của *DES* trước một vài phương pháp tấn công phá mã.

- Tấn công vét cạn khóa (*Brute Force Attack*): Vì khóa của thuật toán *DES* có chiều dài là 56 bit nên để tiến hành *brute-force attack*, cần kiểm tra  $2^{56}$  khóa khác nhau. Hiện nay với những thiết bị phổ dụng, thời gian gian để thử khóa là rất lớn nên việc phá mã là không khả thi (xem bảng). Tuy nhiên vào năm 1998, tổ chức *Electronic Frontier Foundation (EFF)* thông báo đã xây dựng được một thiết bị phá mã DES gồm nhiều máy tính chạy song song, trị giá khoảng 250.000\$. Thời gian thử khóa là 3 ngày. Hiện nay mã *DES* vẫn còn được sử dụng trong thương mại, tuy nhiên người đã bắt đầu áp dụng những phương pháp mã hóa khác có chiều dài khóa lớn hơn (128 bit hay 256 bit) như *Triple DES* hoặc *AES*.

- Phương pháp vi sai (*differential cryptanalysis*): Năm 1990 *Biham* và *Shamir* đã giới thiệu phương pháp phá mã vi sai. Phương pháp vi sai tìm khóa ít tốn thời gian hơn *brute-force*. Tuy nhiên phương pháp phá mã này lại đòi hỏi phải có  $2^{47}$  cặp thông điệp - bản mã được lựa chọn (*chosen-plaintext*). Vì vậy phương pháp này là bất khả thi dù rằng số lần thử có thể ít hơn phương pháp *brute-force*.

- Phương pháp thử tuyến tính (*linear cryptanalysis*): Năm 1997 *Matsui* đưa ra phương pháp phá mã tuyến tính. Trong phương pháp này, cần phải biết trước  $2^{43}$  cặp thông điệp-bản mã. Tuy nhiên  $2^{43}$  cũng là một con số lớn nên phá mã tuyến tính cũng không phải là một phương pháp khả thi.

### 2.3.1.2. Triple DES (3DES)

Một trong những cách để khắc phục yếu điểm về kích thước khóa ngắn của mã hóa DES là sử dụng mã hóa DES nhiều lần với các khóa khác nhau cho cùng một thông điệp. Đơn giản nhất là dùng DES hai lần với hai khóa khác nhau, cách thức này được gọi là Double DES. Công thức 2.2 mô tả về quy trình mã hóa của thuật toán Double DES.

$$C = E(E(P, K_1), K_2) \quad (2.2)$$

Theo công thức 2.2 có thể hiểu Double DES dùng một khóa có kích thước là 112 bit, chỉ có một hạn chế là tốc độ chậm hơn DES vì phải dùng DES hai lần. Tuy nhiên người ta đã tìm được một phương pháp tấn công Double DES có tên gọi là gặp-nhau-ở-giữa (*meet-in-the-middle*). Đây là một phương pháp tấn công *chosen-plaintext* (trình bày ở chương 1). Vì vậy người chọn dùng DES ba lần với ba khóa khác nhau, cách thức này được gọi là Triple DES (xem hình 2.7). Công thức 2.3 mô tả quá trình mã hóa của thuật toán Triple DES.

$$C = E(D(E(P, K_1), K_2), K_3) \quad (2.3)$$

Từ công thức 2.3 có thể thấy chiều dài khóa là 168 bit sẽ gây phức tạp hơn nhiều cho việc phá mã bằng phương pháp tấn công gặp-nhau-ở-giữa. Về thuật toán Triple DES có nhiều phương pháp khác nhau để lựa chọn các cặp khóa sử dụng mà vẫn đảm bảo an toàn cần thiết. Ví dụ công thức 2.4 mô tả thuật toán Triple DES chỉ dùng hai khóa  $K_1, K_2$ .

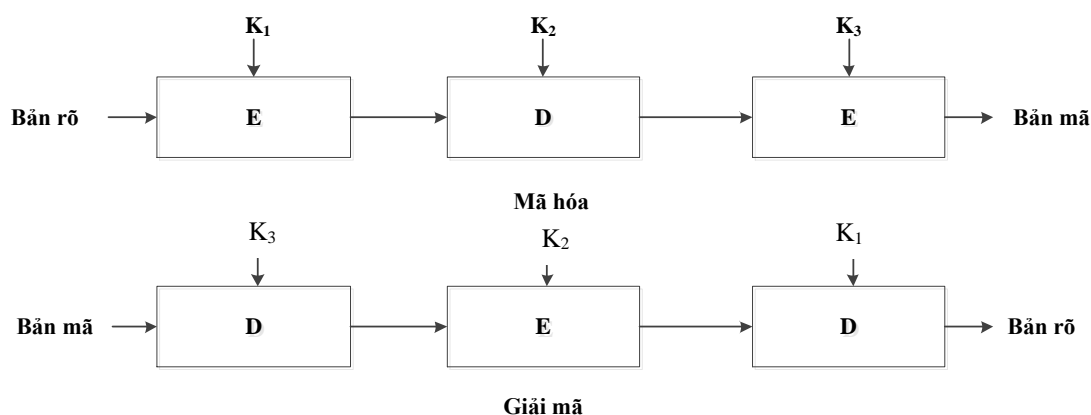
$$C = E(D(E(P, K_1), K_2), K_1) \quad (2.4)$$

Nguyên nhân của việc dùng EDE thay cho EEE là nếu với  $K_1 = K_2 = K$  thì:

$$C = E(E(E(P, K), K), K) = E(P, K) \quad (2.5)$$

Nghĩa là Triple DES suy giảm thành DES.

- Các lựa chọn khóa :
  - Lựa chọn 1: cả 3 khóa độc lập (168 bit)
  - Lựa chọn 2: ví dụ  $K_1$  và  $K_2$  độc lập,  $K_3 = K_1$  (112 bit)
  - Lựa chọn 3: 3 khóa giống nhau,  $K_1 = K_2 = K_3$  (56 bit)
- Sơ đồ mã hóa và giải mã Triple DES



Hình 2. 7. Sơ đồ mã hóa và giải mã *Triple DES*

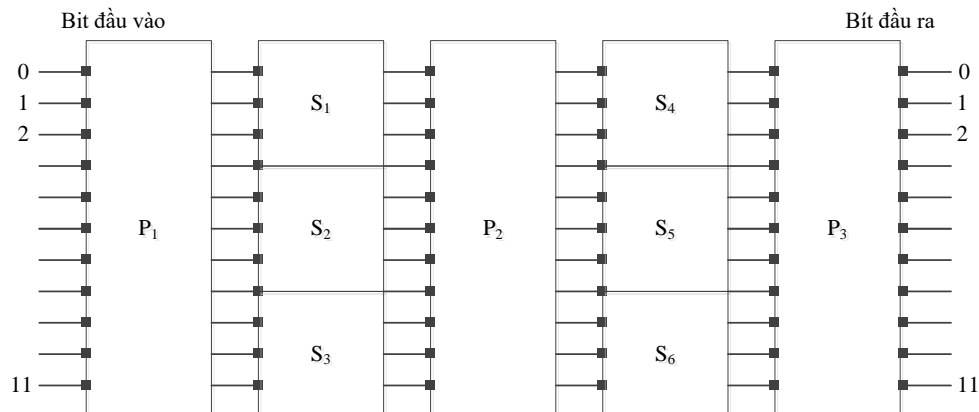
### 2.3.2. Thuật toán mã hóa AES

#### 2.3.2.1. Nguồn gốc ra đời của thuật toán AES

Vào năm 1997 nhận thấy nguy cơ mã hóa *DES* là kích thước khóa ngắn có thể bị phá vỡ, do đó Viện chuẩn quốc gia Hoa Kỳ *US NIST* đã kêu gọi xây dựng một phương pháp mã hóa mới. Đến tháng 10/2000 thuật toán có tên là Rijndael được chọn làm mật mã nâng cao và xuất bản là chuẩn *FIPS PUB 197*. Vào 11/2001 và được đổi tên thành *Advanced Encryption Standard (AES)*. Như đã trình bày ở trên, thuật toán mã hóa *DES* được thực hiện dựa trên mã hóa khối Feistel. Tuy nhiên, cơ chế mã hóa của *AES* lại có điểm khác với *DES* là sử dụng mạng thay thế hoàn vị (*SPN - Substitution-Permutation Network*). Để hiểu rõ hơn về cơ chế mã hóa của thuật toán mã hóa *AES*, trong phần tiếp theo, bài giảng sẽ trình bày tổng quan về *SPN*.

#### 2.3.2.2. Mạng thay thế hoán vị

Trong thực tế, cần tìm phương pháp, cách thức sao cho chỉ cần dùng một khóa có kích thước ngắn để giả lập một bảng tra cứu có độ an toàn xấp xỉ độ an toàn của mã khối lý tưởng. Cách thực hiện là kết hợp hai hay nhiều mã hóa đơn giản lại với nhau để tạo thành một mã hóa tổng (*product cipher*), trong đó mã hóa tổng an toàn hơn rất nhiều so với các mã hóa thành phần. Các mã hóa đơn giản thường là phép thay thế (*substitution, S-box*) và hoán vị (*Permutation, P-box*). Do đó người ta hay gọi mã hóa tổng là *Substitution-Permutation Network* (mạng *SPN*). Hình dưới minh họa một mạng *SPN*.



**Hình 2. 8. Sơ đồ SPN**

Việc kết hợp các *S-box* và *P-box* tạo ra hai tính chất quan trọng của mã hóa là tính khuếch tán (Tính chất này có được dựa vào sử dụng *P-box* kết hợp *S-box*) và tính gây lẫn (Tính chất này có được dựa vào sử dụng *S-box*). Hai tính chất này do *Claude Shannon* giới thiệu vào năm 1946, và là cơ sở của tất cả các mã khối hiện nay. Chương 1 bài giảng đã trình bày 2 tính chất này.

### 2.3.2.3. Biểu diễn dữ liệu trong AES

#### Biểu diễn ma trận byte

*AES* là một mã khối, nhưng khác với các mã khối khác được biết đến trước đây (*DES*, *IDEA*,...), dữ liệu trong *AES* không được biểu diễn dưới dạng một mảng các byte hay các bit mà được biểu diễn dưới dạng một ma trận  $4 \times Nb$  và được gọi là mảng trạng thái (*state*). Trong đó, đối với *AES*,  $Nb$  luôn có giá trị bằng 4. Trong khi thuật toán *Rijndael* hỗ trợ ba giá trị của  $Nb$  là 4, 6, 8 tương ứng với kích thước khối 128, 192 và 256 bit. Biểu diễn ma trận trạng thái của *AES* được thể hiện trong biểu thức 2.6.

Dữ liệu đầu vào được đọc vào ma trận *state* theo từng cột, theo thứ tự từ trên xuống dưới, từ trái qua phải. Dữ liệu đầu ra được đọc từ ma trận cũng theo quy tắc trên.

$$S = \begin{pmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{pmatrix} \quad (2.6)$$

Khóa vòng trong *AES* cũng được biểu diễn hoàn toàn tương tự như cách biểu diễn dữ liệu (xem công thức 2.8). Tuy nhiên, tùy vào kích thước khóa mà số cột của ma trận khóa vòng  $N_k$  sẽ khác nhau. Cụ thể,  $N_k$  nhận các giá trị 4, 6, 8 tương ứng với các kích thước khóa là 128, 192 và 256 bit. Đây chính là điểm mạnh của thuật toán *AES* trong vấn đề mở rộng khóa.

$$k = \begin{pmatrix} k_{00} & k_{01} & \dots & k_{0N_k} \\ k_{10} & k_{11} & \dots & k_{1N_k} \\ k_{20} & k_{21} & \dots & k_{2N_k} \\ k_{30} & k_{31} & \dots & k_{3N_k} \end{pmatrix} \quad (2.8)$$

### Số vòng lặp $N_r$

Số vòng lặp ký hiệu là  $N_r$ , phụ thuộc vào hai đại lượng  $N_b$  và  $N_k$ . Vì  $N_b$  trong *AES* có giá trị cố định nên  $N_r$  chỉ phụ thuộc vào  $N_k$ . Giá trị của  $N_r$  tương ứng với ba giá trị của  $N_k$  là  $N_r = 10, 12, 14$ . Cụ thể, giá trị  $N_r$  được xác định bởi bảng bên dưới:

$N_k$	4	6	8
$N_r$	10	12	14

### Khái niệm từ (*Word*) trong *AES*

Bốn byte trên mỗi cột trong mảng trạng thái *state* tạo thành 1 từ 32 bit, trong đó số thứ tự của hàng  $r$  ( $0 \leq r < 4$ ) cho biết chỉ số của bốn byte trong mỗi từ. Từ định nghĩa *state* ở trên có thể coi *state* là mảng một chiều chứa các từ 32 bit  $s_0 \dots s_3$  bởi công thức 2.9.

$$\begin{aligned} s_0 &= s_{00} s_{10} s_{20} s_{30} \\ s_1 &= s_{01} s_{11} s_{21} s_{31} \\ s_2 &= s_{02} s_{12} s_{22} s_{32} \\ s_3 &= s_{03} s_{13} s_{23} s_{33} \end{aligned} \quad (2.9)$$

Tương tự như đối với mảng khóa cũng có thể biểu diễn thành mảng một chiều chứa các từ 32 bit như công thức dưới đây với số lượng từ khóa phụ thuộc vào  $N_k$  ( $N_k=4, 6, 8$ ).

$$k_0, k_1 \dots k_{N_k}$$

Như vậy mảng khóa vòng sẽ được biểu diễn bởi:  $k_0, k_1 \dots k_{(Nr+1)*4}$

#### 2.3.2.4. Cở sở toán học của AES

**Phép cộng trên trường  $GF(2^8)$ :** Chính là phép XOR hay phép cộng theo modulo 2 giữa các bit tương ứng giữa 2 byte. Nghĩa là:

Giả sử:  $A = a_7a_6a_5a_4a_3a_2a_1a_0$

$B = b_7b_6b_5b_4b_3b_2b_1b_0$

$C = A + B = c_7c_6c_5c_4c_3c_2c_1c_0$

trong đó:  $c_i = (a_i + b_i) \bmod 2$  ( $0 \leq i \leq 7$ )

Ví dụ: tổng của  $A = 73_H$  và  $4E_H$  là:

- Dạng nhị phân      01110011                      + 01001110                      = 00111101
- Dạng đa thức       $(x^6 + x^5 + x^4 + x + 1)$       +  $(x^6 + x^3 + x^2 + x)$                       =  $(x^5 + x^4 + x^3 + x^2 + 1)$
- Dạng thập lục      73<sub>H</sub>    + 4E<sub>H</sub>    = 3D<sub>H</sub>  
phân

**Phép nhân trên trường  $GF(2^8)$ :** Phép nhân được thực hiện trên trường  $GF(2^8)$  bằng cách nhân hai đa thức rút gọn theo modulo của một đa thức bất khả quy  $m(x)$ . Đa thức bất khả quy là đa thức chỉ có ước là 1 và chính nó. Trong AES đa thức bất khả quy này là:

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

Giả sử:  $a(x) = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + 1$

$b(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + 1$

$\Rightarrow c(x) = a(x).b(x) \bmod m(x)$

Ví dụ: tích của  $A = 73_H$  và  $B = 4E_H$  là  $C = AB \bmod m(x)$

- Dạng nhị phân:      01110011                      • 01001110                      = 01011011



- Dạng đa thức:  $(x^6+x^5+x^4+x+1)$  •  $(x^6+x^3+x^2+x)$  =  $(x^6+x^4+x^3+x+1)$
- Dạng thập lục phân:  $73_H$  •  $4E_H$  =  $5B_H$

### 2.3.2.5. Thuật toán AES

Thuật toán AES khá phức tạp, được mô tả khái quát gồm 3 bước như sau:

- Vòng khởi tạo chỉ gồm phép *AddRoundKey*
- Vòng lặp gồm 4 phép biến đổi lần lượt: *SubBytes*, *ShiftRows*, *MixColumns*, *AddRoundKey*.
- Vòng cuối gồm các phép biến đổi giống vòng lặp và không có phép *MixColumns*.

### Đặc tả chi tiết thuật toán mã hóa AES

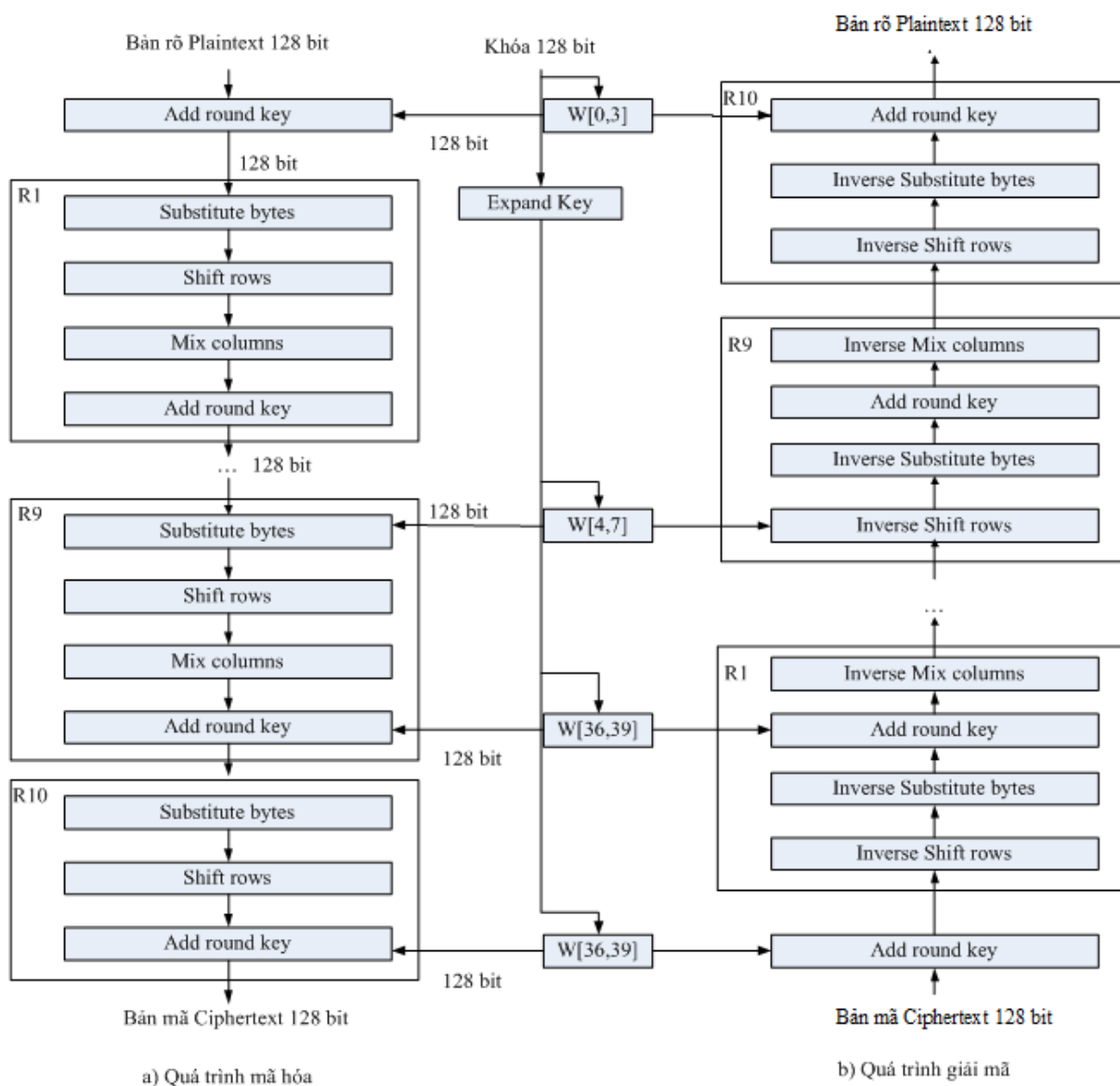
Cấu trúc sơ đồ mã hóa được thể hiện trên Hình 2.9 gồm: vòng khởi tạo,  $Nr-1$  vòng lặp và vòng kết thúc. Trong đó vòng khởi tạo chỉ có phép biến đổi *AddRoundKey*, vòng lặp gồm 4 phép biến đổi chính: *SubBytes*, *ShiftRows*, *MixColumns*, *AddRoundKey*; vòng kết thúc khác với vòng lặp chính ở chỗ không có phép *MixColumns*.

Riêng đối với cấu trúc giải mã trong AES gồm 2 chế độ giải mã:

- Ở cấu trúc giải mã ngược, gồm vòng khởi tạo,  $Nr-1$  vòng lặp và vòng kết thúc. Trong đó vòng khởi tạo chỉ có phép biến đổi *AddRoundKey*, vòng lặp gồm lần lượt 4 phép biến đổi chính: *InvShiftRows*, *InvSubBytes*, *AddRoundKey*, *InvMixColumns*; vòng kết thúc khác với vòng lặp chính ở chỗ không có phép *InvMixColumns*.
- Ngược lại với cấu trúc giải mã ngược là cấu trúc giải mã xuôi, việc ngược lại thể hiện ở điểm: trong cấu trúc giải mã xuôi việc sắp xếp các phép biến đổi ngược giống hệt với cấu trúc mã hóa, cụ thể bao gồm: vòng khởi tạo,  $Nr-1$  vòng lặp và vòng kết thúc. Trong đó vòng khởi là phép *AddRoundKey*; ở vòng lặp thứ tự các phép biến đổi ngược lần lượt là: *InvSubBytes*, *InvShiftRows*, *InvMixColumns*, *AddRoundKey*; vòng kết thúc giống vòng lặp nhưng được lược bỏ phép *InvMixColumns*.

Một điểm khác biệt nữa trong hai cấu trúc giải mã ngược và giải mã xuôi đó là: trong giải mã ngược khóa vòng giải mã chính là khóa vòng mã hóa với thứ tự đảo ngược;

riêng trong giải mã xuôi thì khóa giải mã ngoài việc đảo ngược thứ tự khóa vòng mã hóa còn phải thực hiện phép *InvMixColumns* đối với các khóa vòng của vòng lặp giải mã.



**Hình 2. 9. Sơ đồ mã hóa giải mã AES**

Từ mô hình mã hóa của thuật toán AES trong hình 2.9, mô tả giải thuật AES gồm các bước chính trong vòng lặp như sau:

### ***SubBytes và InvSubBytes***

- Phép biến đổi SubBytes: Là phép thay thế byte phi tuyến tính, ở phép thay thế này nó tác động độc lập đến từng byte trong trạng thái hiện hành. Phép biến đổi *SubBytes*

được thực hiện bằng cách tra cứu bảng thay thế (S-box) với tham số đầu vào là các byte trong bảng trạng thái. S-box được xây dựng như sau:

Bước 1: Điền các con số từ 0 đến 255 vào bảng theo từng hàng. Vậy hàng 0 gồm các con số {00}, {01}, ...{0F} (thập lục phân). Hàng 1 gồm các con số {10}, {11},..., {1F}. Điều này có nghĩa là tại hàng x cột y có giá trị {xy}.

Bước 2: Thay thế mỗi byte trong bảng bằng giá trị nghịch đảo trong trường  $GF(2^8)$ . Quy ước nghịch đảo của {00} cũng là {00}.

Bước 3: Mỗi byte trong ma trận *state* được thay thế bởi 1 byte trong *Rijndael* S-box, hay  $b_{ij} = S(a_{ij})$ .

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \quad (2.10)$$

trong đó,  $0 \leq i \leq 8$  là bit thứ i của byte b tương ứng và  $c_i$  là bit thứ i của byte c với giá trị {63} hay {01100011}.

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (2.11)$$

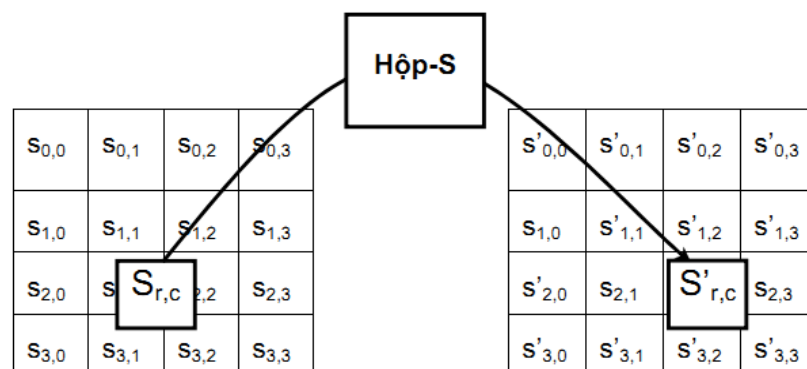
Trong đó phép cộng thực hiện như phép XOR. Bảng 2.20 trình bày nội dung bảng S-box sau khi tính toán.

**Bảng 2. 20. Bảng S-box AES**

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6b	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84

5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Hình 2.10 minh họa tác động của phép SubByte() lên mảng trạng thái *State*:



**Hình 2. 10. Phép biến đổi SubByte**

Như vậy nếu giá trị byte cần thay thế là  $S_{r,c}=\{xy\}$  thì sẽ đóng vào *S-box* hàng  $x$  cột  $y$  thì thu được byte thay thế  $S'_{r,c}$ .

Ví dụ: phép *SubByte*

ea	04	65	85
80	50	5d	96
5e	33	98	b0
f0	ab	ad	c5

SubByte →

87	f2	4d	97
ec	6e	4c	90
4a	c3	46	e7
8c	d8	95	a6

• Phép biến đổi ngược  $InvSubBytes$ : là phép thay thế biến đổi ngược với  $SubBytes$ . Là một phép thay thế byte, các byte thay thế được thực hiện bằng cách tra bảng thay thế ngược  $IS$ . Bảng thay thế ngược  $IS$  này được xây dựng như sau:

Trước tiên, cũng phải xây dựng một bảng  $Inverse SubBytes$  ( $IS$ - box). Nghĩa là nếu với đầu vào  $\{95\}$ ,  $S$ -box cho ra kết quả  $\{2A\}$ , thì với đầu vào là  $\{2A\}$ ,  $IS$  sẽ cho ra lại kết quả  $\{95\}$ . Việc xây dựng hộp  $IS$  cũng giống như xây dựng  $S$ -box tại bước 1 và bước 2. Tại bước 3,  $IS$  thực hiện phép thay thế sau:

$$b_i = b'_i \oplus b'_{(i+2) \bmod 8} \oplus b'_{(i+5) \bmod 8} \oplus b'_{(i+7) \bmod 8} \oplus d_i \quad (2.12)$$

Với  $d_i$  là bit thứ  $i$  của số  $\{05\}$  tức  $d_7 d_6 d_0 = 00000101$ . Và phép nhân ma trận tương đương  $B = YB' \oplus D$  :

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Bảng 2.21 trình bày nội dung hộp thay thế ngược  $IS$ :

**Bảng 2. 21. Hộp thay thế ngược  $IS$**

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73

9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

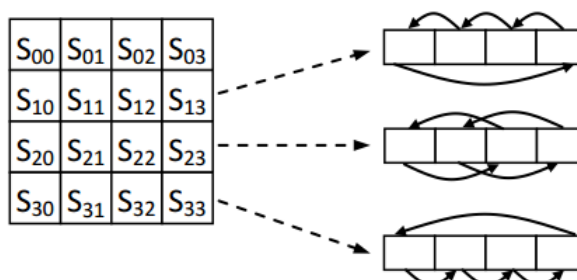
Như vậy: phép biến đổi *InvSubBytes* thực hiện như sau: Mỗi byte trong ma trận state  $S$ , dưới dạng thập lục phân là  $\{xy\}$ , được thay thế bằng giá trị trong bảng  $IS$  tại dòng  $x$  cột  $y$ .

- Mục đích của phép biến đổi *SubBytes*: *S-box* dùng để chống lại hình thức tấn công thám mã vi sai và thám mã tuyến tính. Giữa input và output của phép *Substitute bytes* không thể mô tả bằng một công thức toán đơn giản.

### ***ShiftRows* và *InvShiftRows***

- Phép biến đổi *ShiftRows*: Thao tác *ShiftRows* (xem hình 2.11) thực hiện hoán vị các byte trong ma trận *state* theo cách thức sau:

- ✓ Dòng thứ nhất giữ nguyên
- ✓ Dòng thứ 2 dịch vòng trái 1 byte
- ✓ Dòng thứ 3 dịch vòng trái 2 byte
- ✓ Dòng thứ 4 dịch vòng trái 3 byte



**Hình 2. 11. Phép biến đổi ShiftRows**

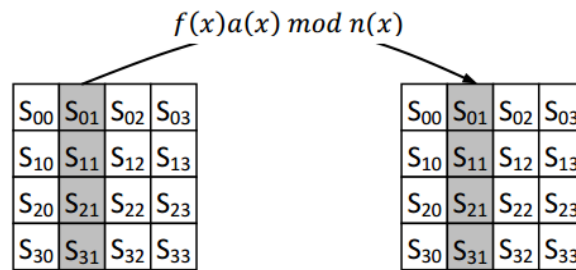
- Phép biến đổi *InvShiftRows*: Phép biến đổi *InvShiftRows* thực hiện ngược lại với phép *ShiftRows*, nghĩa là:
  - ✓ Dòng thứ nhất giữ nguyên
  - ✓ Dòng thứ 2 dịch vòng phải 1 byte
  - ✓ Dòng thứ 3 dịch vòng phải 2 byte
  - ✓ Dòng thứ 4 dịch vòng phải 3 byte
- Mục đích của *ShiftRows*: Xáo trộn các byte để tạo các cột khác nhau trước khi sử dụng cột cho thao tác *MixColumns*.

### ***MixColumns* và *InvMixColumns***

- Phép biến đổi *MixColumns*: Phép biến đổi *MixColumns* thực hiện biến đổi độc lập từng cột trong ma trận *state* bằng một phép nhân đa thức. Mỗi cột của state được coi là biểu diễn của một đa thức  $f(x)$  trong  $GF(2^8)$  như vậy phép biến đổi *MixColumns* chính là phép nhân theo modulo với  $x^4+1$  với một đa thức cố định định nghĩa như sau:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (2.13)$$

Hình 2.12 mô tả thao tác trên các cột của phép biến đổi *MixColumn*:



**Hình 2. 12. Phép biến đổi *MixColumn***

Phép nhân đa thức trên có thể biểu diễn dưới dạng phép nhân ma trận như sau:

$$\begin{bmatrix} s'_{01} \\ s'_{11} \\ s'_{21} \\ s'_{31} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{01} \\ s_{11} \\ s_{21} \\ s_{31} \end{bmatrix} \quad (2.14)$$

Hình 2.13 trình bày một ví dụ về phép *MixColumns*:

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	34	C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	B

Hình 2. 13. Ví dụ về phép MixColumn

- Phép biến đổi ngược  $InvMixColumns$ : Là phép biến đổi ngược với phép biến đổi  $MixColumns$ .  $InvMixColumns$  cũng thực hiện thao tác theo từng cột của trạng thái, xem mỗi cột như một đa thức bậc 3 gồm 4 hạng tử trên trường  $GF(2^8)$ . Các cột của phép  $InvMixColumns$  được nhân theo modulo  $(x^4 + 1)$  với đa thức nghịch đảo  $a(x)$  chính là đa thức  $a^{-1}(x)$  được định nghĩa:

$$a^{-1}(x) = \{0B\}.x^3 + \{0D\}.x^2 + \{09\}.x^1 + \{0E\} \quad (2.14)$$

Như vậy phép  $InvMixColumns$  cũng được biểu diễn tương đương với phép nhân ma trận sau:

$$\begin{bmatrix} s'_{0c} \\ s'_{1c} \\ s'_{2c} \\ s'_{3c} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{bmatrix}$$

trong đó  $c$  là cột thứ  $c$  ( $0 \leq c \leq 3$ ) của ma trận trạng thái  $state$

- Mục đích của  $MixColumns$ :

Việc coi mỗi cột là một đa thức bậc 3, rồi nhân mỗi cột với đa thức  $a(x)$  sau đó modulo  $(x^4 + 1)$  đã làm cho mỗi byte trong cột kết quả đều phụ thuộc vào bốn byte trong cột ban đầu. Thao tác  $MixColumns$  kết hợp với  $ShiftRows$  đảm bảo rằng sau một vài vòng biến đổi, 128 bit trong kết quả đều phụ thuộc vào tất cả 128 bit ban đầu. Điều này tạo ra tính khuếch tán (*diffusion*) cần thiết cho mã hóa.

### AddRoundKey



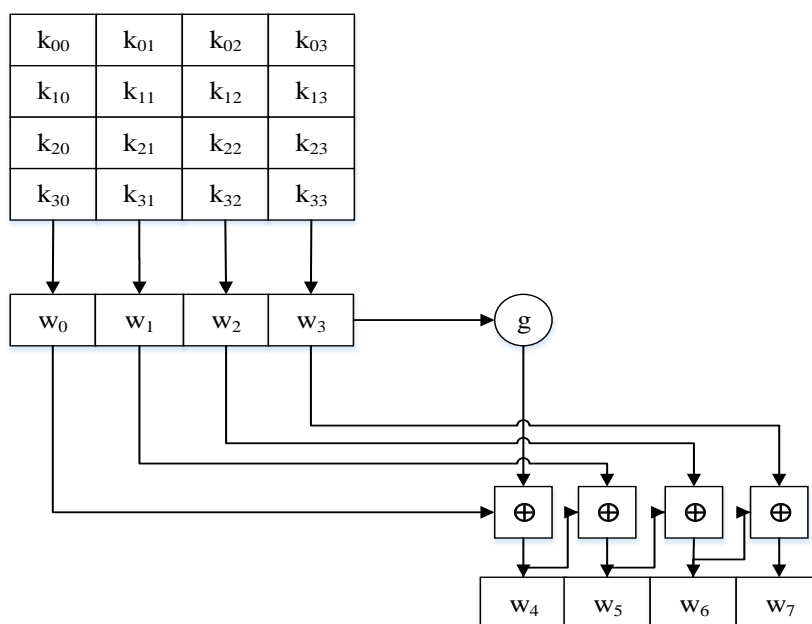
Trong thao tác *AddRoundKey*, 128 bit của ma trận *state* sẽ được *XOR* với 128 bit của khóa con của từng vòng. Vì sử dụng phép *XOR* nên phép biến đổi ngược của *AddRoundKey* trong cấu trúc giải mã cũng chính là *AddRoundKey*.

Việc kết hợp với khóa bí mật tạo ra tính làm rối (*confusion*) của mã hóa. Sự phức tạp của thao tác mở rộng khóa (*KeySchedule*) giúp gia tăng tính làm rối này.

### Mở rộng khóa (*ExpandKey*)

*ExpandKey* là thao tác tạo lược đồ khóa hay mở rộng khóa, tạo ra  $Nr+1$  khóa vòng từ khóa chính  $K$ , mỗi khóa vòng gồm  $N_b$  từ 32 bit, trong đó đối với *AES* thì  $N_b = 4$ , còn  $Nr$  được xác định theo. Các phép biến đổi để tạo khóa vòng trong *ExpandKey* là khác nhau đối với các giá trị khác nhau của kích thước khóa  $K$ .

Sau đây là việc mở rộng khóa đối với khóa mã 128 bit



**Hình 2. 14. Biểu diễn sinh  $N_k$  khóa đầu tiên đối với khóa mã 128 bit**

Trong thao tác mở rộng khóa với khóa mã 128 bit có đầu vào là 16 byte (4 *word*) của khóa mã, và sinh ra một mảng khóa vòng  $(Nr+1) \times 4 = 44$  từ (*word*) hay 176 byte. 44 *word* này được sử dụng cho 11 vòng mã hóa của *AES*, mỗi vòng dùng 4 *word*.

Từ bốn *word* đầu vào  $w_0 w_1 w_2 w_3$ , trong lần lặp đầu tiên thao tác *ExpandKey* sinh ra bốn *word*  $w_4 w_5 w_6 w_7$ , lần lặp thứ 2 từ  $w_4 w_5 w_6 w_7$  sinh ra  $w_8 w_9 w_{10} w_{11}$ , cứ như thế cho

đến lần lặp thứ 10 (tùy thuộc chiều dài khóa) sinh ra bốn word cuối cùng  $w_{40}w_{41}w_{42}w_{43}$  như hình vẽ bên dưới.

Trong mỗi lần lặp để sinh ra 4 word, word đầu tiên sinh ra theo quy tắc  $w_i = w_{i-4} \oplus g$  với  $g = \text{SubWord}(\text{RotWord}(w_{i-1})) \oplus \text{Rcon}[i/4]$ . Ba word tiếp theo sinh ra theo quy tắc  $w_i = w_{i-4} \oplus w_{i-1}$ . Sau đây, sẽ tìm hiểu các hàm SubWord, RotWord và mảng Rcon.

- ✓ *RotWord*: dịch vòng trái một byte. Giả sử word đầu vào có 4 byte là  $[b_0, b_1, b_2, b_3]$  thì kết quả của *RotWord* là  $[b_1, b_2, b_3, b_0]$ .
- ✓ *SubWord*: thay thế mỗi byte trong word đầu vào bằng cách tra cứu bảng S-box trong thao tác Substitute Bytes.
- ✓ *Rcon*: là một mảng hằng số. Mảng này gồm 10 word ứng với 10 vòng AES. Bốn byte của một phần tử  $\text{Rcon}[j]$  là  $(\text{RC}[j], 0, 0, 0)$  với  $\text{RC}[j]$  là mảng 10 byte như sau (với  $\text{RC}[j] = \text{RC}[j-1]*2$  với phép nhân thực hiện trong  $GF(2^8)$ ):

j	1	2	3	4	5	6	7	8	9	10
RC[j]	1	2	4	8	1	20	4	80	1B	36

- *Ví dụ ExpandKey*:

Cho key ban đầu :  $(24\ 75\ A2\ B3\ 34\ 75\ 56\ 88\ 31\ E2\ 12\ 00\ 13\ AA\ 54\ 87)_{16}$ . Bảng 2.22 chính là quá trình sinh khóa.

- *Mục đích của ExpandKey*: dùng để chống lại *known-plaintext attack*
  - ✓ Biết một số bit của khóa hay khóa con cũng không thể tính các bit còn lại.
  - ✓ Không thể tính ngược: biết một khóa con cũng không thể tính lại các khóa con trước đó.
  - ✓ Tính khuếch tán: một bit của khóa chính tác động lên tất cả các bit của các khóa con.

**Bảng 2. 22. Bảng kết quả của quá trình sinh khóa**

Round	T giá trị trung gian lưu $w$	First word in the round	Seconds word in the round	Third word in the round	Fourth word in the round
		w00=2475A2B3	w01=34755688	w02=31E21200	w03=13AA5487
1	AD20177D	w04=8955B5CE	w05=BD20E346	w06=8CC2F146	w07=9F68A5C1
2	470678DB	w08=CE53CD15	w09=73732E53	w10=FFB1DF15	w11=60D97AD4
3	31DA48DO	w12=FF8985C5	w13=8CFAAB96	w14=734B7483	w15=13920E57
4	47AB5B7D	w16=B822DEB8	w17=34D8752E	W18=479301AD	w19=54010FFA
5	6C762D20	w20=D454F398	w21=E08C86B6	w22=A71F871B	w23=F31E88E1
6	52C4F80D	w24=86900B95	w25=661C8D23	w26=C1030A38	w27=321D82D9
7	E4133523	w28=62833EB6	w29=049FB395	w30=C59CB9AD	w31=F7813B74
8	8CE29268	w32=EE61ACDE	w33=EAFE1F4B	w34=2F62A6E6	w35=D8E39D92
9	OA5E4F61	w36=E43FE3BF	w37=OEC1FCF4	w38=21A35A12	w39=F940C780
10	3PC6CD99	w40=DBF92E26	w41=D538D2D2	w42=F49B88CO	w43=ODDB4F40

**Kết luận:** Phương pháp mã hóa *AES* đơn giản, có thể thực hiện hiệu quả trên các vi xử lý 8 bit (dùng trong *smartcard*) cũng như trên các vi xử lý 32 bit, chỉ dùng phép *XOR* và phép *Shift* bit. Đây chính là yếu tố cơ bản để phương pháp này được chọn làm chuẩn mã hóa của *Hoa Kỳ*.

### 2.3.3. Thuật toán mã hóa A5/1

Trong mạng điện thoại GSM, để bảo mật dữ liệu trong quá trình truyền tin giữa trạm thu phát sóng vô tuyến và máy điện thoại người sử dụng A5/1 để mã hóa dữ liệu. Đơn vị mã hóa của A5/1 là một bit. Bộ sinh số ngẫu nhiên mỗi lần sẽ sinh ra bit 1 hoặc bit 0 để XOR với thông điệp để cho ra bản mã.

Như đã trình bày trong phần phân loại mật mã trong bài giảng, thuật toán mã hóa A5/1 là thuật toán mã dòng và mang đầy đủ các tính chất, đặc trưng cơ bản của mã dòng.

Để hiểu rõ hơn về cơ chế mã hóa và giải mã của thuật toán mã hóa A5/1 trước tiên, bài giảng sẽ trình bày về mô hình thu nhỏ của A5/1 là Tiny A5/1.

**a. Tiny A5 / 1**

Cơ chế hoạt động của bộ sinh số trong A5/1:

Bộ sinh số gồm 3 thanh ghi X, Y, Z; thanh ghi X gồm 6 bit ký hiệu ( $x_0 x_1 x_2 \dots x_5$ ); thanh ghi Y gồm 8 bit ( $y_0 y_1 y_2 \dots y_7$ ); thanh ghi Z gồm 9 bit ( $z_0 z_1 z_2 \dots z_8$ ). Khóa K ban đầu có chiều dài 23 bit và lần lượt phân bố vào các thanh ghi X, Y, Z. Các thanh ghi X, Y, Z biến đổi theo 3 nguyên tắc sau đây:

- Quay X gồm các thao tác
  - $t = x_2 \oplus x_4 \oplus x_5$
  - $t_j = x_{j-1}$  với  $j = 5, 4, 3, 2, 1$
  - $x_0 = t$

Ví dụ: giả sử X là 100101, dẫn đến  $t = 0 \oplus 0 \oplus 1 = 1$ , vậy sau khi quay giá trị của X là 110010.

- Quay Y gồm các thao tác
  - $t = y_6 \oplus y_7$
  - $y_j = y_{j-1}$  với  $j = 7, 6, 5, 4, 3, 2, 1$
  - $y_0 = t$

Ví dụ: giả sử Y là 10010111, dẫn đến  $t = 0 \oplus 1 \oplus 1 = 0$ , vậy sau khi quay giá trị của Y là 01001011.

- Quay Z gồm các thao tác
  - $t = z_2 \oplus z_7 \oplus z_8$
  - $z_j = z_{j-1}$  với  $j = 8, 7, 6, 5, 4, 3, 2, 1$
  - $z_0 = t$

Ví dụ: giả sử Z là 10010111, dẫn đến  $t = 0 \oplus 1 \oplus 1 = 0$ , vậy sau khi quay giá trị của Z là 010010111.

Cho ba bit  $x, y, z$ , định nghĩa một hàm  $maj(x, y, z)$  là hàm “*chiếm đa số*”, nghĩa là nếu trong 3 bit  $x, y, z$  có từ hai bit 0 trở lên thì hàm trả về giá trị 0, nếu không hàm trả về giá trị 1.

Tại bước sinh số thứ  $i$ , các phép tính sau được thực hiện:

$$m = maj(x_1, y_2, z_3)$$

- Nếu  $x_1 = m$  thì thực hiện quay  $X$
  - Nếu  $y_2 = m$  thì thực hiện quay  $Y$
  - Nếu  $z_3 = m$  thì thực hiện quay  $Z$
- Bit sinh ra là :  $s_i = x_5 \oplus y_7 \oplus z_8$

Bit  $s_i$  được XOR với bit thứ  $i$  trong thông điệp để có được bit thứ  $i$  trong bản mã theo quy tắc của mã dòng.

**Ví dụ:** mã hóa thông điệp  $P=111$  (chữ h) với khóa  $K$  là :

$$100101.01001110.100110000$$

Ban đầu giá trị của các thanh ghi  $X, Y, Z$  là:

$$X = 10010$$

$$Y = 01001110$$

$$Z = 100110000$$

- Bước 0:  $x_1 = 0, y_3 = 0, z_3 = 1 \rightarrow m = 0 \rightarrow$  quay  $X$ , quay  $Y$

$$X = 110010$$

$$Y = 10100111 \rightarrow s_0 = 0 \oplus 1 \oplus 0 = 1$$

$$Z = 100110000$$

- Bước 1:  $x_1 = 1, y_3 = 0, z_3 = 1 \rightarrow m = 1 \rightarrow$  quay  $X$ , quay  $Z$

$$X = 111001$$

$$Y = 10100111 \rightarrow s_1 = 1 \oplus 1 \oplus 0 = 1$$

$$Z = 010011000$$

- Bước 2:  $x_1 = 1, y_3 = 0, z_3 = 0 \rightarrow m = 0 \rightarrow$  quay  $Y$ , quay  $Z$

$$X = 111001$$

$$Y = 01010011 \rightarrow s_2 = 1 \oplus 1 \oplus 0 = 0$$

$$Z = 001001100$$

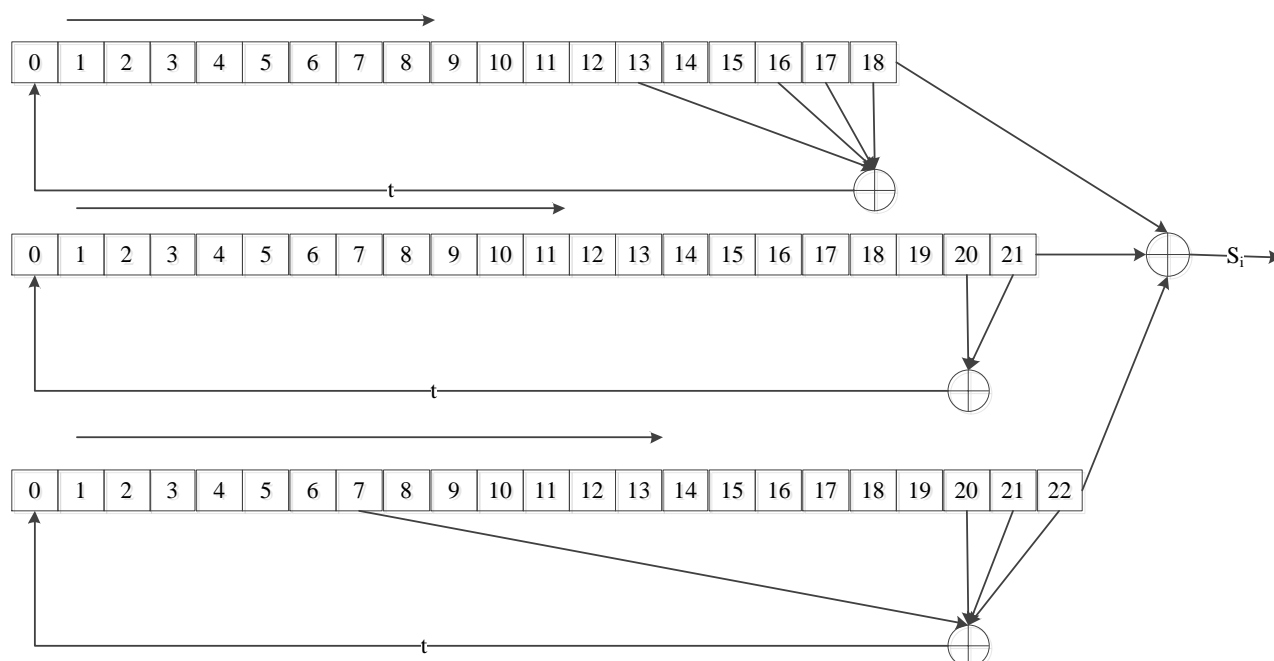
Vậy bản mã là  $C = 111 \oplus 100 = 011$  (chữ  $D$ )

### **b. A5/1**

Về nguyên tắc bộ sinh số A5/1 hoạt động giống như *TinyA5/1*. Kích thước thanh ghi  $X$ ,  $Y$ ,  $Z$  lần lượt là 19, 22 và 23 bit. Các bước quay  $X$ ,  $Y$ ,  $Z$  cụ thể như sau:

- Quay  $X$  gồm các thao tác :
  - $t = x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18}$
  - $x_j = x_{j-1}$  với  $j = 18, 17, 16, 15, \dots, 4, 3, 2, 1$
  - $x_0 = t$
- Quay  $Y$  gồm các thao tác:
  - $t = y_{20} \oplus y_{21}$
  - $y_j = y_{j-1}$  với  $j = 21, 20, \dots, 4, 3, 2, 1$
  - $y_0 = t$
- Quay  $Z$  gồm các thao tác :
  - $t = z_7 \oplus z_{20} \oplus z_{21} \oplus z_{22}$
  - $z_j = z_{j-1}$  với  $j = 22, 21, \dots, 6, 5, 4, 3, 2, 1$
  - $z_0 = t$

Hàm maj tính được trên 3 bit  $x_8$ ;  $y_{10}$ ;  $z_{10}$ . Sau khi xoay xong bit sinh ra là  $s_i = x_8 \oplus y_{10} \oplus z_{10}$ . Toàn bộ quá trình sinh dãy số của A5/1 được minh họa qua hình 2.15:



**Hình 2. 15. Sơ đồ mã hóa giải mã A5/1**

Mã hóa A5/1 có thể được thực hiện dễ dàng bằng các thiết bị phần cứng, tốc độ nhanh. Do đó A5/1 đã từng được sử dụng để mã hóa các dữ liệu trong thời gian thực, vì vậy ngày nay A5/1 được sử dụng để mã hóa dữ liệu cuộc gọi trong mạng điện thoại GSM.

### 2.3.4. Thuật toán mã hóa RC4

Cũng giống như thuật toán mã dòng A5/1, RC4 là một phương pháp mã hóa dòng với kích thước khóa không cố định, được phát triển vào năm 1987 bởi Ron Rivest cho RSA Data Security. Thuật toán RC4 được dùng trong giao thức SSL để bảo mật dữ liệu trong quá trình truyền dữ liệu giữa Web Server và trình duyệt Web. Ngoài ra RC4 còn được sử dụng trong mã hóa WEP của mạng Wireless LAN. Để hiểu rõ hơn về thuật toán mã hóa RC4, trước tiên hãy cùng xét một mô hình thu nhỏ của RC4 gọi là TinyRC4:

#### a. TinyRC4

Khác với TinyA5/1, đơn vị mã hóa của TinyRC4 là 3 bit. TinyRC4 dùng 2 mảng  $S$  và  $T$  mỗi mảng gồm 8 số nguyên 3 bit (từ 0 đến 7). Khóa là một dãy gồm  $N$  số nguyên 3 bit với  $N$  có thể lấy giá trị từ 1 đến 8. Bộ sinh số mỗi lần sinh ra 3 bit để sử dụng trong phép XOR. Quá trình sinh số của TinyRC4 gồm hai giai đoạn:

- Giai đoạn khởi tạo:

---

*/\* Khởi tạo dãy số S và T \*/*

```

for i = 0 to 7 do
  S[i] = i;
  T[i] = K[i mod N];
next i
    
```

*/\* Hoán vị dãy S \*/*

```

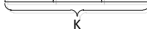
j = 0;
for i = 0 to 7 do
  j = (j + S[i] + T[i]) mod 8;
  Swap(S[i], S[j]);
next i
    
```

---

Dãy  $S$  gồm các số nguyên 3 bit từ 0 đến 7 được sắp xếp thứ tự tăng dần. Sau đó dựa trên các phần tử của khóa  $K$ , các phần tử của  $S$  được hoán vị lẫn nhau đến một mức độ ngẫu nhiên nào đó. Ví dụ: mã hóa thông điệp  $P = 001000110$  (từ “bag”) với khóa  $K$  gồm 3 số 2, 1, 3 ( $N=3$ ):

- Khởi tạo  $S$  và  $T$

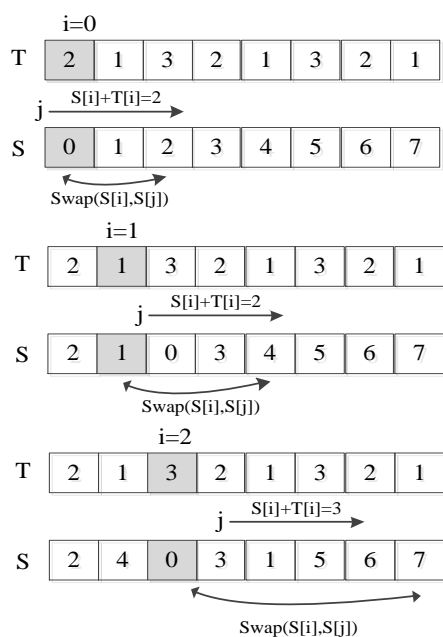
S	0	1	2	3	4	5	6	7
T	2	1	3	2	1	3	2	1


  
 $K$

**Hình 2. 16. Khởi tạo  $S$  và  $T$**

- Hoán vị  $S$





**Hình 2. 17. Quá trình hoán vị S**

Quá trình thực hiện đến khi  $i=7$  và lúc đó dãy S là 6 0 7 1 2 3 5 4

- Giai đoạn sinh số

---

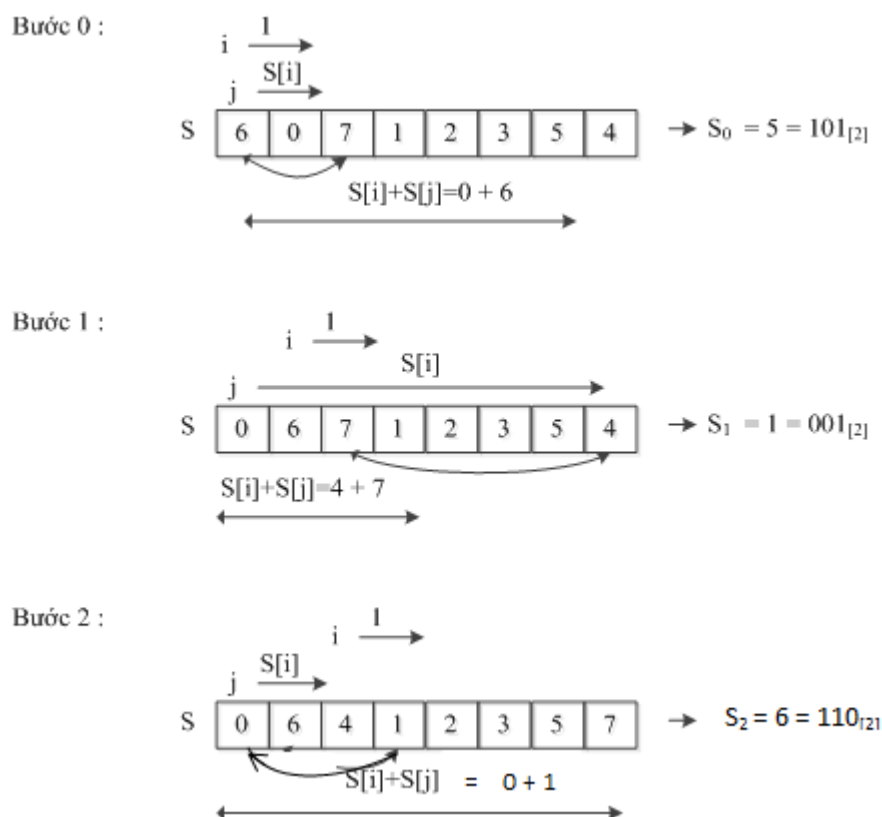
```

i, j = 0;
while (true)
    i = (i + 1) mod 8;
    j = (j + S[i]) mod 8;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 8;
    k = S[t];
end while;
    
```

---

Trong giai đoạn này, các phần tử của S tiếp tục được hoán vị. Tại mỗi bước sinh số, hai phần tử của dãy S được chọn để tính ra số k, 3 bit là số được dùng để XOR với đơn vị mã hóa của thông điệp.

Tiếp tục ví dụ trên, quá trình sinh số mã hóa thông điệp “bag” thực hiện như sau:



Hình 2. 18. Quá trình sinh số mã hóa thông điệp

Vậy bản mã là  $C = 001.000.110 \oplus 101.001.110 = 100.001.000$  (từ EBA)

### b. RC4

Cơ chế hoạt động của *RC4* cũng giống như *TinyRC4* với các đặc tính sau:

- Đơn vị mã hóa của *RC4* là một byte 8 bit.
- Mảng *S* và *T* gồm 256 số nguyên 8 bit.
- Khóa *K* là một dãy gồm *N* số nguyên 8 bit với *N* có thể lấy giá trị từ 1 đến 256.
- Bộ sinh số mỗi lần sinh ra một byte để sử dụng trong phép *XOR*. Hai giai đoạn của

*RC4* là:

- Giai đoạn khởi tạo:

---

**/\* Khởi tạo dãy S và T\***

for  $i = 0$  to 255 do

$S[i] = i$ ;

---

---

```

    T[i] = K[i mod N];
  next i
  /* Hoan vi day S */
  j = 0;
  for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap(S[i], S[j]);
  next i

```

---

- Giai đoạn sinh số :

---

```

i, j = 0;
while (true)
  i = (i + 1) mod 256;
  j = (j + S[i]) mod 256;
  Swap (S[i], S[j]);
  t = (S[i] + S[j]) mod 256;
  k = S[t];
end while;

```

---

Quá trình sinh số của *RC4* cũng sinh ra dãy số ngẫu nhiên, khó đoán trước, vì vậy *RC4* đạt được mức độ an toàn cao. Mã hóa *RC4* hoàn toàn được thực hiện trên các số nguyên một byte do đó tối ưu cho việc thiết lập bằng phần mềm và tốc độ thực hiện nhanh hơn so với mã khối.

## 2.4. ƯU ĐIỂM VÀ NHƯỢC ĐIỂM CỦA MÃ HÓA ĐỐI XỨNG

Như vậy, trên đây đã trình bày chi tiết về một số thuật toán mã hóa đối xứng phổ biến hiện nay. Từ quá trình nghiên cứu về các cơ chế hoạt động của các thuật toán mã hóa này, có thể rút ra một số đánh giá về ưu điểm và nhược điểm của mã hóa đối xứng như sau:

- Ưu điểm:
  - Đơn giản, tức là các yêu cầu về phần cứng không phức tạp;
  - Thời gian tính toán nhanh (tốc độ mã hóa và giải mã nhanh);

- Có tính hiệu quả, thông thường hệ số mở rộng bản tin bằng 1 (số bit đầu ra mã hóa bằng với số bit đầu vào);
- Dễ sử dụng cho các dịch vụ nhạy cảm với độ trễ và các dịch vụ di động.
- Nhược điểm:
  - Yêu cầu phải dùng kênh an toàn để truyền khóa: chi phí cao và việc thiết lập kênh an toàn là khó khăn;
  - Việc tạo khóa và giữ bí mật khóa rất phức tạp;
  - Số lượng khóa cần tạo ra và lưu trữ lớn;
  - Khó khăn trong việc xây dựng các dịch vụ an toàn khác như các dịch vụ đảm bảo tính toàn vẹn dữ liệu, các dịch vụ xác thực và chữ ký số.

## 2.5. CÂU HỎI VÀ BÀI TẬP

- 1) Hãy trình bày đặc điểm chính của mã hóa khóa bí mật?. Vẽ sơ đồ minh họa quy trình mã hóa và giải mã của kỹ thuật mã hóa khóa bí mật?.
- 2) Hãy mô tả các bước mã hóa A5/1?.
- 3) Hãy mô tả các bước mã hóa RC4?.
- 4) Hãy mô tả cấu trúc khối Feistel ?. Hãy vẽ sơ đồ mã hóa theo nguyên tắc cấu trúc khối Feistel ?.
- 5) Hãy mô tả các bước chính của thuật toán mã hóa DES?. Vẽ sơ đồ minh họa về thuật toán mã hóa DES?. Giải thích tại sao thuật toán mã hóa DES lại yêu cầu 16 vòng lặp?.
- 6) Hãy trình bày các ưu điểm và nhược điểm của mã hóa DES?.
- 7) Hãy trình bày cơ sở toán học của thuật toán mã hóa AES?.
- 8) Hãy mô tả về các bước chính của thuật toán mã hóa AES ?. Hãy vẽ sơ đồ minh họa của thuật toán mã hóa AES?.
- 9) Trong thuật toán DES với kích thước khóa đầu vào là 64 bit. Cho khóa sau khi nén xuống 48 bit ở vòng lặp đầu tiên là  $K_1 = 194CD072DE8C_{16}$   
 Bảng nén khóa từ 56 bit  $\rightarrow$  48 bit sau khi tiến hành quá trình dịch bit :

14	17	11	24	1	5
3	25	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Tính giá trị khóa K sau vòng lặp thứ 2?.

10) Tính khóa 48 bit sử dụng cho vòng lặp đầu tiên với dữ liệu đầu vào như sau :

- Khóa 64 bit dưới dạng hexa: **AABB09182736CCDD**.
- Bảng nén khóa từ 64bit → 56 bit :

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

- Bảng nén khóa từ 56 bit → 48 bit sau khi tiến hành quá trình dịch bit :

14	17	11	24	1	5
3	25	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

## CHƯƠNG 3: MÃ HÓA KHÓA BẤT ĐỐI XỨNG

*Chương 3 trình bày một số kiến thức liên quan đến kỹ thuật mã hóa khóa bất đối xứng bao gồm: khái niệm, nguyên tắc mã hóa, giải mã, ưu điểm và nhược điểm. Bên cạnh đó, chương 3 đề cập một số ứng dụng của mã hóa khóa bất đối xứng trong thực tế.*

### 3.1. GIỚI THIỆU VỀ MÃ HÓA BẤT ĐỐI XỨNG

#### 3.1.1. Giới thiệu chung

Mã hóa đối xứng dù rằng đã phát triển từ cổ điển đến hiện đại nhưng vẫn tồn tại một số điểm yếu sau:

- *Vấn đề trao đổi khóa giữa người gửi và người nhận:* Trong quá trình trao đổi khóa cần phải có một kênh an toàn để trao đổi sao cho khóa phải được giữ bí mật chỉ có người gửi và người nhận biết. Chính vì vậy, điều này gây ra một vấn đề rất khó khăn và cần phải có 1 kênh an toàn. Việc thiết lập một kênh an toàn như vậy sẽ tốn kém về mặt chi phí và chậm trễ về mặt thời gian.
- *Tính bí mật của khóa:* Việc khóa bí mật cần trao đổi giữa người gửi và người nhận sẽ dẫn đến trường hợp: khi có sự cố xảy ra thì không có cơ sở quy trách nhiệm nếu khóa bị tiết lộ.

Để khắc phục điểm yếu của mã hóa đối xứng, cần thực hiện 1 số phương án sau:

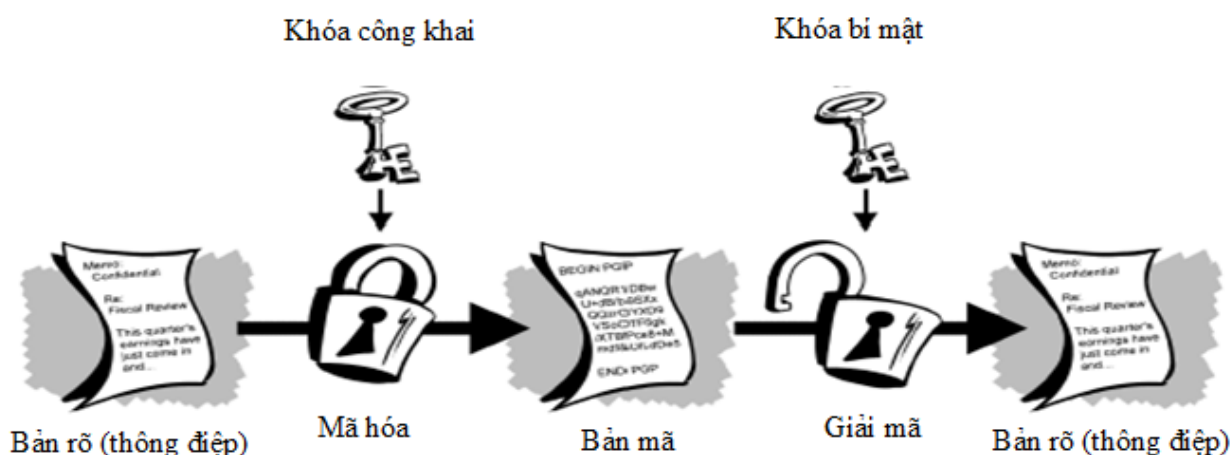
*Phương án 1:* Người nhận (*Bob*) giữ bí mật khóa  $K_2$ , còn khóa  $K_1$  thì công khai cho tất cả mọi người biết. *Alice* muốn gửi dữ liệu cho *Bob* thì dùng khóa  $K_1$  để mã hóa. *Bob* dùng  $K_2$  để giải mã. Ở đây *Trudy* cũng biết khóa  $K_1$ , tuy nhiên không thể dùng chính  $K_1$  để giải mã mà phải dùng  $K_2$ . Do đó chỉ có duy nhất *Bob* mới có thể giải mã được. Điều này bảo đảm tính bí mật của quá trình truyền dữ liệu. Ưu điểm của phương án này là không cần phải truyền khóa  $K_1$  trên kênh an toàn.

*Phương án 2:* Người gửi (*Alice*) giữ bí mật khóa  $K_1$ , còn khóa  $K_2$  thì công khai cho tất cả mọi người biết. *Alice* muốn gửi dữ liệu cho *Bob* thì dùng khóa  $K_1$  để mã hóa. *Bob* dùng  $K_2$  để giải mã. Ở đây *Trudy* cũng biết khóa  $K_2$  nên *Trudy* cũng có thể giải mã được. Do đó phương án này không đảm bảo tính bí mật. Vì vậy nếu kết hợp phương án 1 và

phương án 2, thì mô hình đề xuất sẽ khắc phục được các nhược điểm của mã hóa đối xứng.

Vào năm 1976 *Whitfield Diffie* và *Martin Hellman* đã tìm ra một phương pháp mã hóa khác mà có thể giải quyết được hai vấn đề trên, đó là mã hóa khóa công khai hay còn gọi là mã hóa bất đối xứng. Đây có thể xem là một bước đột phá quan trọng nhất trong lĩnh vực mã hóa.

Mã hóa bất đối xứng là các phương pháp mà quy trình mã hóa và giải mã sử dụng các khóa khác nhau. Khóa dùng để mã hóa là khóa công khai và khóa dùng để giải mã là khóa bí mật. Hình 3.1 thể hiện quy trình mã hóa và giải mã của mô hình mã hóa khóa công khai.



Hình 3. 1. Mô hình mã hóa bất đối xứng

Trong cả hai phương án, một khóa được giữ bí mật và chỉ một người biết, còn khóa kia được công khai. Để thuận tiện, trong bài giảng quy ước lại các ký hiệu như sau:

- Khóa bí mật được gọi là khóa riêng và ký hiệu là  $K_R$ .
- Khóa công khai (*khóa công khai*) được ký hiệu là  $K_U$ .
- Thông điệp được ký hiệu là  $M$ , còn bản mã giữ nguyên ký hiệu là  $C$
- Phương án 1 viết lại theo công thức 3.1 và 3.2 :

$$C = E(M, K_U) \quad (3.1)$$

$$M = D(C, K_R) \quad (3.2)$$

- Phương án 2 viết lại theo công thức 3.3 và 3.4:

$$C = E(M, K_R) \quad (3.3)$$

$$M = D(C, K_U) \quad (3.4)$$

Vậy giữa khóa riêng và khóa công khai có mối quan hệ với nhau như thế nào?. Chúng độc lập hay phụ thuộc vào nhau?. Trong thuật toán mã hóa khóa bất đối xứng thì dĩ nhiên là hai khóa  $K_U$  và  $K_R$  không thể nào hoàn toàn độc lập với nhau. Phải có một mối quan hệ nào đó giữa  $K_U$  và  $K_R$  thì mới có thể tiến hành mã hóa và giải mã được. Có nghĩa là  $K_R = f(K_U)$ . Tuy nhiên một yêu cầu rất quan trọng là việc tính  $K_R = f(K_U)$  phải là bất khả thi về mặt thời gian. Nếu nguyên tắc này bị vi phạm thì việc giữ bí mật khóa  $K_R$  không còn ý nghĩa vì từ khóa công khai  $K_U$  có thể tính được khóa riêng  $K_R$ .

Để có được cặp khóa  $K_R$  và  $K_U$  như trên, người ta thường dùng các hàm một chiều (*oneway function*). Các hàm một chiều có tính chất là hàm phi nghịch đảo vì vậy rất khó thực hiện. Sau đây là ví dụ về hàm một chiều: việc sinh ra hai số nguyên tố lớn  $p, q$  và tính tích  $N = pq$  thì thực hiện dễ dàng. Tuy nhiên nếu chỉ cho trước  $N$  và thực hiện phân tích  $N$  để tìm lại hai số nguyên tố  $p, q$  là việc hoàn toàn bất khả thi về mặt thời gian. Đây chính là cơ sở để xây dựng thuật toán mã hóa bất đối xứng *RSA*. Trong mục 3.2 sẽ trình bày chi tiết về vấn đề này.

### 3.1.2. Một số thuật toán mã hóa bất đối xứng

Các nghiên cứu trên thế giới từ năm 1976 cho đến nay đã đưa ra được một số bài toán một chiều như sau:

- Bài toán logarit rời rạc (Hệ mật Omura-Massey, Elgama...);
- Bài toán phân tích thừa số (gắn với hệ mật nổi tiếng *RSA*);
- Bài toán xếp ba lô (Hệ mật Merkle – Hellman; Hệ mật Chor-Rivest);
- Bài toán mã sửa sai (Hệ mật McEliece...);
- Bài toán xây dựng hệ mật trên đường cong Elliptic.

Trong bài giảng này, chỉ tập trung vào tìm hiểu thuật toán *RSA*. Các thuật toán khác có thể đọc trong phần tài liệu tham khảo.



### 3.2. THUẬT TOÁN MÃ HÓA RSA

Thuật toán mã hóa *RSA* là một thuật toán mã hóa khóa công khai. Thuật toán *RSA* được xây dựng bởi các tác giả *Ron Rivest*, *Adi Shamir* và *Len Adleman* tại học viện *MIT* vào năm 1977, và ngày nay đang được sử dụng rộng rãi. Về mặt tổng quát thuật toán *RSA* là một phương pháp mã hóa theo khối. Trong đó thông điệp  $M$  và bản mã  $C$  là các số nguyên từ 0 đến  $2^i$  với  $i$  số bit của khối. Kích thước thường dùng của  $i$  là 1024 bit. Thuật toán *RSA* sử dụng hàm một chiều là vấn đề phân tích một số thành thừa số nguyên tố và bài toán Logarith rời rạc.

#### Nguyên tắc thực hiện quá trình mã hóa và giải mã thuật toán RSA :

##### ✓ Quá trình mã hóa:

Để thực hiện mã hóa và giải mã, thuật toán *RSA* dùng phép lũy thừa modulo của lý thuyết số. Các bước thực hiện như sau:

- Chọn hai số nguyên tố lớn  $p$  và  $q$  và tính  $N = pq$ . Cần chọn  $p$  và  $q$  sao cho:

$$M < 2^{i-1} < N < 2^i.$$

- Tính  $\Phi(n) = (p - 1)(q - 1)$
- Tìm một số  $e$  sao cho:  $\{e \text{ và } \Phi(n) \text{ là 2 số cùng nhau và } 0 < e < \Phi(n)\}$
- Tìm một số  $d$  sao cho:  $e.d \equiv 1 \pmod{\Phi(n)}$  (hay:  $d = e^{-1} \pmod{\Phi(n)}$ )

( $d$  là nghịch đảo của  $e$  trong phép modulo  $\Phi(n)$ )

- Hủy bỏ  $n$ ,  $p$  và  $q$ . Chọn khóa công khai  $K_U$  là cặp  $(e, N)$ , khóa riêng  $K_R$  là cặp  $(d, N)$
- Việc mã hóa thực hiện theo công thức:

- Theo phương án 1, mã hóa:  $C = E(M, K_U) = M^e \pmod{N}$  (3.5)

- Theo phương án 2, mã hóa chứng thực:  $C = E(M, K_R) = M^d \pmod{N}$  (3.6)

##### ✓ Quá trình giải mã: Việc giải mã thực hiện theo công thức:

- Theo phương án 1, giải mã:  $M = D(C, K_R) = C^d \pmod{N}$  (3.7)

- Theo phương án 2, mã hóa chứng thực:  $M = D(C, K_U) = C^e \pmod{N}$  (3.8)

Thông điệp  $M$  có kích thước  $i-1$  bit, bản mã  $C$  có kích thước  $i$  bit.

**Tính đúng của thuật toán RSA:**

Trên đây, bài giảng đã mô tả quy trình mã hóa và giải mã thuật toán RSA. Tiếp theo bài giảng sẽ trình bày tính đúng đắn của thuật toán RSA. Giả sử bản mã chính là thông điệp  $\overline{M} = M$ . Xét phương án 1:

- Từ bước 4 suy ra được  $e.d = k.n + 1$  với  $k$  là một số nào đó.

$$\begin{aligned} \text{- Vậy } \overline{M} &= C^d \bmod N \\ &= M^{ed} \bmod N \\ &= M^{kn+1} \bmod N \\ &= M^{k(p-1)(q-1)+1} \bmod N \end{aligned}$$

Điều này có nghĩa là RSA đúng khi và chỉ khi  $M^{k(p-1)(q-1)+1} \bmod N \equiv M \bmod p$ . Xét hai khả năng của  $M$ :

- $M$  chia hết cho  $p$  khi đó  $M \bmod p = 0$  do đó  $M^{k(p-1)(q-1)+1} \equiv M \equiv 0 \bmod p$
- $M$  không chia hết cho  $p$ : Vì  $p$  là số nguyên tố nên  $M$  và  $p$  là hai số nguyên tố cùng nhau. Vậy :  $M^{k(p-1)(q-1)+1} \bmod p = M.(M^{p-1})^{(q-1)k} \bmod p = M.1^{k(q-1)} \bmod p$  (theo định lý Fermat)  $= M \bmod p$

Vậy :  $M^{k(p-1)(q-1)+1} - M \equiv 0 \bmod p \quad \forall M$ . Hay nói cách khác  $M^{k(p-1)(q-1)+1} - M$  chia hết cho  $p$ . Vì  $q, p$  là hai số nguyên tố nên suy ra được  $M^{k(p-1)(q-1)+1} - M$  chia hết cho  $N = p.q$ . Như vậy:

$$\begin{aligned} M^{k(p-1)(q-1)+1} &\equiv M \bmod N \\ \Rightarrow \overline{M} &= M^{k(p-1)(q-1)+1} \bmod N = M \text{ (do } M < N \text{)} \end{aligned}$$

Vì  $e$  và  $d$  đối xứng nên có thể thấy trong phương án 2, cũng có  $\overline{M} = M$ .

- Không thể suy ra  $K_R$  từ  $K_U$ , nghĩa là tìm cặp  $(d, N)$  từ cặp  $(e, N)$ : Có  $e$  và  $N$ , muốn tìm  $d$ , phải dựa vào công thức:  $e.d \equiv 1 \bmod n$ . Do đó phải tính được  $n$ .

Vì  $\Phi(n) = (p-1)(q-1)$  nên suy ra phải tính được  $p$  và  $q$ . Vì  $N = pq$  nên chỉ có thể tính được  $p$  và  $q$  từ  $N$ . Tuy nhiên điều này là bất khả thi vì  $N = pq$  là hàm một chiều. Vậy không thể tính được  $K_R$  từ  $K_U$

**Ví dụ RSA :**

- Chọn  $p = 11, q = 3$  do đó  $N = p.q = 33 = 100001_2$
- $\Phi(n) = (q-1)(p-1) = 20$
- Chọn  $e = 3$  nguyên tố cùng nhau với  $\Phi(n) = 20$ .
- Tính nghịch đảo của  $e$  trong phép modulo  $n$  được  $d = 7 (3.7 = 21)$
- Khóa công khai  $K_U = (e, N) = (3, 33)$ .
- Khóa bí mật  $K_R = (d, N) = (7, 33)$ .

Với mã hóa bảo mật:

- Mã hóa thông điệp  $M = 17$   

$$C = M^e \bmod N = 17^3 \bmod 33 = 29$$
- Giải mã bản mã  $C = 29$   

$$\overline{M} = C^d \bmod N = 29^7 \bmod 33 = 17 = M$$

Với mã hóa chứng thực:

- Mã hóa thông điệp  $M = 17$   

$$C = M^d \bmod N = 17^7 \bmod 33 = 8$$
- Giải mã bản mã  $C = 8$   

$$\overline{M} = C^e \bmod N = 8^3 \bmod 33 = 17 = M$$

**Độ phức tạp trong tính toán RSA:**

Có hai vấn đề về độ phức tạp tính toán trong phương pháp mã hóa RSA. Đó là các phép tính sinh khóa và các phép tính mã hóa/giải mã.

- *Phép tính mã hóa và giải mã:* Phép tính mã hóa và giải mã dùng phép lũy thừa modulo. Để an toàn, thường phải chọn  $N, e, d, M$  lớn. Tuy nhiên số học modulo có một tính chất sau:

$$(a \times b) \equiv [(a \bmod n) \times (b \bmod n)] \bmod n$$

Có thể sử dụng tính chất này để đơn giản phép tính lũy thừa modulo thông qua một phương pháp gọi là “bình phương liên tiếp”.

Ví dụ cần tính  $x^{16} \bmod n$  thì cần thực hiện các bước sau:

đầu tiên tính  $a = x \bmod n$ ,

tiếp theo tính  $b = x^2 \bmod n = a^2 \bmod n$ ,

tiếp theo tính  $c = x^4 \bmod n = b^2 \bmod n$ ,

tiếp theo tính  $d = x^8 \bmod n = c^2 \bmod n$ ,

cuối cùng có  $x^{16} \bmod n = d^2 \bmod n$ . Các số  $a, b, c, d$  luôn nhỏ hơn  $n$  do đó tránh được việc tính số lũy thừa lớn đồng thời nâng cao tốc độ tính toán.

Trong trường hợp tổng quát để tính  $x^b \bmod n$ , viết  $b$  dưới dạng số nhị phân:

$$b = b_k b_{k-1} \dots b_2 b_1 b_0 \text{ trong đó } b_i \text{ là các bit } 0, 1.$$

Như vậy  $b = \sum_{b_i \neq 0} 2^i$ . Do đó  $x^b = \prod_{b_i \neq 0} x^{2^i} \bmod n$

Như vậy ứng với các  $b_i = 1$ , dùng phương pháp bình phương liên tiếp để tính các  $x^{2^i}$ . Sau đó nhân các kết quả với nhau và rút gọn modulo để có kết quả cuối cùng. Để tăng tốc độ quá trình mã hóa dữ liệu, người ta thường chọn một khóa công khai  $e$  cụ thể nào đó. Thường thì  $e$  là  $65537 (2^{16} + 1)$ .

- *Phép tính sinh khóa:*

Phép tính sinh khóa là chọn  $p$  và  $q$  nguyên tố để tính  $N$ . Để phân tích số  $N$  thành tích hai thừa số nguyên tố  $p, q$ , chỉ có một cách duy nhất là thử từng số  $p$  và  $q$ . Do đó phải chọn  $p, q$  đủ lớn để việc thử là không khả thi. Dựa vào lý thuyết số nguyên tố, người ta ước tính rằng cần thử trung bình khoảng 70 số lẻ để tìm ra một số nguyên tố lớn chừng  $2^{200}$ . Vì đã chọn  $e$  trước là 65537 (hay 3 hoặc 17 ...), do đó cần kiểm tra xem  $e$  có nguyên tố cùng nhau với  $\Phi(n) = (p-1)(q-1)$  hay không. Nếu không phải thử lại với  $p$  và  $q$  khác. Sau khi đã tìm  $p$  và  $q$  thích hợp, cần tìm  $d$  sao cho  $e.d \equiv 1 \bmod \Phi(n)$ .

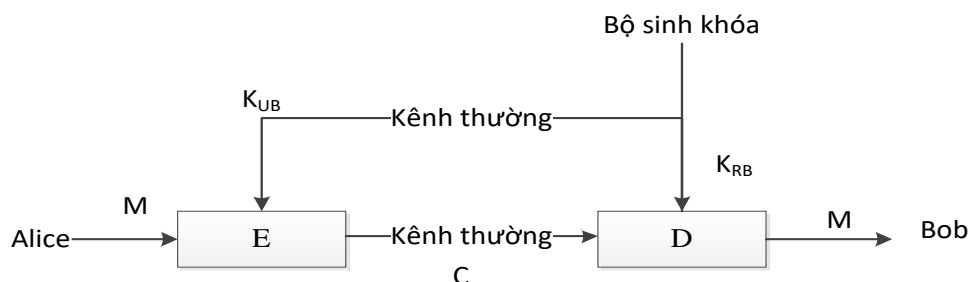
### 3.3. ƯU ĐIỂM, NHƯỢC ĐIỂM CỦA MÃ HÓA BẤT ĐỐI XỨNG

*Xét tính Chứng thực trong mã hóa bất đối xứng:* Giả sử Alice muốn gửi dữ liệu cho Bob dùng mã hóa khóa công khai, trước tiên Alice và Bob sẽ chọn cặp khóa riêng và khóa công khai. Ký hiệu khóa riêng và khóa công khai của Alice là  $K_{RA}$  và  $K_{UA}$ , của Bob là  $K_{RB}$

và  $K_{UB}$ . Như vậy để gửi dữ liệu cho *Bob*, *Alice* sẽ dùng phương án 1: mã hóa dữ liệu bằng khóa công khai  $K_{UB}$  của *Bob*, và *Bob* dùng khóa riêng  $K_{RB}$  để giải mã :

$$C = E(M, K_{UB}) \quad (3.9)$$

$$M = D(C, K_{RB})$$

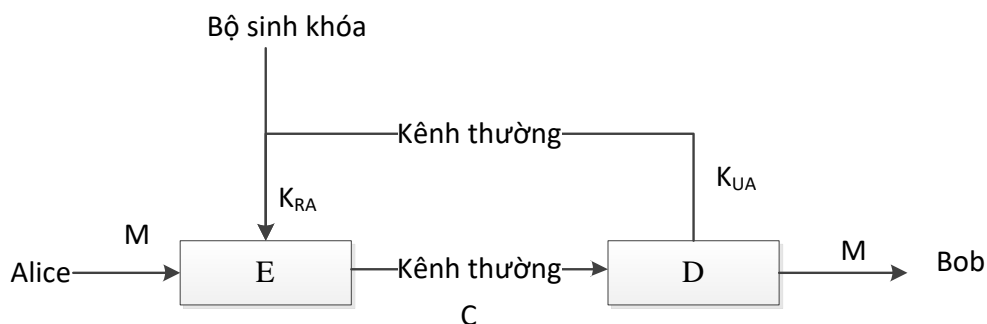


**Hình 3. 2. Mô hình bảo mật**

Để đảm bảo tính chứng thực và *Alice* không chối bỏ trách nhiệm gửi dữ liệu, *Alice* sẽ dùng phương án 2: *Alice* mã hóa dữ liệu bằng khóa riêng  $K_{RA}$ , và *Bob* dùng khóa công khai  $K_{UA}$  của *Alice* để giải mã.

$$C = E(M, K_{RA})$$

$$M = D(C, K_{UA}) \quad (3.10)$$



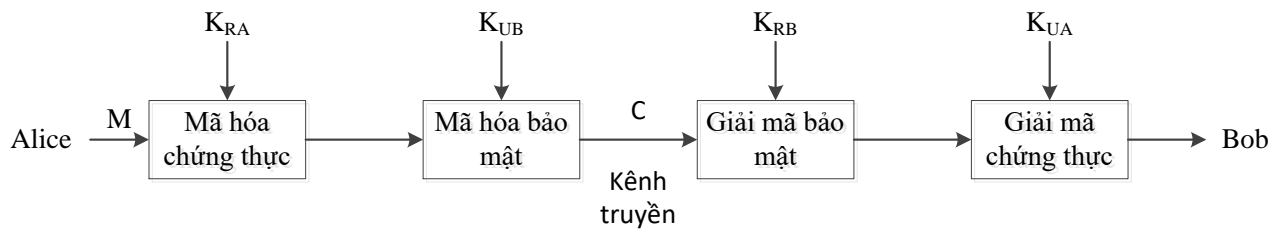
**Hình 3. 3. Mô hình xác thực**

Cũng với giả định rằng thông điệp có ý nghĩa là một dãy bit có cấu trúc, nếu bản giải mã của *Bob* là hợp lệ thì rõ ràng là *Alice* là người gửi vì chỉ có *Alice* mới có khóa riêng  $K_{RA}$ . Giống như mã hóa đối xứng, nếu *Trudy* (kẻ tấn công) can thiệp chỉnh sửa trên bản mã  $C$  thì *Bob* sẽ giải mã ra thông điệp là một dãy bit vô nghĩa. Còn nếu *Trudy* có được khóa  $K_{RA}$  thì *Alice* không thể thoái thác trách nhiệm làm lộ khóa.

Tuy nhiên mô hình như trên lại không đảm bảo tính bí mật. Vì không chỉ riêng *Bob*, *Trudy* cũng biết được khóa công khai  $K_{UA}$  của *Alice*. Do đó *Trudy* có thể giải mã bản mã  $C$  và biết được nội dung thông điệp  $M$ . Để giải quyết vấn đề trên, người kết hợp tính bí mật, chứng thực và không từ chối qua hình 3.4.

$$C = E(E(M, K_{RA}), K_{UB}) \quad (3.11)$$

$$M = D(D(C, K_{RB}), K_{UA})$$



**Hình 3. 4. Mô hình chứng thực và không chối bỏ**

Từ những phân tích trên, cũng như kết hợp với đặc điểm của thuật toán mã hóa khóa bất đối xứng, có thể thấy một số ưu điểm và nhược điểm của mã hóa bất đối xứng như sau:

- Ưu điểm:
  - Kích thước khóa lớn, độ an toàn cao;
  - Việc quản lý và phân phối khóa dễ dàng hơn so với mã hóa đối xứng.
- Nhược điểm:
  - Tốc độ mã hóa, giải mã chậm hơn so với mã hóa đối xứng.

### 3.4. ỨNG DỤNG CỦA MÃ HÓA BẤT ĐỐI XỨNG

Ứng dụng cơ bản của các thuật toán mã hóa bất đối xứng nói chung bao gồm:

- Bí mật trong truyền tin (Confidentiality);
- Chứng thực;
- Kết hợp tính bí mật và tin cậy.

Ứng dụng thực tế của các thuật toán mã hóa bất đối xứng:

- Dùng để vận chuyển và bảo mật khóa bí mật cho mã hóa đối xứng;
- Mã hóa email hoặc xác thực người gửi email (OpenPGP hay S/MIME);

- Mã hóa hoặc nhận thực văn bản (Các tiêu chuẩn chữ ký XML\* hoặc mã hóa XML\* khi văn bản được thể hiện dưới dạng XML);
- Xác thực người dùng ứng dụng (Đăng nhập bằng thẻ thông minh; nhận thực người dùng trong SSL);
- Ứng dụng trong chữ ký số, chứng chỉ số.

### 3.5. CÂU HỎI VÀ BÀI TẬP

- 1) Hãy trình bày đặc điểm chính của thuật toán mã hóa khóa công khai?. Vẽ sơ đồ minh họa quy trình mã hóa và giải mã của thuật toán mã hóa khóa công khai?.
- 2) Hãy trình bày về chứng thực trong mô hình mã hóa bất đối xứng?.
- 3) Hãy nêu các yêu cầu cơ bản trong thủ tục sinh khóa trong thuật toán mã hóa RSA?. Hãy phân tích các yêu cầu cơ bản đó?.
- 4) Hãy trình bày về quá trình mã hóa và giải mã của thuật toán RSA?. Hãy lấy ví dụ minh họa về quá trình mã hóa và giải mã của thuật toán RSA?.
- 5) Hãy phân tích mức độ an toàn của thuật toán RSA?.
- 6) Hãy trình bày về ưu nhược điểm của mã hóa bất đối xứng?.
- 7) Hãy nêu một vài ứng dụng của mã hóa bất đối xứng?.

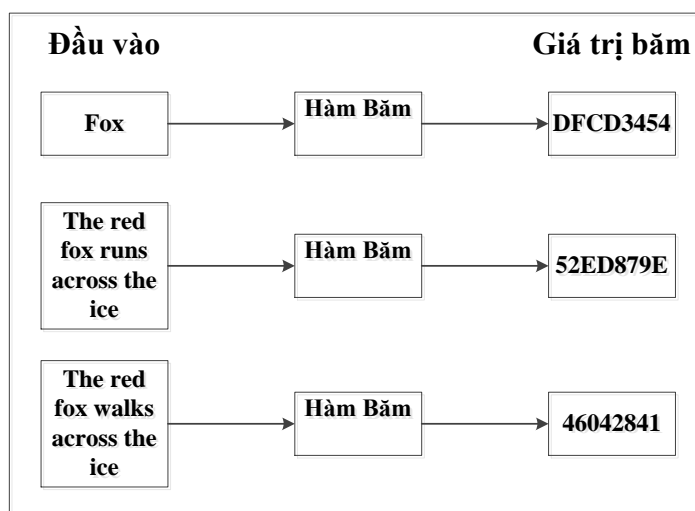
## CHƯƠNG 4: CÁC HÀM BẮM

*Chương này cung cấp các kiến thức liên quan đến hàm băm bao gồm: định nghĩa, khái niệm, tính chất, vai trò của hàm băm. Ngoài ra, chương 4 còn trình bày về một số hàm băm phổ biến hiện nay cũng như nguyên tắc làm việc của một số hàm băm đó.*

### 4.1. GIỚI THIỆU VỀ HÀM BẮM

#### 4.1.1. Định nghĩa

Hàm băm là hàm chuyển đổi một thông điệp (chuỗi bit) có độ dài bất kỳ hữu hạn thành một dãy bit có độ dài cố định  $n$  bit ( $n > 0$ ). Dãy bit đầu ra của hàm băm được gọi là giá trị băm hay mã băm.



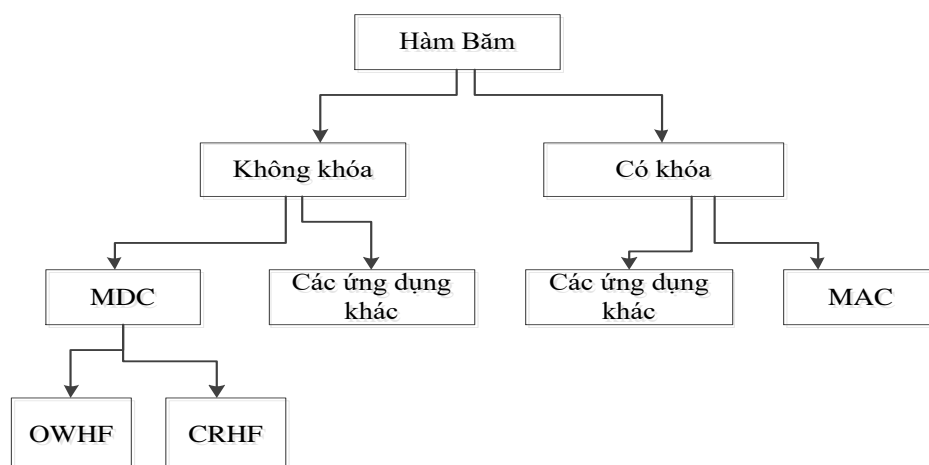
Hình 4. 1. Ví dụ về hàm băm

Hình 4.1. trình bày về ví dụ của hàm băm.

#### 4.1.2. Phân loại

Cũng giống như mã hóa, việc phân loại hàm băm có nhiều cách khác nhau. Cơ chế của việc phân loại này dựa trên các tính chất hoặc theo khóa. Hình 4.2 trình bày một cách để phân loại hàm băm.





**Hình 4. 2. Phân loại hàm băm**

Theo đó, một số cách để phân loại hàm băm như sau:

*Cách 1. Theo khóa sử dụng:* hàm băm được chia thành 2 loại:

- Hàm băm không khóa (unkeyed): đầu vào chỉ là thông điệp  $x$  với hàm băm  $h$ ,
- Hàm băm có khóa (keyed): đầu vào gồm thông điệp và khóa bí mật (theo dạng  $h(x, K)$ , với hàm băm  $h$  và thông điệp  $x$  và  $K$  là khóa bí mật).

*Cách 2. Theo chức năng:* có thể chia các hàm băm thành 2 loại chính:

- Mã phát hiện sửa đổi (MDC - Modification Detection Code): MDC thường được sử dụng để tạo chuỗi đại diện cho thông điệp và dùng kết hợp với các kỹ thuật khác như chữ ký số để đảm bảo tính toàn vẹn của thông điệp. MDC thuộc loại hàm băm không khóa. MDC gồm 2 loại nhỏ:

- Hàm băm một chiều (OWHF - One-way hash functions): Với hàm băm một chiều, việc tính giá trị băm là dễ dàng, nhưng việc khôi phục thông điệp từ giá trị băm là rất khó khăn;
- Hàm băm chống đụng độ (CRHF - Collision resistant hash functions): Với hàm băm chống đụng độ, sẽ là rất khó để tìm được 2 thông điệp khác nhau nhưng có cùng giá trị băm.

- Mã xác thực thông điệp (MAC - Message Authentication Code): MAC cũng được dùng để đảm bảo tính toàn vẹn của thông điệp mà không cần một kỹ thuật bổ sung nào khác. MAC là loại hàm băm có khóa.

#### 4.1.3. Các tính chất cơ bản

Như đã trình bày ở trên, hàm băm được phân thành 2 loại là hàm băm có khóa và hàm băm không có khóa. Như vậy, khi xét đến tính chất của hàm băm cũng cần phân loại giữa hàm băm có khóa và hàm băm không khóa. Dưới đây sẽ trình bày về các tính chất cơ bản của hàm băm có khóa và hàm băm không khóa

#### 4.1.3.1. Tính chất của hàm băm không khóa

Ngoài hai tính chất cơ bản của hàm băm đã trình bày ở trên, hàm băm không khóa còn có các tính chất sau:

- **Tính khó tính toán nghịch ảnh:** Với bất kỳ giá trị băm  $h$ , không thể tính được  $x$  sao cho  $H(x)=h$ . Hay  $H$  được gọi là **hàm một chiều**.
- **Tính bền xung đột yếu (weak collision resistance):** với bất kỳ giá trị  $x$ , không thể tính được  $y \neq x$  sao cho  $H(y) = H(x)$ .
- **Tính bền xung đột mạnh (strong collision resistance):** Không thể tính được một cặp  $(x, y)$  sao cho  $H(x) = H(y)$ .
- **Kháng tiền ảnh (Pre-image resistance):** Với một mã băm  $h$  bất kỳ, khó tìm được một thông điệp  $m$  nào mà  $h=hash(m)$ . Trong góc độ hàm số toán học, mã băm là ảnh còn thông điệp là tạo ảnh của mã băm, hay gọi là tiền ảnh. Sức kháng cự tấn công từ ảnh ngược về tiền ảnh gọi là kháng tiền ảnh. Một hàm băm có kháng tiền ảnh yếu là lỗ hổng cho các cuộc tấn công tiền ảnh.
- **Kháng tiền ảnh thứ hai (Second pre-image resistance)** Với một thông điệp  $m1$  bất kỳ, khó tìm được một thông điệp thứ hai  $m2$  sao cho  $m1 \neq m2$  và  $hash(m1) = hash(m2)$ . Xác suất xảy ra biến cố có thông điệp  $m2$  như thế tương tự biến cố “Cùng ngày sinh như bạn”. Một hàm băm có kháng tiền ảnh thứ hai yếu là lỗ hổng cho các cuộc tấn công tiền ảnh thứ hai.

*Lưu ý:* trên đây bài giảng đã trình bày các tính chất cơ bản của hàm băm không khóa. Tuy nhiên, cần chú ý rằng: trong hàm băm không khóa tồn tại Hàm băm một chiều và Hàm băm chống đụng độ. Trong bài giảng đã tổng hợp các tính chất của cả 2 loại hàm băm này. Chi tiết hơn về các tính chất của hàm băm một chiều và hàm băm chống đụng độ có thể tham khảo trong tài liệu tham khảo của bài giảng.

### 4.1.3.2. Tính chất của hàm băm có khóa

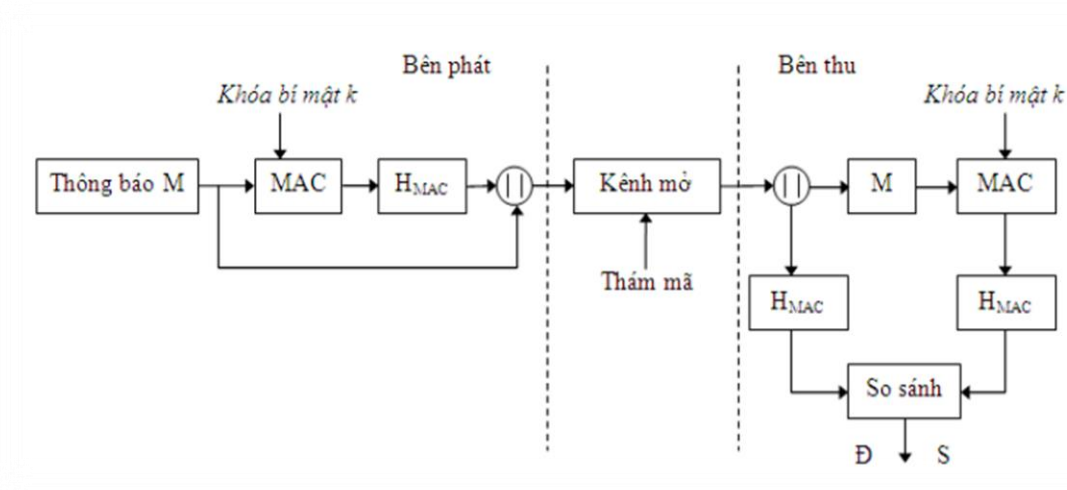
Thuật toán MAC là một họ các hàm  $H_k$  (được tham số hoá bằng một khoá bí mật  $k$ ) có các tính chất sau:

- *Tính chất nén:* với  $H_k$  đã biết và giá trị  $k$  cho trước và một đầu vào  $x$ , thì dễ dàng tính toán được  $H_k(x)$  ( $H_k(x)$  được gọi là giá trị MAC).
- *Tính chất dễ dàng tính toán:*  $H_k$  ánh xạ một đầu vào  $x$  có độ dài bit hữu hạn thì dễ dàng tính được đầu ra  $H_k(x)$  có độ dài bit  $n$  cố định
- *Tính khó tính toán:* Với các cặp giá trị đầu vào là  $x$  và  $x_i$  với  $x \neq x_i$  thì không có khả năng tìm được cặp  $H_k(x)$  và  $H_k(x_i)$  thỏa mãn  $H_k(x) = H_k(x_i)$ . Nếu tính chất này không được thỏa mãn thì thông điệp bị coi là giả mạo.

### 4.1.4. Vai trò

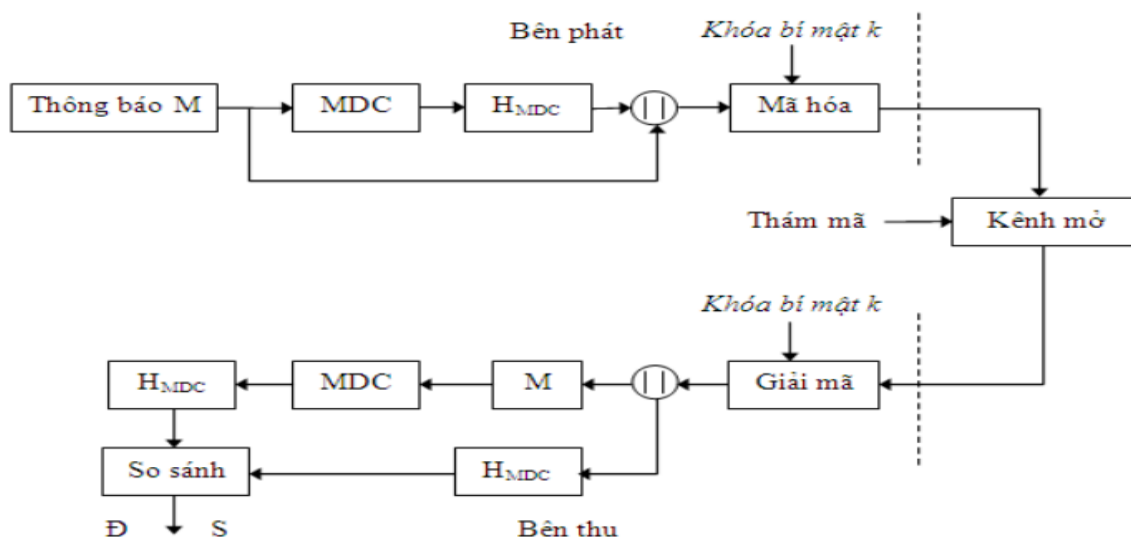
Vai trò và ứng dụng của hàm băm trong thực tế rất lớn. Ngoài việc ứng dụng của hàm băm nhằm đảm bảo các thuộc tính của an toàn thông tin thì hàm băm còn có vai trò là xác định tính toàn vẹn của dữ liệu và xác thực thông điệp. Tính toàn vẹn của dữ liệu và xác thực thông điệp là tính chất đảm bảo dữ liệu không bị sửa đổi một cách bất hợp pháp kể từ khi dữ liệu được tạo ra, được trao đổi hoặc được lưu giữ bởi một nguồn được xác định. Có ba phương pháp cung cấp tính toàn vẹn của dữ liệu bằng cách dùng các hàm băm.

- Chỉ dùng MAC



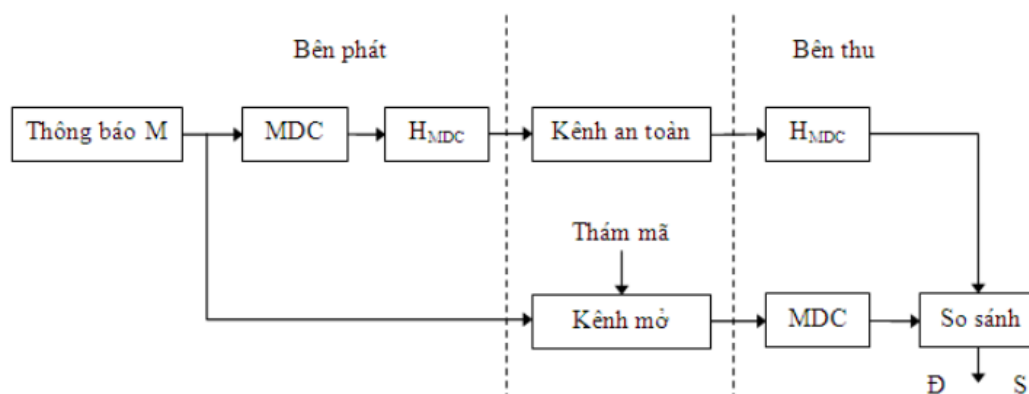
Hình 4. 3. Sơ đồ kiểm tra toàn vẹn chỉ dùng MAC

- Dùng *MDC* và mã hóa



Hình 4. 4. Sơ đồ kiểm tra toàn vẹn dùng *MDC* và mã hóa

- Sử dụng *MDC* và kênh an toàn



Hình 4. 5. Sơ đồ kiểm tra tính toàn vẹn dùng *MDC* và kênh an toàn

## 4.2. CÁC HÀM BẮM KHÔNG KHÓA

### Định nghĩa 4.2.1:

Mã hóa khối  $(n, r)$  là một mã hóa khối xác định một hàm khả nghịch từ các thông điệp  $n$  bit sang các bản mã  $n$  bit bằng cách sử dụng một khóa  $k$  ( $r$  bit). Nếu  $E$  là một phép mã hóa như vậy thì ký hiệu  $E_k(x)$  cho phép mã hóa  $x$  bằng khóa  $k$ .

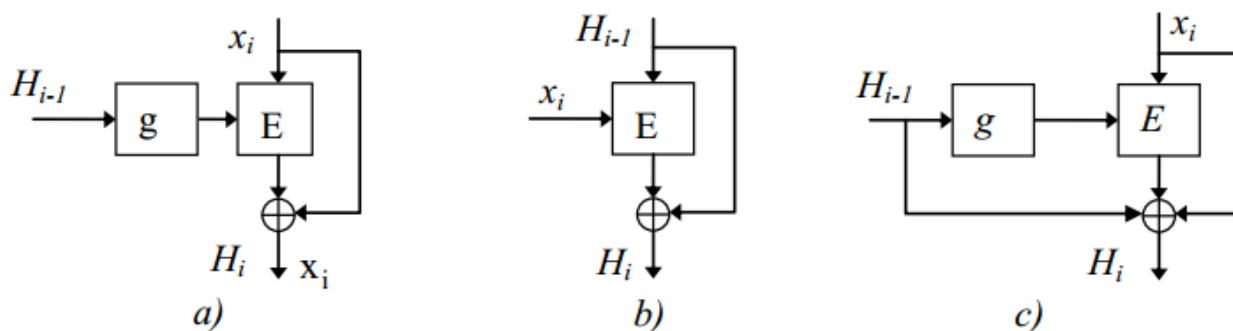
Định nghĩa 4.2.2:

Cho  $h$  là một hàm băm có cấu trúc lặp được xây dựng từ một mã hóa khối với hàm nén thực hiện  $s$  phép mã hóa khối để xử lý từng khối thông điệp  $n$  bit. Khi đó tốc độ của  $h$  là  $1/s$ .

**4.2.1. MDC độ dài đơn**

Các sơ đồ (xem hình 4.6) có liên quan chặt chẽ với các hàm băm độ dài đơn, được xây dựng trên các mã hóa khối. Các sơ đồ này có sử dụng các thành phần được xác định trước như sau:

- Một mã hóa khối  $n$  bit khởi sinh  $E_k$  được tham số hoá bằng một khoá đối xứng  $k$ .
- Một hàm  $g$  ánh xạ  $n$  bit vào thành khoá  $k$  sử dụng cho  $E$  (Nếu các khoá cho  $E$  cũng có độ dài  $n$  thì  $g$  có thể là hàm đồng nhất).
- Một giá trị khởi tạo thích hợp  $H_0$  thích hợp để dùng với  $E$ .



**Hình 4. 6. Các sơ đồ hàm băm MDC đơn**

Trong đó: a) *Matyas-Mayer-Oseas*; b) *Davies-Mayer*; c) *Miyaguchi – Preneel*.

**a) Thuật toán băm *Matyas - Mayer - Oseas*.**

- Input : Chuỗi đầu vào  $x$
- Output : Chuỗi đầu ra sau khi băm có kích thước  $n$  bit.
- Mô tả thuật toán :
  - Đầu vào chuỗi  $x$  được chia thành các khối  $n$  bit. Nếu  $length(x) \div n \neq 0$  thêm vào một số bit cần thiết để tạo thành khối hoàn chỉnh. Thu được  $m$  khối  $n$  bit:  $x_1..x_n$ . Phải xác định trước một giá trị khởi tạo  $H_0$ .

- Đầu ra là  $H_m$  được xác định như sau :

$$H_i = E_{g(H_{i-1})}(x_i) \oplus x_i \quad (1 \leq i \leq m) \quad (4.1)$$

**b) Thuật toán băm Davies – Mayer**

- Input : Chuỗi đầu vào  $x$
- Output : Chuỗi đầu ra sau khi băm có kích thước  $n$  bit.
- Mô tả thuật toán :
  - Đầu vào chuỗi  $x$  được chia thành các khối  $n$  bit. Nếu  $\text{length}(x) \bmod n \neq 0$  thêm vào một số bit cần thiết để tạo thành khối hoàn chỉnh. Thu được  $m$  khối  $n$  bit:  $x_1..x_m$ . Phải xác định trước một giá trị khởi tạo  $H_0$ .
  - Đầu ra là  $H_m$  được xác định như sau :

$$H_i = E_{x_i}(H_{i-1})(x_i) \oplus x_i \quad (1 \leq i \leq m) \quad (4.2)$$

**c) Thuật toán băm Miyaguchi – Preneel**

- Input : Chuỗi đầu vào  $x$
- Output : Chuỗi đầu ra sau khi băm có kích thước  $n$  bit.
- Mô tả thuật toán:

Sơ đồ này tương tự như sơ đồ **Matyas - Mayer - Oseas** ngoại trừ  $H_{i-1}$  (đầu ra ở giai đoạn trước) được cộng  $\text{mod } 2$  với tín hiệu ra ở giai đoạn hiện thời. Như vậy:

$$H_i = E_{g(H_{i-1})}(x_i) \oplus x_i \oplus H_{i-1} \quad (1 \leq i \leq m). \quad (4.3)$$

- Nhận xét: Sơ đồ **Davies - Mayer** có thể coi là sơ đồ đối ngẫu với sơ đồ **Miyaguchi – Preneel** theo nghĩa  $x_i$  và  $H_{i-1}$  đổi lẫn vai trò.

**4.2.2. MDC độ dài kép: MDC-2 và MDC-4**

**MDC-2** và **MDC-4** là các mã phát hiện sự sửa đổi yêu cầu tương ứng là 2 và 4 phép toán mã hóa khối trên mỗi khối đầu vào hàm băm. Sử dụng 2 hoặc 4 phép lặp của sơ đồ **Matyas - Mayer – Oseas** để tạo ra hàm băm có độ dài kép. Khi dùng **DES** sẽ tạo ra mã băm 128 bit. Tuy nhiên trong cấu trúc tổng quát có thể dùng các hệ mật mã khối khác **MDC-2** và **MDC-4** sử dụng các thành phần xác định như sau:

- **DES** được dùng làm mật mã khối  $E_k$  có đầu vào/ra 64 bit và được tham số hoá bằng khoá  $k$  56 bit.

- Hai hàm  $g$  và ánh xạ các giá trị 64 bit  $U$  thành các khoá  $DES$  56 bit như sau:

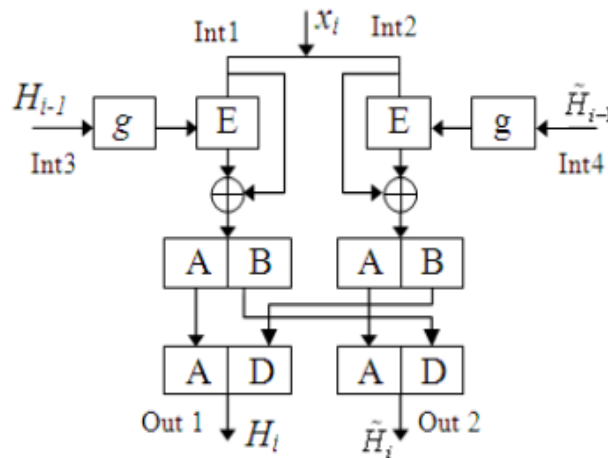
Cho  $U = u_1u_2 \dots u_{64}$ , xoá mọi bit thứ 8 và đặt các bit thứ 2 và thứ 3 về "10" đối với  $g$  và "01" đối với  $g$ .

$$g(U) = u_110u_4 u_5 u_6 u_7 u_8 u_9 u_{10} \dots u_{63}$$

$$g(U) = u_101u_4 u_5 u_6 u_7 u_8 u_9 u_{10} \dots u_{63}$$

Đồng thời điều này cũng phải đảm bảo rằng chúng không phải là các khoá  $DES$  yếu hoặc nửa yếu vì các khoá loại này có bit thứ hai bằng bit thứ ba. Ngoài ra nó cũng đảm bảo yêu cầu bảo mật là  $g(H_0) \neq g(U)$ .

Hình 4.7 mô tả thuật toán  $MDC-2$ .



Hình 4. 7. Sơ đồ thuật toán  $MDC-2$

### Thuật toán $MDC-2$

- Input : Chuỗi đầu vào  $x$  có độ dài  $r = 64t$  với  $t \geq 2$
- Output : Chuỗi đầu ra sau khi băm có kích thước 128 bit.
- Mô tả thuật toán:
  - Phân xâu  $x$  ra làm các khối 64 bit  $x_i : x_1 \dots x_{64}$
  - Chọn các hàng số đầu vào  $H_0$  từ một tập giá trị định trước (dạng  $HEXA$ ).

$$H_0 = 0x5252525252525252$$

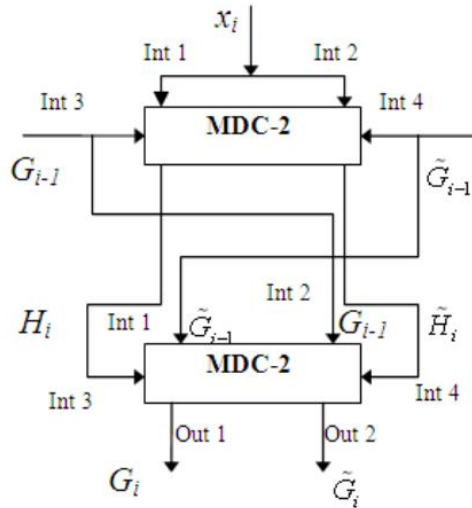
$$H_0 = 0x2525252525252525$$

- Ký hiệu  $\parallel$  là phép ghép và  $C_i^L, C_i^R$  là các nửa 32bit phải và trái của  $C_i$

Đầu ra  $h(x) = H_i // H_t$  được xác định như sau ( $1 \leq i \leq t$ )

$$k_i = g(H_{i-1}), C_i = E_{k_i}(x_i) \oplus_i, H_i = C_i^L // C_i^R \quad (4.4)$$

**Hình 4.8 mô tả thuật toán MDC-4:**



**Hình 4. 8. Sơ đồ thuật toán MDC-4**

### 4.3. CÁC HÀM BẮM CÓ KHÓA

Các hàm băm có khoá được sử dụng để xác thực thông điệp và thường được gọi là các thuật toán tạo mã xác thực thông điệp (MAC). MAC dựa trên các mật mã khối. Nguyên tắc làm việc của MAC như sau:

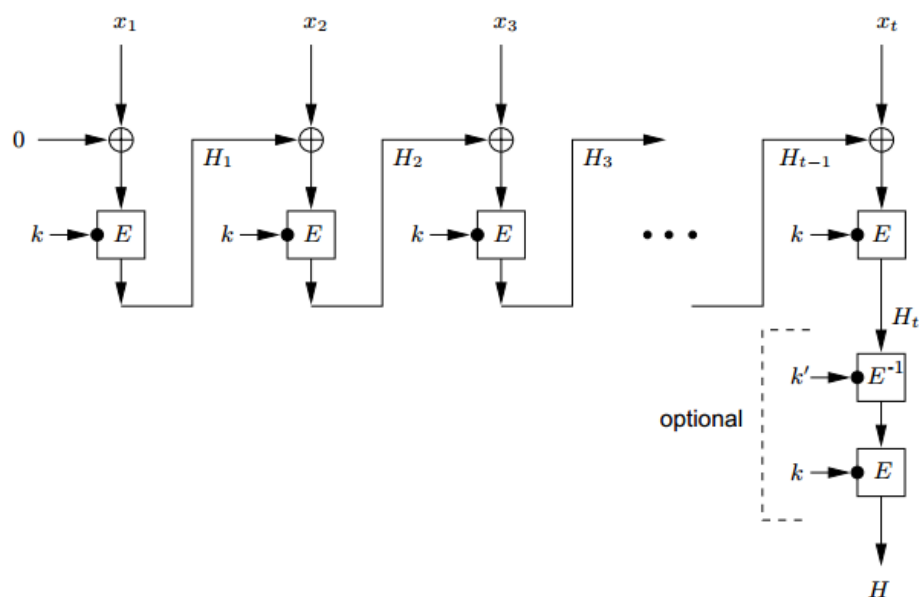
#### Thuật toán MAC :

- Input : Dữ liệu đầu vào  $x$ , hàm mã hóa khối  $E$ , khóa MAC bí mật  $k$  của  $E$ .
- Output :  $n$  bit MAC trên  $x$  ( $n$  là độ dài khối của  $E$ )
- Mô tả thuật toán (xem hình 4.9):
  - Đệm và chia khối: Đệm thêm các bit vào  $x$  sao cho dữ liệu đầu vào chia thành từng khối  $n$  bit :  $x_1, x_2, \dots, x_t$ .
  - Xử lý theo chế độ CBC
  - Ký hiệu  $E_k$  là phép mã hóa  $E$  với khóa  $k$
  - Tính khối  $H_t$  như sau :

$$\checkmark H_1 \leftarrow E_k(x_1)$$



- ✓  $H_i \leftarrow E_k(H_{i-1} \oplus x_i)$  với  $2 \leq i \leq t$
- Xử lý tăng thêm sức mạnh của MAC
  - ✓ Dùng một khóa bí mật  $k' \neq k$ . Tính  $H_t' \leftarrow E_{k'}^{-1}(H_t)$ ,  $H_t = E_k(H_t')$
- Hoàn thành. Hàm MAC là khối  $n$  bit  $H_t$



Hình 4. 9. Sơ đồ hàm băm MAC

#### 4.4. MỘT SỐ HÀM BẮM THÔNG DỤNG

Bảng 4.1 liệt kê một số hàm băm thông dụng. Để tìm hiểu các hàm băm này, có thể nghiên cứu các tài liệu tham khảo. Trong bài giảng này, chỉ tập trung vào phân tích 2 hàm băm chính là hàm băm họ MD và SHA.

Bảng 4. 1. Các hàm băm thông dụng

Thuật toán	Kích thước đầu ra	Kích thước trạng thái trong	Kích thước khối	Độ dài	Kích thước word	Xung đột
HAVAL	256/224/192/160/128	256	1024	64	32	Có
MD2	128	384	128	Không	8	Khả năng lớn
MD4	128	128	512	64	32	Có

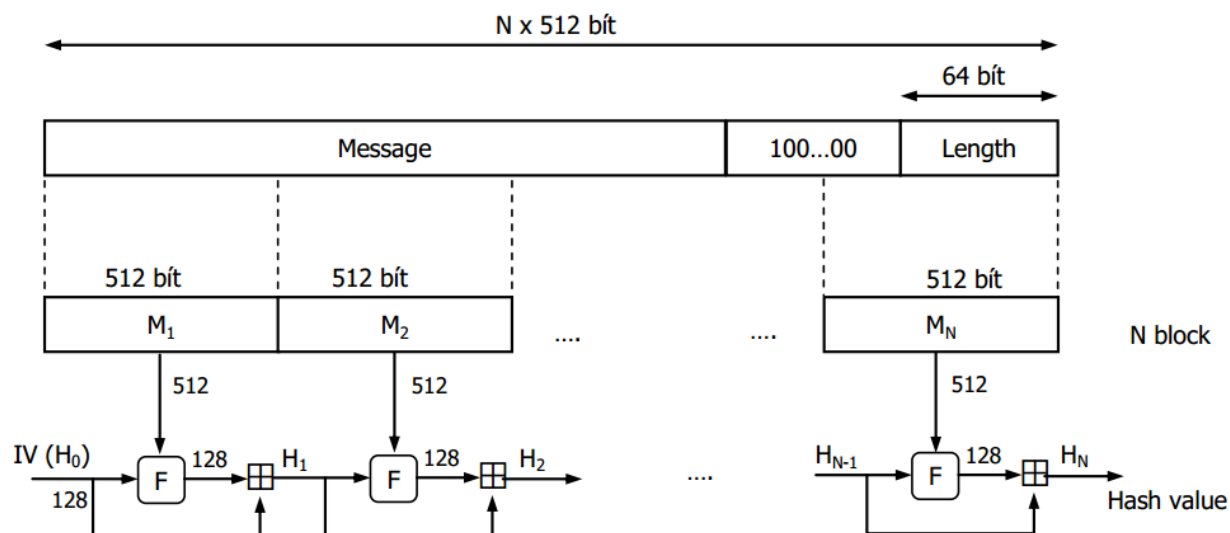
MD5	128	128	512	64	32	Có
PANAMA	256	8736	256	No	32	Có lỗi
RIPEMD	128	128	512	64	32	Có
RIPEMD-128/256	128/256	128/256	512	64	32	Không
RIPEMD-160/320	160/320	160/320	512	64	32	Không
SHA-0	160	160	512	64	32	Không
SHA-1	160	160	512	64	32	Có lỗi
SHA-256/224	256/224	256	512	64	32	Không
SHA-512/384	512/384	512	1024	128	64	Không
Tiger(2)-192/160/128	192/160/128	192	512	64	64	Không
VEST-4/8 (hash mode)	160/256	256/384	8	80/128	1	Không
VEST-16/32 (hash mode)	320/512	512/768	8	160/256	1	Không
WHIRLPOOL	512	512	512	256	8	Không

#### 4.4.1. Hàm băm họ MD - Message Digest (MD5)

*MD5* được phát minh bởi *Ron Rivest*, người đã tham gia xây dựng *RSA*. *MD5*, viết tắt từ chữ *Message Digest*, được phát triển lên từ *MD4* và trước đó là *MD2*, do *MD2* và *MD4* không còn an toàn. Kích thước giá trị băm của *MD5* là 128 bit. Trong phần này sẽ trình bày về hàm băm *MD5* với kích thước giá trị băm là 128 bit, được sử dụng để tính giá trị của thông điệp có kích thước tối đa  $2^{64}$  bit.

##### Thuật toán *MD5*:

- Input: xâu đầu vào x có độ dài tối đa  $2^{64}$  bit.
- Output: chuỗi băm 128 bit.
- Sơ đồ thuật toán :



Hình 4. 10. Sơ đồ hàm băm MD5

Quy trình băm của hàm băm MD5 theo hình 4.10 được trình bày như sau:

- Trước tiên thông điệp được đệm vào dãy padding 100...00. Chiều dài của dãy padding được chọn sao cho thông điệp cuối cùng có thể chia làm  $N$  block 512 bit  $M_1, M_2, \dots, M_N$ . Quá trình tính giá trị băm của thông điệp là quá trình lũy tiến. Trước tiên block  $M_1$  kết hợp với giá trị khởi tạo  $H_0$  thông qua hàm  $F$  để tính giá trị hash  $H_1$ . Sau đó block  $M_2$  được kết hợp với  $H_1$  để cho ra giá trị hash là  $H_2$ . Block  $M_3$  kết hợp với  $H_2$  cho ra giá trị  $H_3$ . Cứ như vậy cho đến block  $M_N$  thì có giá trị băm của toàn bộ thông điệp là  $H_N$  (xem hình 4.11).

-  $H_0$  là một dãy 128 bit được chia thành 4 từ 32 bit, ký hiệu 4 từ 32 bit trên là  $abcd$ . Với  $a, b, c, d$  là các hằng số như sau (viết dưới dạng thập lục phân):

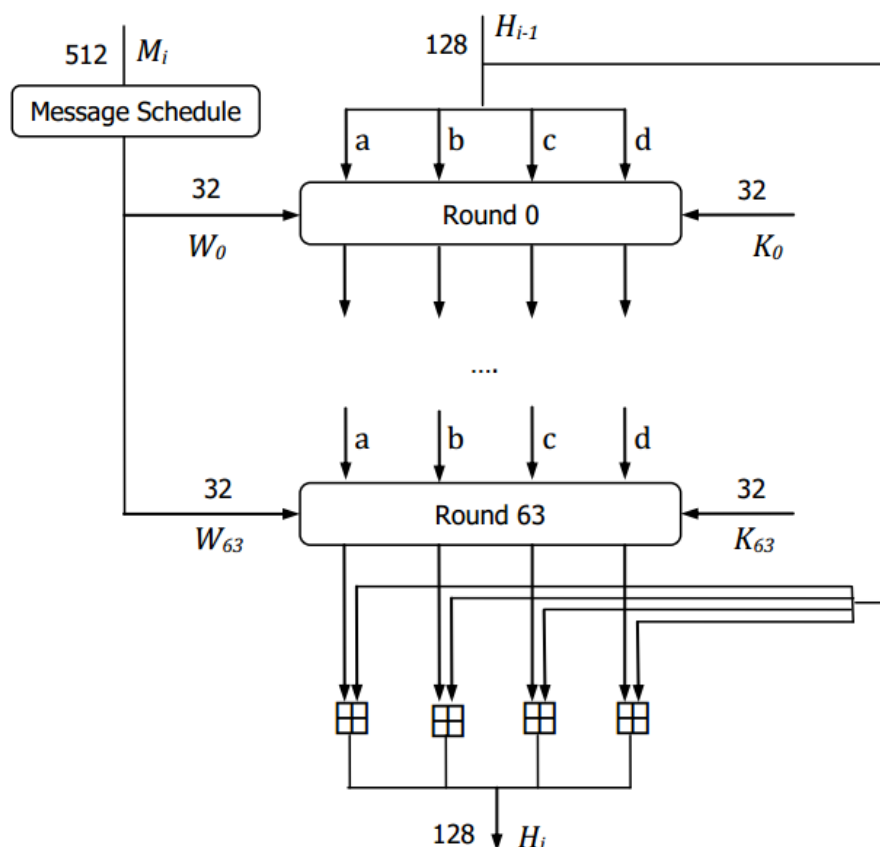
$$a = 01234567$$

$$b = 89abcdef$$

$$c = fedbca98$$

$$d = 76543210$$

Cấu trúc của hàm  $F$  như sau:



**Hình 4. 11. Sơ đồ cấu trúc hàm  $F$**

Tại mỗi bước lũy tiến, các giá trị  $abcd$  của giá trị hash  $H_{i-1}$  được biến đổi qua 64 vòng từ 0 đến 63. Tại vòng thứ  $j$  sẽ có 2 tham số là  $K_j$  và  $W_j$  đều có kích thước 32 bit. Các tham số  $K_j$  được tính từ công thức:  $K_j$  là phần nguyên của số  $2^{32} \text{abs}(\sin(i))$  với  $i$  biểu diễn theo  $rad$ .

Giá trị block  $M_i$  512 bit được biến đổi qua một hàm message schedule cho ra 64 giá trị  $W_0, W_1, \dots, W_{63}$  mỗi giá trị 32 bit. Block  $M_i$  512 bit được chia thành 16 block 32 bit ứng với các giá trị  $W_0, W_1, \dots, W_{15}$  ( $16 \times 32 = 512$ ). Tiếp theo, 16 giá trị này được lặp lại 3 lần tạo thành dãy 64 giá trị.

Sau vòng cuối cùng, các giá trị  $abcde$  được cộng với các giá trị  $abcd$  của  $H_{i-1}$  để cho ra các giá trị  $abcd$  của  $H_i$ . Phép cộng ở đây là phép cộng modulo  $2^{32}$ .

Tiếp theo tìm hiểu cấu trúc của một vòng. Việc biến đổi các giá trị  $abcd$  trong vòng thứ  $i$  được thể hiện trong hình bên dưới.

Ở đây  $b \rightarrow c$ ,  $c \rightarrow d$ ,  $d \rightarrow a$ .

Giá trị  $b$  được tính qua hàm:

$$t = a + f(b, c, d) + W_i + K_i$$

$$b = b + ROTL(t, s)$$

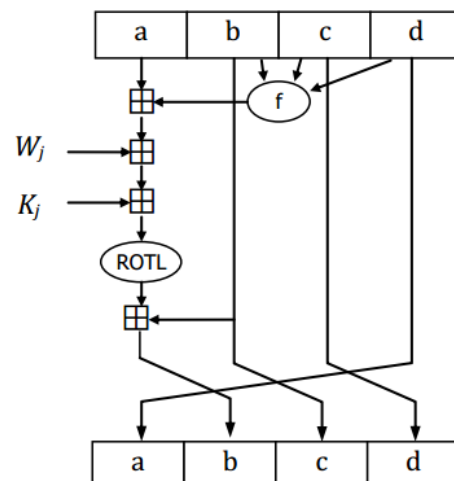
Trong đó : Hàm  $f(x, y, z)$ :

$$f(x, y, z) = (x \wedge y) \vee (\neg x \wedge z) \text{ nếu là vòng 0 đến 15}$$

$$f(x, y, z) = (z \wedge x) \vee (\neg z \wedge y) \text{ nếu là vòng 16 đến 32}$$

$$f(x, y, z) = x \oplus y \oplus z \text{ nếu là vòng 32 đến 48}$$

$$f(x, y, z) = y \oplus (x \vee \neg z) \text{ nếu là vòng 49 đến 63}$$



Hình 4. 12. Cấu trúc biến đổi 1 vòng MD5

Hàm  $ROTL(t, s)$ :  $t$  được dịch vòng trái  $s$  bit, với  $s$  là các hằng số cho vòng thứ  $i$  như sau:

Bảng 4. 2. Bảng tính giá trị  $ROTL(t, s)$

$i$	$s$	$i$	$s$
0, 4, 8, 12	7	32, 36, 40, 44	4
1, 5, 9, 13	12	33, 37, 41, 45	11
2, 6, 10, 14	17	34, 38, 42, 46	16
3, 7, 11, 15	22	35, 39, 43, 47	23
16, 20, 24, 28	5	48, 52, 56, 60	6
17, 21, 25, 29	9	49, 53, 57, 61	10
18, 22, 26, 30	14	50, 54, 58, 62	15
19, 23, 27, 31	20	51, 55, 59, 63	21

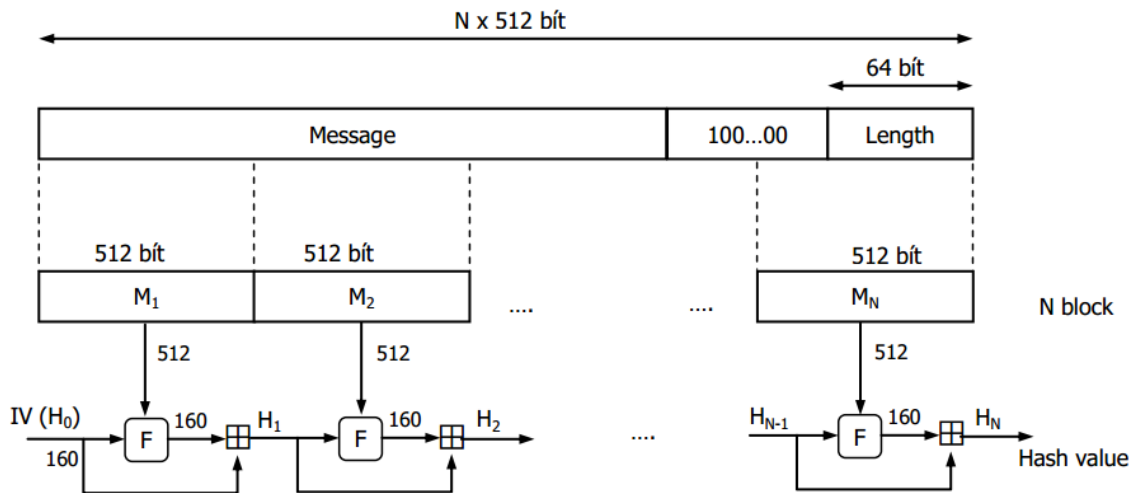
Phép  $+$  ( hay  $\boxplus$ ) là phép cộng modul  $2^{32}$

#### 4.4.2. Hàm băm họ SHA - Secure Hash Algorithm (SHA-1)

Hàm băm MD5 từ lúc ra đời đã có nhiều chuyên gia nhận định là không an toàn. Đặc biệt, vào năm 1994 và 1998, một phương pháp tấn công MD5 đã được thực hiện thành công và một số thông điệp có cùng giá trị băm MD5 được chỉ ra (2 thông điệp có cùng bản băm). Vì MD5 không còn được xem là an toàn, nên người ta đã xây dựng thuật

toán băm khác. Một trong những thuật toán đó là *SHA1* (*Secure Hash Algorithm*) mà đã được chính phủ Mỹ chọn làm chuẩn quốc gia.

Hàm băm *SHA-1* tạo ra các giá trị băm có kích thước là 160 bit, kích thước đầu vào của *SHA-1* là thông điệp có kích thước tối đa là  $2^{64}$  bit. Sơ đồ tổng thể của *SHA-1* cũng giống như của *MD5* (xem hình 4.13).



#### Hình 4. 13. Sơ đồ thuật toán băm SHA-1

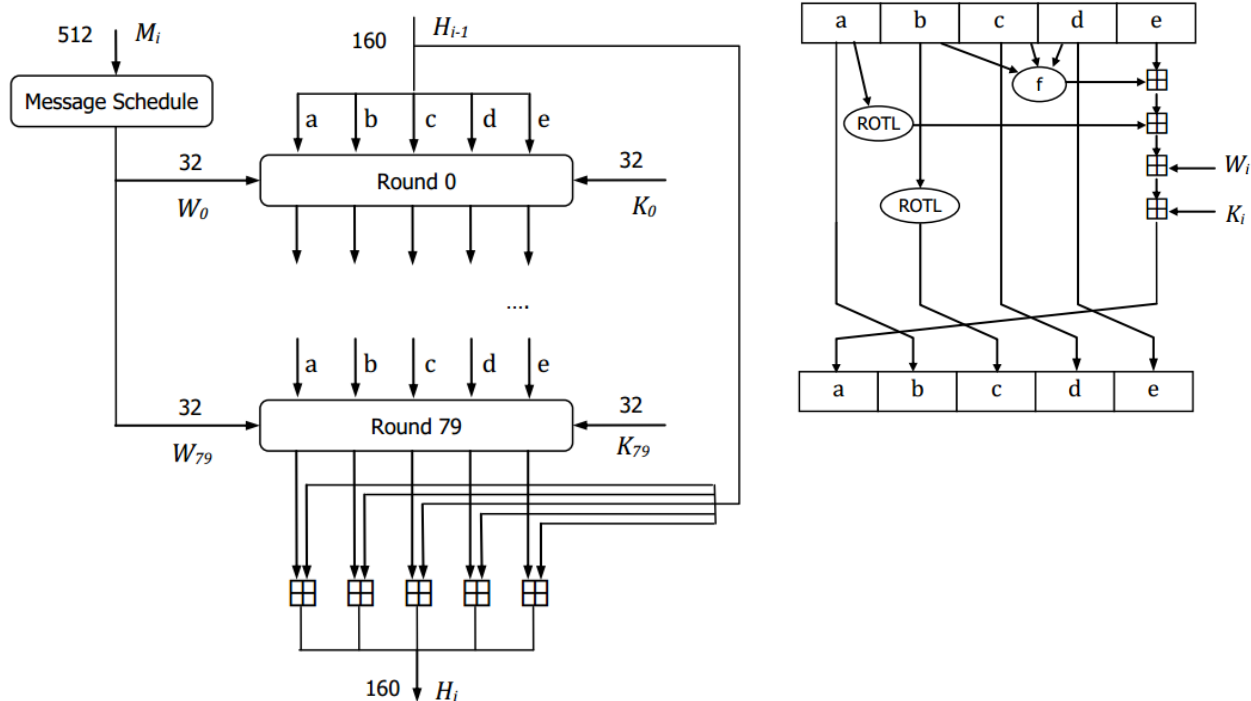
Từ sơ đồ thuật toán 4.13 cho thấy quy trình làm việc của hàm băm *SHA1* như sau:

$H_0$  là một dãy 160 bit được chia thành 5 từ 32 bit, ký hiệu 5 từ 32 bit trên là  $abcde$ .

Với  $a, b, c, d, e$  là các hằng số như sau:

$$a = 67452301 ; b = efcdab89 ; c = 98badcfe ; d = 10325476 ; e = c3d2e1f0$$

Cấu trúc của hàm  $F$  của  $SHA$  cũng tương tự như  $MD5$ , tuy nhiên được thực hiện trên 80 vòng (xem hình 4.14).



Hình 4. 14. Cấu trúc hàm F của thuật toán băm SHA-1

Giá trị  $K_0, K_1, \dots, K_{79}$  là các hằng số sau:

$$K_i = 5A827999 \quad \text{với} \quad 0 \leq i \leq 19$$

$$K_i = 6ED9EBA1 \quad \text{với} \quad 20 \leq i \leq 39$$

$$K_i = 8F1BBCDC \quad \text{với} \quad 40 \leq i \leq 59$$

$$K_i = CA62C1D6 \quad \text{với} \quad 60 \leq i \leq 79$$

Giá trị block  $M_i$  512 bit được biến đổi qua một hàm message schedule cho ra 80 giá trị  $W_0, W_1, \dots, W_{79}$  mỗi giá trị 32 bit, theo quy tắc:

- Trước tiên block  $M_i$  512 bit được chia thành 16 block 32 bit ứng với các giá trị  $W_0, W_1, \dots, W_{15}$  ( $16 \times 32 = 512$ ).

- Các giá trị  $W_t$  ( $16 \leq t \leq 79$ ) được tính theo công thức:

$$W_t = \text{ROTL}(W_{t-3} + W_{t-8} + W_{t-14} + W_{t-16}, 1) \text{ với phép cộng modulo } 2^{32}.$$

- Việc biến đổi các giá trị  $abcde$  trong vòng thứ  $i$  được thể hiện trong hình bên trên.

Ở đây  $a \rightarrow b, c \rightarrow d, d \rightarrow e$ . Giá trị  $a$  và  $c$  được tính qua các hàm:

$$a = \text{ROTL}(a, 5) + f(b, c, d) + e + W_i + K_i$$

$$c = \text{ROTL}(b, 30)$$

Trong đó hàm  $f(x,y,z)$  được tính :

$$f(x, y, z) = Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z) \quad \text{nếu là vòng 0 đến 19}$$

$$f(x, y, z) = Parity(x, y, z) = x \oplus y \oplus z \quad \text{nếu là vòng 20 đến 39}$$

$$f(x, y, z) = Maj(x, y, z) = (x \wedge y) \oplus (y \wedge z) \oplus (z \wedge x) \quad \text{nếu là vòng 40 đến 59}$$

$$f(x, y, z) = Parity(x, y, z) = x \oplus y \oplus z \quad \text{nếu là vòng 60 đến 79}$$

Ý nghĩa của hàm Maj và hàm Ch:

- Hàm Maj: giả sử  $x_i, y_i, z_i$  là bit thứ  $i$  của  $x, y, z$ , thì bit thứ  $i$  của hàm Maj là giá trị nào chiếm đa số, 0 hay 1 (giống như hàm maj được định nghĩa trong phần thuật toán A5/1).

- Hàm Ch: bit thứ  $i$  của hàm Ch là phép chọn: *if  $x_i$  then  $y_i$  else  $z_i$ .*

#### 4.5. CÂU HỎI VÀ BÀI TẬP

- 1) Hãy nêu định nghĩa về hàm băm?
- 2) Hãy trình bày các tính chất của hàm băm có khóa và hàm băm không khóa?
- 3) Hãy trình bày về các phương pháp phân loại hàm băm?
- 4) Hãy mô tả thuật toán MDC-2?
- 5) Hãy mô tả thuật toán MDC-4?
- 6) Hãy mô tả thuật toán MAC?
- 7) Hãy trình bày về hàm băm MD5?. Hãy vẽ sơ đồ minh họa về hàm băm MD5?.
- 8) Hãy trình bày về hàm băm SHA-1?. Hãy vẽ sơ đồ minh họa về hàm băm SHA-1?.



## CHƯƠNG 5: QUẢN LÝ KHÓA

*Chương này cung cấp các kiến thức liên quan đến vấn đề quản lý và phân phối khóa bao gồm: các khái niệm, phân loại, vai trò... Ngoài ra, chương 5 còn mô tả quy trình quản lý và phân phối khóa của một số giao thức phổ biến hiện nay.*

### 5.1. GIỚI THIỆU

Trong quá trình trao đổi thông tin, ngoài việc lựa chọn một thuật toán mã hóa đáng tin cậy, cần phải quan tâm đến vấn đề làm thế nào có thể trao đổi, chia sẻ các khóa dùng trong mã hóa và giải mã một cách an toàn. Trước hết ta bắt đầu với một số khái niệm trong phân phối và quản lý khóa:

- Quản lý khóa (Key management): là một tập các kỹ thuật cho phép thiết lập và duy trì các quan hệ khóa giữa các bên có thẩm quyền.
- Quan hệ khóa (Keying relationship): là trạng thái mà trong đó các bên tham gia truyền thông chia sẻ các dữ liệu chia sẻ (thường là khóa hoặc thành phần tạo ra khóa) để sử dụng cho các kỹ thuật mã hóa. Các dữ liệu chia sẻ có thể là:

- Khóa bí mật;
- Khóa công khai;
- Các giá trị khởi tạo;
- Các tham số bổ sung không bí mật.

Quản lý khóa cung cấp các kỹ thuật và thủ tục cần thiết, cho phép các bên có thẩm quyền thực hiện một số quá trình sau:

- Khởi tạo các người dùng hệ thống (system users) trong một vùng (domain);
- Sinh khóa, phân phối và cài đặt các dữ liệu khóa;
- Kiểm soát việc sử dụng các dữ liệu khóa;
- Cập nhật, thu hồi và hủy các dữ liệu khóa;
- Lưu, sao lưu/khôi phục và lưu trữ các dữ liệu khóa.

Trong quá trình trao đổi thông tin, có rất nhiều loại khóa khác nhau được sử dụng phù hợp với từng đối tượng và từng mục đích. Chính vì thế, cần phân biệt rõ các loại khóa

này để hiểu và sử dụng chúng cho phù hợp. Có nhiều phương pháp phân loại khóa, dưới đây trình bày 2 phương pháp, đó là: theo khả năng sử dụng và theo thời gian sử dụng.

- Phân loại các lớp khóa theo khả năng sử dụng gồm 3 loại khóa chính:
  - Khóa chủ (Master key): Là các khóa ở mức cao nhất và không được bảo vệ bằng các kỹ thuật mật mã. Các khóa chủ thường được chuyển giao trực tiếp và được bảo vệ bằng các cơ chế kiểm soát vật lý.
  - Khóa dùng cho trao đổi khóa (Key – encrypting keys): Là những khóa được sử dụng để vận chuyển hoặc lưu trữ các khóa khác. Các khóa này cũng có thể được bảo vệ bằng khóa khác.
  - Khóa dữ liệu (Data keys): Là các khóa được sử dụng để mã hóa dữ liệu cho người dùng. Trong thực tế, khóa dữ liệu thường là các khóa ngắn hạn.
- Phân loại các lớp khóa theo thời gian sử dụng gồm 2 loại khóa chính:
  - Khóa dài hạn (long-term keys): Là các khóa được sử dụng trong một khoảng thời gian dài. Một số loại khóa dài hạn thường gặp: khóa chủ, khóa dùng cho trao đổi khóa, hoặc khóa dùng cho thỏa thuận khóa.
  - Khóa ngắn hạn (short-term keys): Là các khóa được sử dụng trong một khoảng thời gian ngắn hoặc chỉ trong một phiên làm việc. Khóa ngắn hạn gồm các khóa được trao đổi trong quá trình trao đổi khóa, thỏa thuận khóa, dùng để mã hóa dữ liệu của người dùng.

Quản lý khóa đóng vai trò rất quan trọng trong việc đảm bảo các yêu cầu an toàn cần thiết của một hệ truyền tin. Cụ thể, quản lý khóa cung cấp các tính năng:

- Tính bí mật;
- Toàn vẹn;
- Xác thực;
- Không thể chối bỏ;
- Chữ ký số.

Như vậy, có thể thấy rằng: quản lý khóa phù hợp sẽ đảm bảo cho các thông tin khóa được an toàn, đặc biệt khi có nhiều thực thể tham gia truyền thông. Thông tin khóa được an toàn sẽ đảm bảo tính an toàn của hệ mã hóa.

Cũng giống như các kỹ thuật mã hóa, trong các phương pháp quản lý và phân phối khóa cũng có những mối đe dọa và các kỹ thuật tấn công nhằm tìm được khóa. Một số mối đe dọa cụ thể như sau:

- Các khóa bí mật bị lộ;
- Tính xác thực của các khóa bí mật và công khai bị thỏa hiệp (compromise): Tính xác thực bao gồm các hiểu biết và việc kiểm chứng thông tin nhận dạng của một bên mà khóa được chia sẻ;
- Sử dụng trái phép các khóa bí mật và công khai:
  - Sử dụng các khóa đã hết hiệu lực;
  - Sử dụng các khóa sai mục đích.

Nhằm đảm bảo an toàn và giữ bí mật cho khóa, trong quá trình phân phối và quản lý, các thực thể liên quan đến quá trình cần phải thực hiện nghiêm túc các quy định đã được đặt ra. Một trong những vấn đề rất cần quan tâm trong quy trình phân phối và quản lý khóa chính là chính sách an ninh. Quản lý khóa luôn được thực hiện trong khuôn khổ chính sách an ninh cụ thể, bao gồm:

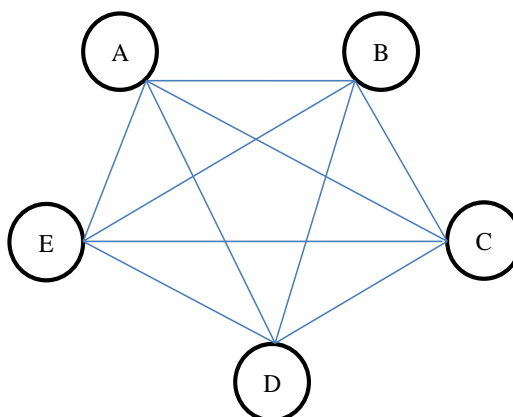
- Các thực thể và thủ tục cần thực hiện trong các khía cạnh kỹ thuật và quản trị khóa tự động hoặc bằng tay;
- Trách nhiệm của các bên có liên quan;
- Các bản ghi dữ liệu cần phải lưu để tạo các báo cáo về các vấn đề có liên quan đến an toàn khóa.

Trên đây, bài giảng đã trình bày một số khái niệm cơ bản cần lưu ý trong vấn đề quản lý và phân phối khóa. Một điểm cần quan tâm trong kỹ thuật quản lý và phân phối khóa chính là các mô hình thiết lập khóa (Key establishment). Một số mô hình có thể tìm hiểu là:

- Mô hình phân phối  $n^2$  khóa: Nếu một hệ thống có  $n$  người dùng tham gia truyền thông sử dụng kỹ thuật mã hóa khóa đối xứng và mỗi cặp người dùng cần trao đổi thông tin an toàn. Một số đặc điểm cần chú ý trong mô hình:
  - Mỗi cặp người dùng cần chia sẻ một khóa bí mật duy nhất;
  - Mỗi người dùng cần sở hữu  $(n-1)$  khóa bí mật;

- Tổng số khóa cần quản lý trong hệ thống là  $n(n-1)/2 \approx n^2$ ;
- Số khóa cần quản lý sẽ rất lớn nếu số người dùng lớn.

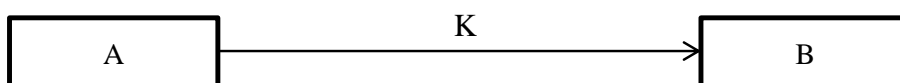
Ví dụ: một hệ thống có 5 người dùng (A, B, C, D, E) trao đổi dữ liệu thì số khóa cần quản lý sẽ là:  $5 \times 4 / 2 = 10$



**Hình 5. 1. Sơ đồ trao đổi khóa trong một hệ thống 5 người**

Từ quy trình thực hiện và tính toán trên có thể thấy rằng: nhược điểm lớn nhất của mô hình phân phối  $n^2$  khóa là người dùng phải nắm rõ tất cả các khóa của những người còn lại trong hệ thống. Nếu một hệ thống có quá nhiều người thì việc này rất phức tạp và khó khăn. Chính vì vậy, để khắc phục nhược điểm này, có thể sử dụng máy chủ trung tâm để quản lý và phân phối khóa. Máy chủ trung tâm để quản lý và phân phối khóa sẽ được giới thiệu vào mục 5.2 của bài giảng.

- Mô hình phân phối khóa điểm – điểm (Point-to-point key distribution): Việc phân phối khóa chỉ liên quan trực tiếp đến 2 thực thể tham gia truyền thông. Hình 5.2 mô tả quá trình trao đổi khóa giữa hai người dùng A và B.



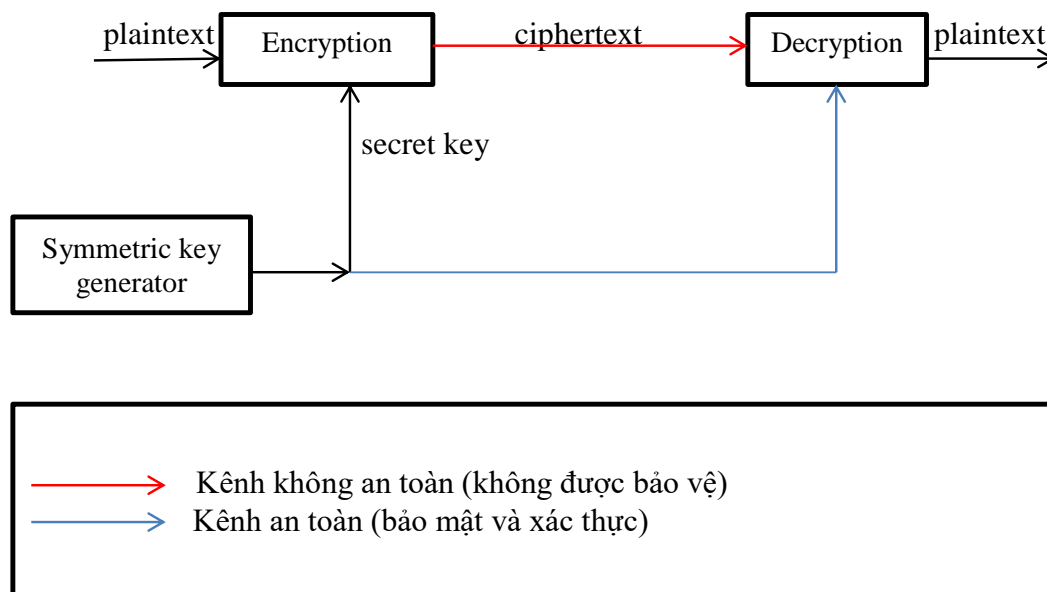
**Hình 5. 2. Phân phối khóa điểm – điểm**

## 5.2. PHÂN PHỐI VÀ THỎA THUẬN KHÓA BÍ MẬT

### 5.2.1. Các kỹ thuật phân phối và thỏa thuận khóa bí mật

### 5.2.1.1. Giới thiệu chung:

Đặc điểm chung của các kỹ thuật phân phối và thỏa thuận khóa bí mật là chỉ có một khóa duy nhất  $K_S$  (secret key) được sinh ra, được gửi đến cho cả 2 bên qua một kênh truyền an toàn. Hình 5.3 mô tả một mô hình tạo và sử dụng khóa trong hệ mã hóa khóa đối xứng:



**Hình 5. 3. Mô hình tạo và sử dụng khóa của hệ mã hóa khóa đối xứng**

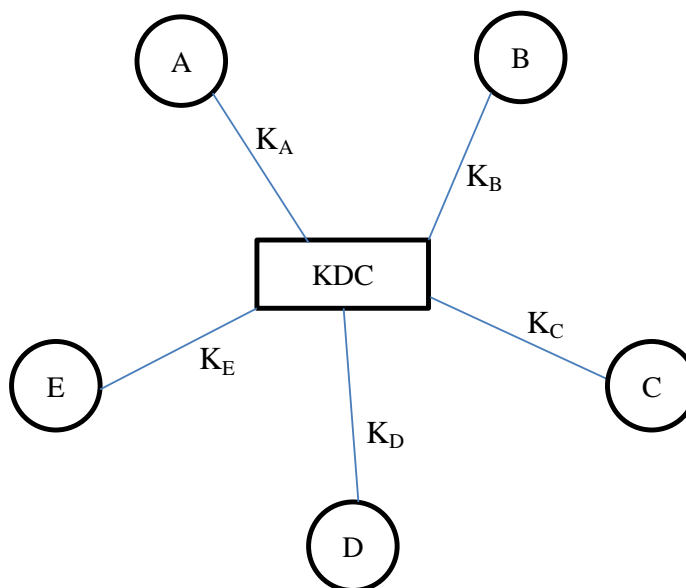
Từ hình 5.3 có thể thấy quá trình truyền tin giữa hai thực thể A và B diễn ra như sau:

- A muốn gửi tin cho B;
- A dùng  $K_S$  mã hóa bản rõ (plaintext), sau đó gửi bản mã (ciphertext) cho B qua một kênh truyền (không cần là kênh an toàn);
- B nhận được bản mã sẽ dùng khóa  $K_S$  để giải mã và thu được bản rõ.

### 5.2.1.2. Trung tâm phân phối khóa KDC (Key Distribution Center):

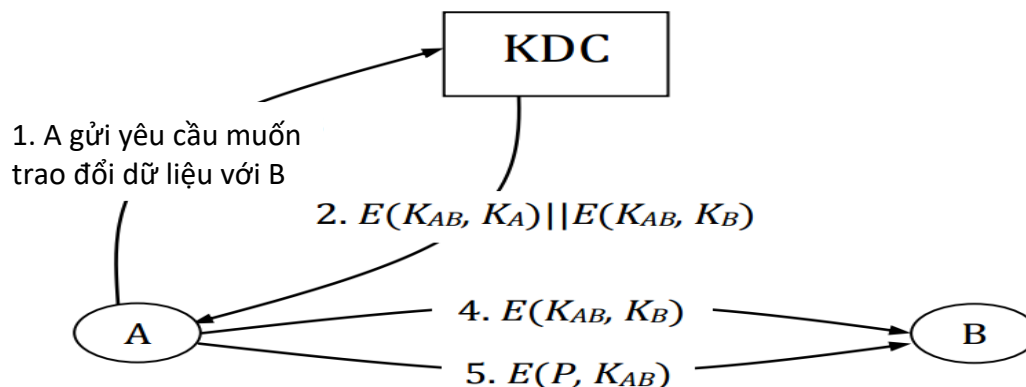
Như đã trình bày ở trên, để giải quyết nhược điểm trong vấn đề phân phối  $n^2$  khóa, các chuyên gia đề xuất phương pháp quản lý khóa thông qua trung tâm phân phối khóa KDC. Điểm cần chú ý trong trung tâm phân phối khóa KDC là: Mỗi người dùng chỉ cần có 1 khóa chia sẻ với KDC còn khóa dùng để chia sẻ dữ liệu giữa những người dùng sẽ do KDC cung cấp (xem ví dụ ở hình 5.4).

Hình 5.4 cho ví dụ về một mô hình trao đổi khóa tập trung cho 5 người (A, B, C, D, E) với  $K_A$ ,  $K_B$ ,  $K_C$ ,  $K_D$ ,  $K_E$  lần lượt là khóa của A, B, C, D, E dùng để chia sẻ với KDC.



**Hình 5. 4. Mô hình trao đổi khóa tập trung**

Một trong những vấn đề cần quan tâm đối với trung tâm phân phối khóa KDC chính là thủ tục trao đổi khóa sử dụng KDC. Hình 5.5 mô tả thủ tục trao đổi khóa đơn giản sử dụng KDC.



**Hình 5. 5. Mô hình trao đổi khóa đơn giản sử dụng KDC**

Trong đó:

- A có khóa bí mật  $K_A$  với KDC;
- B có khóa bí mật  $K_B$  với KDC;
- KDC sẽ cung cấp khóa dùng để trao đổi dữ liệu giữa A và B ( $K_{AB}$ ).

Quy trình trao đổi khóa diễn ra như sau:

- A gửi yêu cầu muốn trao đổi dữ liệu với B cho KDC nhằm mục đích lấy được khóa chung  $K_{AB}$ . Yêu cầu này không cần thiết phải mã hóa, nó chứa định danh của A và định danh của B;
- KDC tạo ra một khóa bí mật  $K_{AB}$  và mã hóa thành 2 bản: Một bản được mã hóa bằng khóa bí mật của A –  $E(K_{AB}, K_A)$ ; một bản được mã hóa bằng khóa bí mật của B –  $E(K_{AB}, K_B)$ ;
- KDC gửi trả về cho A 2 bản mã này. A giải mã 1 bản bằng khóa bí mật  $K_A$  của mình để lấy ra  $K_{AB}$  và gửi bản còn lại cho B. B giải mã bằng khóa bí mật  $K_B$  của B để lấy được  $K_{AB}$ ;
- A và B trao đổi dữ liệu với nhau qua bằng khóa  $K_{AB}$ . Kết thúc quá trình trao đổi dữ liệu,  $K_{AB}$  sẽ được hủy bỏ. Nếu lần sau A muốn trao đổi dữ liệu với B thì KDC sẽ tạo ra một khóa  $K_{AB}$  khác.

### 5.2.1.3. Trung tâm dịch chuyển khóa KTC (Key Translation Center)

Trong thực tế, để giải quyết khó khăn trong việc phân phối  $n^2$  khóa, có thể sử dụng trung tâm dịch chuyển khóa KTC (Key Translation Center). Tiếp theo, bài giảng sẽ trình bày về trung tâm dịch chuyển khóa KTC để thấy được sự giống và khác nhau giữa KDC và KTC.

Về cơ bản, KTC cũng được sử dụng để phân phối khóa như KDC. Tuy nhiên, một bên tham gia truyền thông sẽ cung cấp khóa phiên (session key). Một số điểm cần chú ý trong KTC như sau:

- A sở hữu khóa dài hạn  $K_{AT}$  – chia sẻ với KTC;
- B sở hữu khóa dài hạn  $K_{BT}$  – chia sẻ với KTC;
- Trung tâm dịch khóa T là một máy chủ tin cậy, cho phép hai bên A và B không trực tiếp chia sẻ thông tin khóa thiết lập kênh truyền thông an toàn sử dụng hai khóa dài hạn  $K_{AT}$  và  $K_{BT}$ .

Thuật toán phân phối khóa sử dụng KTC được mô tả như sau:

- Các thông điệp trao đổi trong quá trình hoạt động:

$$(1) \quad A \rightarrow T: A, E((B \parallel M), K_{AT})$$

$$(2) \quad T \rightarrow A: E((M \parallel A), K_{BT})$$

$$(3) \quad A \rightarrow B: E((M \parallel A), K_{BT})$$

- Các bước thực hiện cụ thể:

- Đầu tiên, A mã hóa thông điệp bí mật M (M là 1 khóa phiên) và số định danh của B (người nhận) sử dụng khóa  $K_{AT}$  và gửi thông điệp kèm theo số định danh của A cho T;
- T nhận được thông điệp từ A. T giải mã thông điệp, xác định được người nhận là B. T mã hóa M sử dụng khóa  $K_{BT}$ . Tiếp theo, T gửi lại thông điệp đã mã hóa cho A để A để chuyển cho B;
- Sau khi B nhận được thông điệp từ A, B giải mã thông điệp sử dụng khóa  $K_{BT}$  để có được M;

- Kết quả: A và B sử dụng khóa phiên M để trao đổi dữ liệu với nhau.

Để không phải duy trì một cơ sở dữ liệu an toàn lưu các khóa bí mật của người dùng (hoặc phải sao chép cơ sở dữ liệu này đến nhiều máy chủ), các chuyên gia đề xuất chứng chỉ khóa đối xứng (Symmetric-key certificates) trong KTC. Đặc điểm của chứng chỉ khóa đối xứng như sau:

- Không phải yêu cầu các khóa bí mật của người dùng từ một cơ sở dữ liệu an toàn.
- Mỗi chứng chỉ khóa đối xứng có một thời hạn sử dụng xác định.

Cơ chế sử dụng chứng chỉ khóa đối xứng trong KTC như sau:

- Thực thể B sở hữu khóa dài hạn  $K_{BT}$  – chia sẻ với KTC T;
- Khóa dài hạn  $K_{BT}$  được lưu trên T nhúng trong chứng chỉ khóa đối xứng  $E((K_{BT}, B), K_T)$ ;
- $K_T$  là khóa chủ của T và chỉ T được biết;
- Chứng chỉ khóa đối xứng được sử dụng như là bản ghi nhớ cho chính T (chỉ T mới có thể mở chứng chỉ này);



- Chứng chỉ khóa đối xứng cũng có thể được chuyển cho B khi có yêu cầu truy nhập khóa  $K_{BT}$  để dịch thông điệp;

Từ quy trình trên có thể thấy rằng, thay vì phải lưu trữ toàn bộ khóa dài hạn của người dùng, với chứng chỉ khóa đối xứng, T chỉ cần lưu an toàn khóa chủ  $K_T$  của mình.

Thuật toán phân phối khóa sử dụng KTC với chứng chỉ khóa đối xứng:

- Các thông điệp trao đổi:

$$(1) A \rightarrow T: \text{Scert}_A \parallel E((B \parallel M), K_{AT}) \parallel \text{SCert}_B$$

$$(2) T \rightarrow A: E((A \parallel M), K_{BT})$$

$$(3) A \rightarrow B: E((A \parallel M), K_{BT})$$

Trong đó:  $\text{Scert}_A = E((K_{AT} \parallel A), K_T)$ ;  $\text{Scert}_B = E((K_{BT} \parallel A), K_T)$ .

- Các bước thực hiện cụ thể như sau:

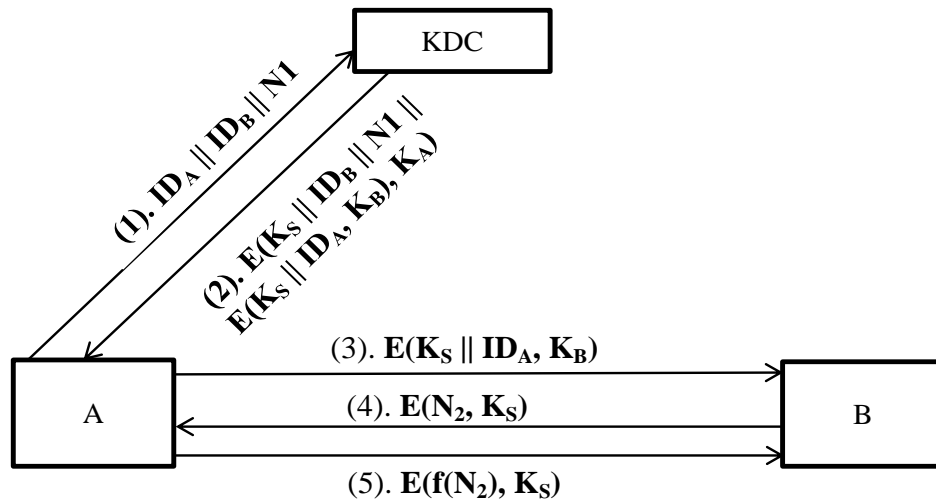
- Đầu tiên, A mã hóa M và số định danh của B (người nhận) sử dụng khóa  $K_{AT}$  và gửi thông điệp kèm chứng chỉ khóa đối xứng của A ( $\text{SCert}_A$ ) và B ( $\text{SCert}_B$ ) lấy từ cơ sở dữ liệu công cộng cho T. Cơ sở dữ liệu công cộng được sử dụng để lưu định danh người dùng và chứng chỉ khóa đối xứng của họ.
- T nhận được yêu cầu từ A, T sử dụng  $K_T$  để giải mã chứng chỉ của A và B, lấy được  $K_{AT}$  và  $K_{BT}$ . T sử dụng  $K_{AT}$  giải mã thông điệp để có B và M. Đồng thời T kiểm tra định danh của B có trùng với định danh lưu trong chứng chỉ khóa đối xứng của B. T mã hóa M sử dụng khóa  $K_{BT}$ . T gửi lại thông điệp đã mã hóa cho A để A chuyển cho B, hoặc T có thể gửi thẳng cho B.
- B sau khi nhận được thông điệp từ A hoặc T, B giải mã thông điệp sử dụng khóa  $K_{BT}$  để có M.

## 5.2.2. Một số giao thức phân phối và thỏa thuận khóa bí mật

### 5.2.2.1. Giao thức Needham-Schroeder:

Giao thức Needham-Schroeder do Roger Needham và Michael Schroeder phát minh vào năm 1978, cho phép các thực thể trao đổi thông tin có thể chứng minh nhận dạng của mình đồng thời chống lại việc nghe lén và phát hiện, ngăn chặn việc truy cập, thay đổi thông tin trên đường truyền. Needham-Schroeder sử dụng “nonces” để chống lại

kỹ thuật replay attack. Một nonce là một con số mà chỉ được sử dụng một lần. Nó có thể là một số ngẫu nhiên, một biến trạng thái, hoặc một nhãn thời gian. Hình 5.6 mô tả quá trình trao đổi khóa của giao thức Needham-Schroeder.



**Hình 5. 6.** Sơ đồ mô tả quá trình trao đổi khóa của giao thức Needham-Schroeder

Các bước thực hiện:

- **A → KDC:  $ID_A || ID_B || N1$ :** A gửi cho KDC định danh của A –  $ID_A$ , định danh của B –  $ID_B$  và nonce của A –  $N1$  ;
- **KDC → A:  $E(K_S || ID_B || N1 || E(K_S || ID_A, K_B), K_A)$ :** KDC sẽ gửi cho A một bản tin được mã hóa bằng khóa bí mật của A với KDC ( $K_A$ ). Bản tin này chứa khóa phiên  $K_S$ ,  $ID_B$ ,  $N1$  và 1 “vé” được mã hóa riêng dành cho B. Vé này chứa  $K_S$  và  $ID_A$ , được mã hóa bằng khóa bí mật của B với KDC ( $K_B$ );
- A giải mã sẽ lấy được  $K_S$  và vé của B -  $E(K_S || ID_A, K_B)$ ;
- **A → B:  $E(K_S || ID_A, K_B)$ :** A gửi vé của B đến cho B;
- **B → A:  $E(N2, K_S)$ :** B gửi đến cho A nonce của B được mã hóa bằng khóa phiên  $K_S$ ;
- **A → B:  $E(f(N2), K_S)$ :** A gửi đến cho B giá trị  $f(N2)$  được mã hóa bằng  $K_S$ . Hàm  $f$  là một hàm bất kỳ được chọn sẵn từ trước. Mục đích của bước này là để B xác nhận xem có đúng là A đang trao đổi với mình hay không;

- Sau khi xác thực xong, A và B tiến hành trao đổi dữ liệu bằng  $K_s$ .

#### 5.2.2.2. Giao thức Otway-Rees:

Giao thức Otway-Rees được dùng trong các mạng máy tính không an toàn (chẳng hạn như Internet), cho phép các cá nhân có thể trao đổi thông tin để xác thực, đồng thời chống lại việc nghe trộm cũng như gửi lại các gói tin cũ hay sửa đổi các gói tin trên đường truyền.

Các bước thực hiện:

- **A  $\rightarrow$  B:  $M \parallel ID_A \parallel ID_B \parallel E(M \parallel ID_A \parallel ID_B \parallel N_A, K_A)$ :** A gửi cho B một thông điệp bao gồm: nonce chung (M), định danh của A ( $ID_A$ ), định danh của B ( $ID_B$ ) và một “vé” được mã hóa bằng khóa bí mật của A với KDC ( $K_A$ ). Vé này chứa M,  $ID_A$ ,  $ID_B$ , và nonce của A ( $N_A$ );
- **B  $\rightarrow$  KDC:  $M \parallel ID_A \parallel ID_B \parallel E(M \parallel ID_A \parallel ID_B \parallel N_A, K_A) \parallel E(M \parallel ID_A \parallel ID_B \parallel N_B, K_B)$ :** B tạo ra một “vé” tương tự bao gồm M,  $ID_A$ ,  $ID_B$ , định danh của B ( $N_A$ ). Vé này được mã hóa bằng khóa bí mật của B với KDC ( $K_B$ ). B sẽ gửi đến KDC toàn bộ thông điệp nhận được từ A kèm theo vé vừa tạo;
- **KDC  $\rightarrow$  B:  $M \parallel E(N_A \parallel K_{AB}, K_A) \parallel E(N_B \parallel K_{AB}, K_B)$ :** KDC tạo ra khóa phiên  $K_{AB}$ , sau đó gửi cho B một thông điệp bao gồm:
  - Nonce chung M
  - Vé của A, được mã hóa bằng  $K_A$ , chứa  $N_A$  và  $K_{AB}$
  - Vé của B, được mã hóa bằng  $K_B$ , chứa  $N_B$  và  $K_{AB}$
- **B  $\rightarrow$  A:  $M \parallel E(N_A \parallel K_{AB}, K_A)$ :** B giữ lại vé của mình và gửi cho A thông điệp chứa M và vé của A; B giải mã vé của mình bằng  $K_B$  để lấy được  $K_{AB}$ ; A giải mã vé của mình bằng  $K_A$  để lấy được  $K_{AB}$ ;
- A và B sử dụng  $K_{AB}$  để trao đổi dữ liệu với nhau.

#### 5.2.2.3. Giao thức không khóa Shamir

Giao thức không khóa Shamir là giao thức cho phép 2 thực thể trao đổi thông tin mà không cần trao đổi trước khóa.

Khởi tạo các tham số cho quá trình trao đổi:

- Lựa chọn và thống nhất dùng chung một số nguyên tố  $p$  sao cho việc tính toán logarit rời rạc modulo của  $p$  là không khả thi về mặt tính toán và phân tích;

- A chọn cho mình số nguyên bí mật  $a$

B chọn cho mình số nguyên bí mật  $b$

Chú ý:  $a$  và  $b$  đều là nguyên tố cùng nhau với  $(p-1)$ ;  $0 \leq a, b \leq (p-2)$

- A tính:  $a^{(1/a) \bmod (p-1)}$

B tính:  $b^{(1/b) \bmod (p-1)}$

Từ quá trình khởi tạo thấy rằng, trong giao thức Shamir, hai bên không cần thỏa thuận trước khóa dùng chung mà chỉ cần tự lựa chọn cho mình tham số bí mật, tính toán và gửi cho bên liên quan giá trị tính toán cuối cùng. Độ an toàn của giao thức dựa trên bài toán logarit rời rạc với modulo của  $p$ .

Các thông điệp trao đổi như sau:

(1)  $A \rightarrow B: K^a \bmod p$

(2)  $B \rightarrow A: (K^a \bmod p)^b \bmod p = (K^a)^b \bmod p$

(3)  $A \rightarrow B: ((K^a \bmod p)^b \bmod p)^{(1/a) \bmod (p-1)} = (K^{ab})^{(1/a)} \bmod p = K^b \bmod p$

Các bước tính toán trong quá trình trao đổi khóa của giao thức Shamir được thể hiện trong bảng 5.2:

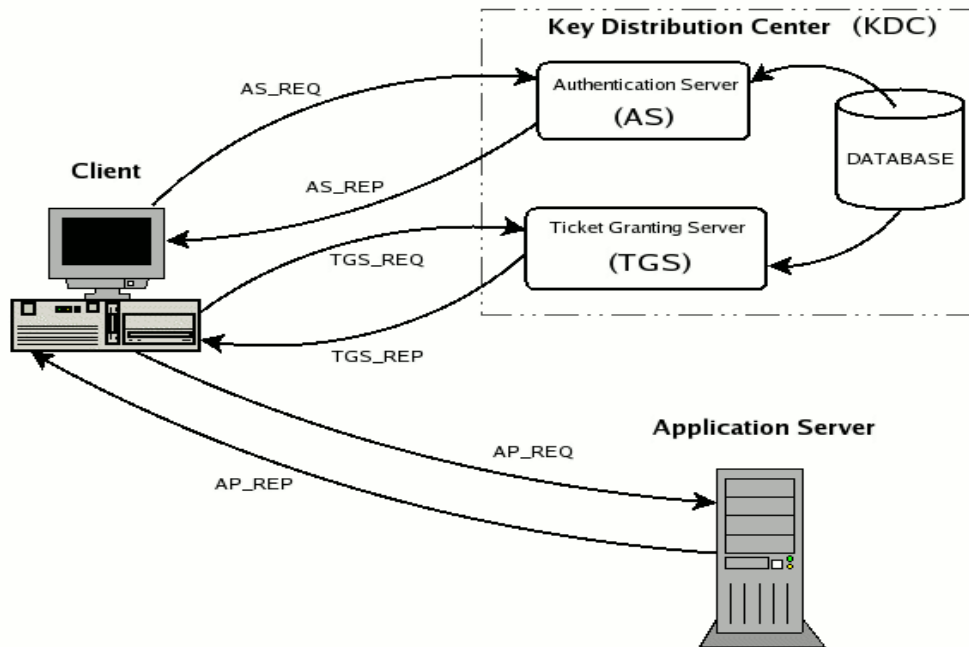
**Bảng 5. 1. Quá trình trao đổi khóa giữa hai thực thể trong giao thức Shamir**

A	B
Chọn khóa ngẫu nhiên $K$ thỏa mãn điều kiện: $1 \leq K \leq p - 1$	
Tính toán và gửi cho B thông điệp (1)	B nhận thông điệp (1), tính toán và gửi cho A thông điệp (2)
A nhận thông điệp (2), tính toán và gửi cho B thông điệp (3)	B nhận thông điệp (3), tính toán ra khóa chung bí mật là $K \bmod p$
A sử dụng khóa chung bí mật là $K \bmod p$	B sử dụng khóa chung bí mật là $K \bmod p$

#### 5.2.2.4. Giao thức Kerberos

Trong chương 1 của bài giảng đã trình bày chi tiết về khái niệm, chức năng và vai trò của Kerberos. Tiếp theo, bài giảng sẽ phân tích phương pháp, kỹ thuật phân phối và quản

lý khóa của giao thức Kerberos. Hình 5.7 Mô tả quy trình hoạt động của giao thức Kerberos.



**Hình 5. 7. Hoạt động của Kerberos**

Phân tích hình 5.7 thấy được quy trình hoạt động của giao thức Kerberos như sau:

**Chú ý:** Trong các mô tả dưới đây, dữ liệu không được mã hóa nằm trong ngoặc tròn () và dữ liệu đã được mã hóa nằm trong ngoặc nhọn {}:  $(x, y, z)$  nghĩa là  $x, y, z$  chưa được mã hóa;  $\{x, y, z\}_K$  nghĩa là cả  $x, y, z$  đã được mã hóa bằng khóa đối xứng  $K$ . Thứ tự các thành phần được liệt kê trong gói tin không liên quan tới thứ tự trong thông điệp thực tế.

**1) Authentication Server Request (AS\_REQ):** Trong pha này, máy khách yêu cầu máy chủ cấp phát vé (cụ thể là máy chủ xác thực) cấp cho một *Ticket Granting Ticket* (TGT). Yêu cầu này không được mã hóa và có dạng:

$$AS\_REQ = (Principal_{Client}, Principal_{Service}, IP\_list, Lifetime)$$

Trong đó:

- $Principal_{Client}$  là *principal* liên kết tới người dùng đang xin được xác thực (ví dụ: *nam@EXAMPLE.COM*);

- $Principal_{Service}$  là *principal* liên kết tới dịch vụ. Do vé đang được yêu cầu nên *principal* này là “ $krbtgt/REALM@REALM$ ”;
- $IP\_list$  là danh sách địa chỉ *IP* của các *host* có thể sử dụng vé;
- $Lifetime$  là thời hạn tối đa để phát vé (thời gian sống).

**2) Authentication Server Reply (AS\_REP):** Khi nhận được yêu cầu ở trên, máy chủ xác thực sẽ kiểm tra xem  $Principal_{Client}$  và  $Principal_{Service}$  có trong cơ sở dữ liệu của máy chủ phân phối khóa hay không. Nếu không tồn tại ít nhất một trong hai trường thì một thông báo lỗi sẽ được gửi tới máy khách, ngược lại, máy chủ xác thực sẽ trả lời như sau:

- Tạo một khóa phiên ngẫu nhiên dùng làm bí mật được trao đổi giữa máy khách và máy chủ cấp phát vé. Gọi khóa ngẫu nhiên này là  $SK_{TGS}$ ;
- Tạo *Ticket Granting Ticket*, chèn vào trong đó *principal* của người gửi yêu cầu, *principal* dịch vụ (thường là  $krbtgt/REALM@REALM$ ), danh sách địa chỉ *IP* (ba thành phần này được sao chép từ gói tin  $AS\_REQ$ ), ngày giờ của *KDC* ở dạng nhãn thời gian, thời gian sống và cuối cùng là khóa phiên  $SK_{TGS}$
- Tạo và gửi gói tin trả lời, bao gồm vé sinh ra ở bước trước (mã hóa bằng khóa bí mật của dịch vụ  $K_{TGS}$ ); *principal* dịch vụ, nhãn thời gian, thời gian sống và khóa phiên, tất cả được mã hóa bằng khóa bí mật của người gửi yêu cầu ( $K_{User}$ )

$$AS\_REP = \{ Principal_{Service}, Timestamp, Lifetime, SK_{TGS} \}_{K_{User}} \{ TGT \}_{K_{TGS}}$$

**Chú ý:** Trong triển khai thực tế, có thể giới hạn thời gian sống trong cài đặt của *KDC* và áp dụng cho toàn bộ vé.

**3) Ticket Granting Server Request (TGS\_REQ):** Khi đã chứng minh được danh tính, người dùng muốn truy cập dịch vụ nhưng vẫn chưa có vé phù hợp, họ gửi yêu cầu  $TGS\_REQ$  đến máy chủ cấp phát vé. Quá trình tạo gói tin  $TGS\_REQ$  như sau:

- Tạo một authenticator với *principal* người dùng, nhãn thời gian của máy khách và mã hóa bằng khóa phiên  $TGS$ :

$$Authenticator = \{ Principal_{Client}, Timestamp \}_{SK_{TGS}}$$

- Tạo gói tin yêu cầu gồm principal của dịch vụ mà khách hàng yêu cầu cùng nhãn thời gian ở dạng không mã hóa; Ticket Granting Ticket mã hóa bằng khóa bí mật của TGS và authenticator vừa tạo:

$$TGS\_REQ = ( Principal_{Service} , Lifetime , Authenticator ) \{ TGT \} K_{TGS}$$

**4) Ticket Granting Server Reply (TGS\_REP):** Khi nhận được gói tin  $TGS\_REQ$ , bước đầu tiên, máy chủ cấp phát vé sẽ kiểm tra xem principal của dịch vụ mà người dùng yêu cầu ( $Principal_{Service}$ ) có trong cơ sở dữ liệu của trung tâm phân phối khóa hay không. Nếu có, nó dùng khóa của  $krbtgt/REALM@REALM$  để mở  $TGT$  và trích xuất ra khóa phiên ( $SK_{TGS}$ ). Để có thể cấp phát vé dịch vụ, máy chủ  $TGS$  sẽ kiểm tra các điều kiện:

- $TGT$  chưa hết hạn;
- $Principal_{Client}$  trong authenticator trùng với trong  $Principal_{Client}$   $TGT$ ;
- $Authenticator$  chưa hết hạn và không có trong bộ đệm phát lại;
- Nếu  $IP\_list$  không rỗng, địa chỉ  $IP$  nguồn của gói tin yêu cầu ( $TGS\_REQ$ ) phải trùng với một trong các địa chỉ  $IP$  có trong danh sách  $IP$ .

Nếu thỏa mãn các điều kiện trên thì  $TGT$  thực sự được gửi bởi người yêu cầu dịch vụ và do đó, máy chủ  $TGS$  sẽ tạo gói tin trả lời như sau:

- Tạo một khóa phiên ngẫu nhiên dùng làm khóa bí mật chia sẻ giữa máy khách và dịch vụ. Gọi khóa ngẫu nhiên này là  $SK_{Service}$ ;
- Tạo vé dịch vụ, trong vé gồm có principal của người dùng, danh sách địa chỉ  $IP$ , ngày và giờ (của máy chủ  $KDC$ ) ở dạng nhãn thời gian, thời gian sống (là giá trị nhỏ nhất giữa thời gian sống của  $TGT$  và thời gian sống của principal dịch vụ) và cuối cùng là khóa phiên  $SK_{Service}$ . Gọi vé này là  $T_{Service}$ :

$$T_{Service} = ( Principal_{Client} , Principal_{Service} , IP\_list , Timestamp , Lifetime , SK_{Service} )$$

- Gửi thông điệp trả lời, bao gồm vé vừa tạo, mã hóa bằng khóa bí mật của dịch vụ gọi là  $K_{Service}$ ; principal dịch vụ, nhãn thời gian, thời gian sống và khóa phiên mới, tất cả được mã hóa bằng khóa được trích xuất từ  $TGT$ :

$$TGS\_REP = \{ Principal_{Service} , Timestamp , Lifetime , SK_{Service} \} SK_{TGS} \{ T_{Service} \} K_{Service}$$

Khi nhận được trả lời, máy khách sẽ dùng khóa phiên  $SK_{TGS}$  trong bộ đệm ủy nhiệm để giải mã phần thông điệp có chứa khóa phiên mới để lấy được  $SK_{Service}$  và lưu nó lại cùng với vé dịch vụ  $T_{Service}$ , tuy nhiên vẫn ở dạng mã hóa.

**5) Application Request ( $AP\_REQ$ ):** Khi có đủ giấy ủy nhiệm (vé và khóa phiên có liên quan) để truy cập dịch vụ, máy khách yêu cầu máy chủ ứng dụng cấp quyền truy cập tài nguyên bằng cách gửi gói tin  $AP\_REQ$ . Khác với những gói tin trao đổi với máy chủ  $KDC$ ,  $AP\_REQ$  không theo một chuẩn nào mà tùy thuộc vào từng ứng dụng. Có thể xem xét kịch bản sau:

- Máy khách tạo một authenticator gồm principal người dùng và nhãn thời gian, tất cả được mã hóa bằng khóa phiên  $SK_{Service}$ , là khóa được chia sẻ giữa máy khách và máy chủ ứng dụng:

$$Authenticator = \{ Principal_{Client}, Timestamp \} SK_{Service}$$

- Tạo một gói tin yêu cầu gồm authenticator vừa tạo và vé dịch vụ  $T_{Service}$ , mã hóa bằng khóa  $K_{Service}$ :

$$AP\_REQ = Authenticator \{ T_{Service} \} K_{Service}$$

- Khi nhận được  $AP\_REQ$ , máy chủ ứng dụng sẽ giải mã gói tin bằng khóa bí mật  $K_{Service}$  (khóa bí mật của dịch vụ) để lấy được vé  $T_{Service}$  và trích xuất ra khóa phiên  $SK_{Service}$ . Khóa phiên  $SK_{Service}$  là khóa được sử dụng để mã hóa authenticator. Để xác thực và cấp quyền truy cập dịch vụ cho người dùng, máy chủ sẽ kiểm tra các điều kiện:

- Vé chưa hết hạn;
- *Principal* người dùng trong *authenticator* trùng với *principal* người dùng trong vé;
- *Authenticator* không có trong bộ đệm phát lại và chưa hết hạn;
- Nếu *IP\_list* (được trích xuất từ vé) không rỗng, địa chỉ *IP* nguồn của gói tin yêu cầu ( $TGS\_REQ$ ) phải trùng với một trong các địa chỉ *IP* có trong danh sách *IP*.

**6) Application Reply ( $AP\_REP$ ):** Đây là bước tùy chọn, được sử dụng khi có nhu cầu xác thực lẫn nhau: không chỉ máy khách phải chứng minh danh tính với dịch vụ mà ngược lại, dịch vụ cũng cần chứng minh tính xác thực của mình. Nếu người dùng yêu cầu cấp vé dịch vụ một cách trực tiếp, bước  $TGS\_REQ$  và  $TGS\_REP$  sẽ bị bỏ qua.



**Chú ý:** Trong nguyên tắc hoạt động của giao thức Kerberos cần lưu ý đến các loại vé được trao đổi giữa các bên liên quan. Tùy thuộc vào các thuộc tính (hoặc các cờ) của vé mà nó được xử lý theo nhiều cách khác nhau. Dưới đây là một số loại vé quan trọng.

- **Vé khởi tạo:** Vé khởi tạo là vé được nhận trực tiếp từ máy chủ xác thực AS, tức là khi người dùng phải xác thực bằng cách nhập mật khẩu. Do đó, có thể coi TGT là một vé khởi tạo. Một số ví dụ về vé khởi tạo đã được trình bày chi tiết trong tài liệu tham khảo.

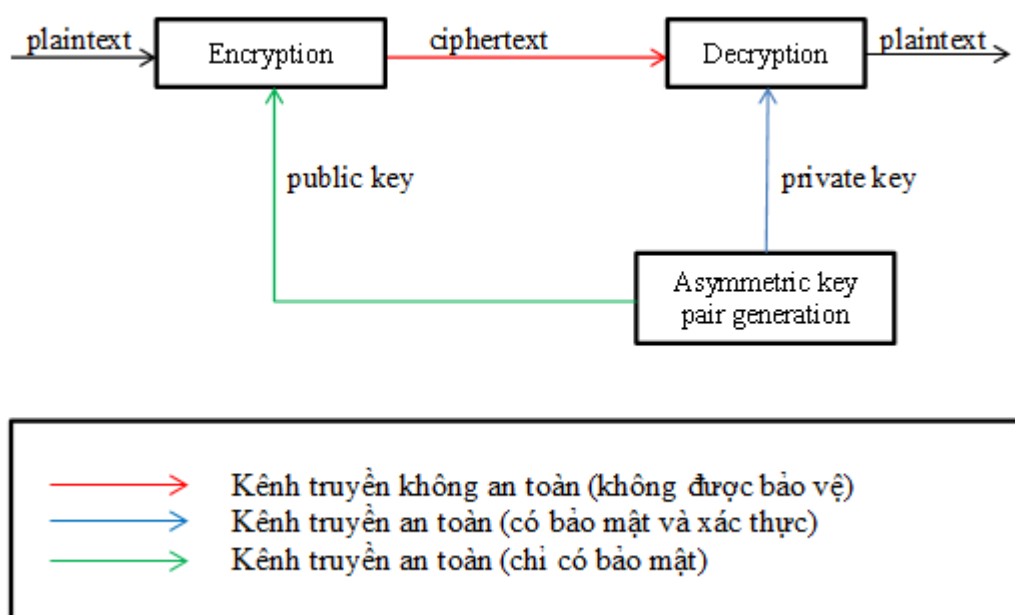
- **Vé có khả năng gia hạn:** Vé có khả năng gia hạn có thể được gửi lại đến máy chủ KDC để làm mới. Máy chủ KDC chỉ cho phép gia hạn khi vé chưa hết hạn và chưa vượt quá thời gian gia hạn tối đa (cài đặt trong cơ sở dữ liệu của trung tâm phân phối khóa).

- **Xác thực chéo:** Người dùng thuộc *realm* (một miền quản trị xác thực) có thể được xác thực và truy cập dịch vụ của máy chủ thuộc *realm* khác. Đây là tính năng xác thực chéo, dựa trên mối quan hệ tin tưởng lẫn nhau giữa các *realm*. Xác thực chéo có thể là đơn hướng (người dùng thuộc A có thể truy cập dịch vụ thuộc *realm* B nhưng chiều ngược lại thì không thể) hoặc song hướng. Trong xác thực chéo, có nhiều kiểu quan hệ như: quan hệ tin tưởng trực tiếp, quan hệ tin tưởng bắc cầu, quan hệ tin tưởng phân cấp. Trong một số tài liệu tham khảo của bài giảng đã trình bày chi tiết về quy trình hoạt động của những kiểu quan hệ này.

### 5.3. PHÂN PHỐI VÀ THỎA THUẬN KHÓA CÔNG KHAI

#### 5.3.1. Các kỹ thuật phân phối và thỏa thuận khóa công khai

Các kỹ thuật phân phối khóa công khai thường giả thiết các bên tham gia truyền thông sở hữu khóa công khai có tính xác thực (authentic public keys), tức là các khóa công khai được tạo ra và sử dụng hợp pháp. Việc phân phối khóa công khai cần đảm bảo tính xác thực của chủ thể khóa công khai. Hình 5.8 trình bày mô hình tổng quát về quy trình tạo và sử dụng khóa trong hệ mã hóa khóa bất đối xứng.



**Hình 5. 8. Mô hình tạo và sử dụng khóa của hệ mã hóa khóa bất đối xứng**

Quá trình sinh khóa và truyền tin trong hình 5.8 diễn ra như sau:

- B sinh ra một cặp khóa: khóa công khai và khóa bí mật. Sau đó, B gửi cho A khóa công khai của mình qua một kênh truyền an toàn (chỉ dùng để xác thực);
- A sau khi nhận được khóa công khai của B. A dùng khóa công khai của B để mã hóa bản rõ, sau đó gửi bản mã cho B qua một kênh truyền (không cần là kênh an toàn);
- B nhận được bản mã sẽ dùng khóa bí mật của mình để giải mã và thu được bản rõ.

Cũng giống như kỹ thuật phân phối khóa bí mật, trong kỹ thuật phân phối khóa công khai cũng có nhiều phương pháp phân phối khóa. Tiếp theo, bài giảng sẽ trình bày một số phương pháp phân phối khóa phổ biến.

**Trao đổi kiểu điểm-điểm thông qua kênh tin cậy:**

Đặc điểm của phương pháp phân phối khóa công khai theo kiểu điểm-điểm thông qua kênh tin cậy như sau: Các bên trực tiếp trao đổi khóa công khai với nhau thông qua các kênh tin cậy như thư bảo đảm hoặc các phương tiện chuyển giao đảm bảo khác, thích hợp với các hệ thống đóng kín hoặc cỡ nhỏ và có thể sử dụng với các trao đổi không

thường xuyên. Tuy nhiên, nhược điểm lớn nhất của phương pháp này là: độ trễ lớn và sử dụng kênh tin cậy riêng đắt tiền.

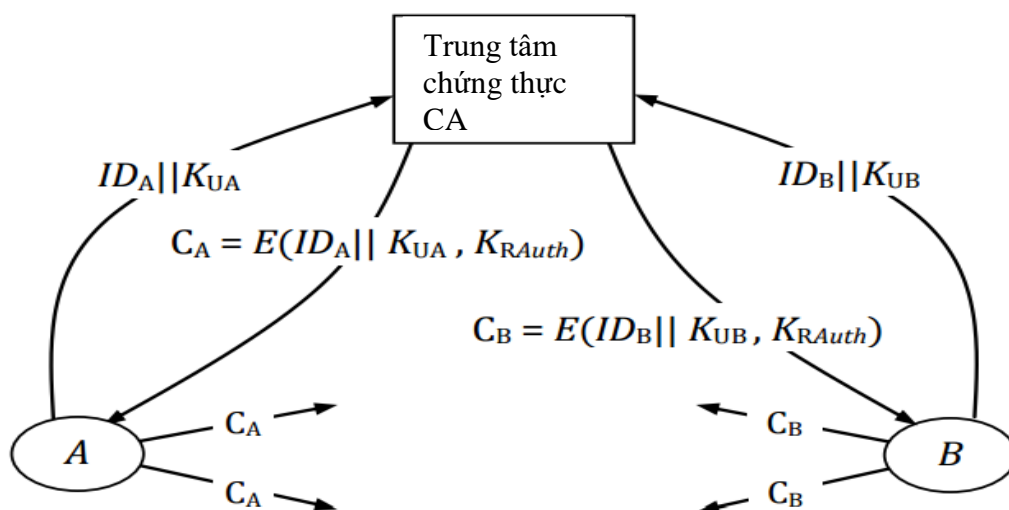
**Truy nhập trực tiếp vào danh mục công cộng (public-key registry):**

Đặc điểm của phương pháp phân phối khóa công khai theo kiểu truy nhập trực tiếp vào danh mục công cộng như sau: Một cơ sở dữ liệu công cộng tin cậy được thiết lập, bao gồm tên người dùng và khóa công khai tương ứng. Cơ sở dữ liệu công cộng này có thể được vận hành bởi 1 bên tin cậy. Người dùng có thể truy nhập khóa công khai từ cơ sở dữ liệu này.

**Sử dụng một máy chủ trực tuyến tin cậy:**

Đặc điểm của phương pháp phân phối khóa công khai theo kiểu sử dụng một máy chủ trực tuyến tin cậy như sau: Máy chủ trực tuyến tin cậy cung cấp các khóa công khai và khóa công khai được ký và gửi cho bên yêu cầu. Ưu điểm của phương pháp này là: Kênh truyền không đòi hỏi phải bí mật và bên yêu cầu sử dụng khóa công khai của máy chủ để xác thực chữ ký của máy chủ và qua đó kiểm tra tính xác thực, toàn vẹn của khóa. Tuy nhiên, phương pháp này cũng có nhược điểm là: Máy chủ phải luôn trực tuyến và máy chủ có thể trở thành điểm nút cổ chai.

**Sử dụng một máy chủ không trực tuyến và chứng chỉ:**



**Hình 5. 9. Mô hình trao đổi khóa công khai thông qua sử dụng một máy chủ không trực tuyến và chứng chỉ**

Hình 5.9 mô tả một mô hình trao đổi khóa công khai thông qua sử dụng một máy chủ không trực tuyến và chứng chỉ. Các bước thực hiện cấp chứng chỉ:

- A gửi định danh  $ID_A$  và khóa công khai  $K_{UA}$  đến trung tâm chứng thực CA. Trung tâm chứng nhận kiểm tra tính hợp lệ của A. A phải có bằng chứng chứng minh mình thực sự là A với CA. Sau khi xác nhận A, trung tâm chứng thực sẽ cấp một chứng chỉ  $C_A$  để xác nhận rằng khóa công khai  $K_{UA}$  đó là tương ứng với  $ID_A$ . Chứng chỉ được ký chứng thực bằng khóa riêng của trung tâm để đảm bảo rằng nội dung của chứng chỉ là do trung tâm ban hành;

$$C_A = E(ID_A || K_{UA}, K_{RAuth})$$

- A công khai chứng chỉ  $C_A$ ;
- B muốn trao đổi thông tin với A thì sẽ giải mã  $C_A$  bằng khóa công khai của trung tâm chứng thực để có được khóa công khai  $K_{UA}$  của A. Nếu B tin tưởng trung tâm chứng thực thì B sẽ tin tưởng rằng  $K_{UA}$  là khóa công khai của A.

Như vậy, thay vì phải xác nhận và tin tưởng khóa công khai của rất nhiều người khi muốn trao đổi dữ liệu, thì người gửi chỉ cần tin tưởng trung tâm chứng thực và chứng chỉ khóa công khai cũng có thể được đưa vào danh mục công cộng và người dùng có thể truy nhập.

### Sử dụng các hệ thống đảm bảo tính xác thực với các tham số công cộng:

Đặc điểm của phương pháp phân phối khóa công khai theo kiểu sử dụng các hệ thống đảm bảo tính xác thực với các tham số công cộng như sau: Các hệ thống dựa trên định danh (Identity-based systems) và sử dụng các khóa được chứng thực mặc nhiên (implicitly certified keys). Ưu điểm của phương pháp này là hệ thống có khả năng phát hiện các sửa đổi với các tham số công cộng.

#### **5.3.2. Chứng chỉ xác thực khóa công khai X.509**

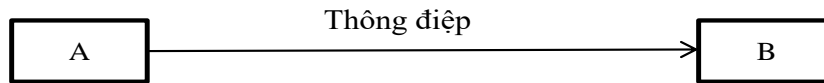
Trong kỹ thuật phân phối và thỏa thuận khóa công khai có sử dụng nhiều giao thức để phân phối khóa. Trong bài giảng này sẽ phân tích làm rõ quy trình hoạt động của chứng chỉ xác thực khóa công khai X.509.

X.509 (v3) là chứng chỉ xác thực khóa công khai được sử dụng phổ biến nhất hiện nay, do ITU đưa ra lần đầu tiên vào năm 1988. Dịch vụ xác thực X.509 được hầu hết các nhà cung cấp PKI hiện nay triển khai. Các thành phần của X.509 bao gồm:

- *Version*: xác định phiên bản của chứng chỉ.
- *Serial Number*: do CA cấp, là định danh duy nhất.
- *Signature Algorithm*: chỉ ra thuật toán CA sử dụng để kí chứng chỉ.
- *Issuer Name*: tên của CA thực hiện cấp chứng chỉ này.
- *Validity Period*: chỉ ra khoảng thời gian chứng chỉ có hiệu lực, gồm 2 giá trị: “not before” – thời gian chứng chỉ bắt đầu có hiệu lực và “not after” – thời gian chứng chỉ hết hiệu lực.
- *Subject Name*: xác định thực thể mà khoá công khai của thực thể này được xác nhận. Tên của subject phải là duy nhất đối với mỗi thực thể được CA xác nhận.
- *Public Key*: chứa khoá công khai và những tham số liên quan; xác định thuật toán (ví dụ RSA) được sử dụng cùng với khoá.
- *Issuer Unique ID*: là trường không bắt buộc, trường này cho phép sử dụng lại tên người cấp. Trường này hiếm được sử dụng trong triển khai thực tế.
- *Subject Unique ID*: là trường tùy chọn, cho phép sử dụng lại tên của Subject khi quá hạn. Trường này cũng hiếm được sử dụng trong thực tế.
- *Extensions*: chỉ có trong chứng chỉ X.509v3.
- *Signature*: gồm có 3 phần. Phần 1 chứa tất cả những trường còn lại của chứng chỉ. Phần 2 chứa bản tóm tắt của phần 1 được mã hóa bằng khóa công khai của CA. Phần 3 gồm các thuật toán được sử dụng trong phần 2.

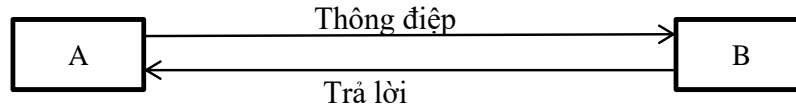
Trong X.509 bao gồm 3 thủ tục xác thực tùy chọn:

- *Xác thực một chiều (One-way Authentication)*: giúp xác nhận danh tính của A và bản tin đúng là từ A gửi đi. Hình 5.10 mô tả cơ chế xác thực này.



**Hình 5. 10. Cơ chế xác thực một chiều**

- *Xác thực hai chiều (Two-way Authentication):* Xác thực thực thể với tem thời gian



**Hình 5. 11. Cơ chế xác thực hai chiều**

Các ký hiệu được sử dụng trong quá trình trao đổi:

- $t_A, t_B$ : tem thời gian do A và B tạo ra, chỉ rõ thời gian hết hạn của thông điệp;
- $r_A, r_B$ : các số ngẫu nhiên sinh bởi A, B và không được dùng lại (để tránh tấn công kiểu phát lại);
- $P_A(k_1), P_B(k_2)$ : bản mã của dữ liệu  $k_1/k_2$ ; được mã hóa bởi khóa công khai của A/B.
- $S_A(D_A), S_B(D_B)$ : chữ ký của A/B, sử dụng khóa riêng của A/B trên dữ liệu  $D_A/D_B$ .
- $cert_A, cert_B$ : chứng chỉ số của A/B, kết hợp khóa công khai của A/B với thông tin định danh của A/B. Khóa công khai của A/B cho phép mã hóa và kiểm tra chữ ký (giải mã).

Các bước thực hiện trong quá trình trao đổi như sau:

- A tạo ra  $D_A$  và gửi cho B thông điệp (1). Các thành phần  $data_1$  và khóa bí mật  $k_1$  là tùy chọn;

$$D_A = (t_A, r_A, B, data_1^*, P_B(k_1)^*)$$

$$A \rightarrow B: cert_A, D_A, S_A(D_A) \quad (1)$$

- B kiểm tra tính xác thực của  $cert_A$  (kiểm tra chữ ký, ngày hết hạn,...), tách lấy khóa công khai của A và kiểm tra chữ ký của A trên khối dữ liệu  $D_A$ . B kiểm tra định danh của nó trong thông điệp (1), tem thời gian  $t_A$  của A, và tham số  $r_A$  có bị lặp (dùng lại) hay không. Nếu tất cả các kiểm tra đều cho kết quả là hợp lệ, B xác nhận việc xác thực A thành công. B giải mã  $P_B(k_1)$  sử dụng khóa riêng của mình và

lưu  $k_1$  làm khóa chia sẻ. Sau đó, B tạo  $D_B$  và gửi cho A thông điệp (2). Các thành phần  $data_2$  và khóa bí mật  $k_2$  là tùy chọn:

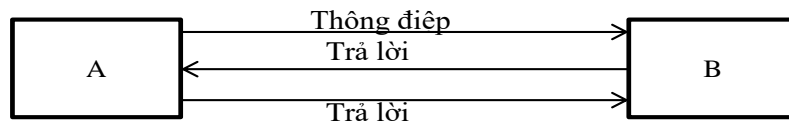
$$D_B = (t_B, r_B, A, r_A, data_2^*, P_A(k_2)^*)$$

$$B \rightarrow A: cert_B, D_B, S_B(D_B) \quad (2)$$

- A sau khi nhận được thông điệp từ B, cũng sẽ tiến hành các thủ tục kiểm tra các thông tin trong thông điệp (2) tương tự B đã thực hiện. Nếu tất cả các kiểm tra đều cho kết quả hợp lệ, A xác nhận việc xác thực B thành công. Sau đó, A lưu khóa  $k_2$  để sử dụng.

**Kết luận:** Như vậy A và B đã xác thực được nhau và cùng chia sẻ khóa  $k_1$  và  $k_2$ .

- *Xác thực ba chiều (Three-way Authentication):* Xác thực thực thể sử dụng giao thức Thách thức – Trả lời.



**Hình 5. 12. Cơ chế xác thực ba chiều**

Nhìn chung, thủ tục xác thực ba chiều giống với xác thực hai chiều. Tuy nhiên, xác thực ba chiều vẫn có một số điểm khác biệt sau:

- Tem thời gian  $t_A, t_B$  được đặt là 0 và không cần kiểm tra;
- Khi nhận được thông điệp (2), A kiểm tra  $r_A$  nhận được phải giống  $r_A$  ban đầu;
- A gửi thông điệp thứ 3 cho B:

$$A \rightarrow B: (r_B, B), S_A(r_B, B) \quad (3)$$

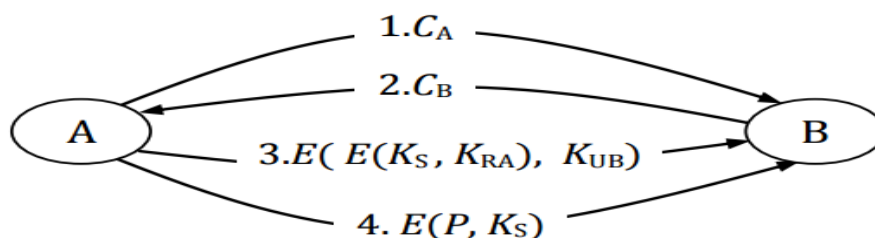
- Khi nhận được thông điệp (3), B kiểm tra chữ ký trùng với chuỗi được tạo từ bản rõ, định danh B phải khớp và tham số  $r_B$  nhận được phải trùng với  $r_B$  gửi đi trong thông điệp (2).

## 5.4. PHÂN PHỐI VÀ THỎA THUẬN KHÓA KẾT HỢP

### 5.4.1. Giới thiệu chung

Phương pháp mã hóa khóa công khai có đặc điểm là thời gian mã hóa và giải mã chậm hơn phương pháp mã hóa khóa đối xứng. Chính vì thế, trong phân phối khóa, có

một phương pháp kết hợp sử dụng khóa công khai để thiết lập khóa bí mật dùng cho mỗi phiên trao đổi dữ liệu. Hình 5.13 mô tả quá trình thiết lập khóa phiên giữa A và B.



**Hình 5. 13. Quá trình thiết lập khóa phiên giữa hai thực thể**

Theo hình 5.13, các bước thực hiện cụ thể như sau:

- A gửi chứng chỉ xác thực  $C_A$  của mình cho B;
- B gửi chứng chỉ xác thực  $C_B$  của mình cho A;
- Sau khi đã xác thực với nhau, A tạo ra 1 khóa phiên  $K_S$  và tiến hành 2 bước mã hóa:
  - Mã hóa  $K_S$  bằng khóa bí mật  $K_{RA}$  của A:  $E(K_S, K_{RA})$ ;
  - Mã hóa  $E(K_S, K_{RA})$  bằng khóa công khai của B:  $E(E(K_S, K_{RA}), K_{UB})$ ;
 A gửi cho B thông điệp vừa được mã hóa.
- B nhận được thông điệp sẽ tiến hành 2 bước giải mã:
  - Dùng khóa bí mật của B để giải mã thông điệp, thu được  $E(K_S, K_{RA})$ ;
  - Dùng khóa công khai của A để tiếp tục giải mã  $E(K_S, K_{RA})$ , thu được khóa phiên  $K_S$ ;

Như vậy, A và B đã có thể dùng khóa phiên  $K_S$  để tiến hành trao đổi thông tin. Sau khi kết thúc việc trao đổi thông tin,  $K_S$  sẽ bị hủy bỏ.

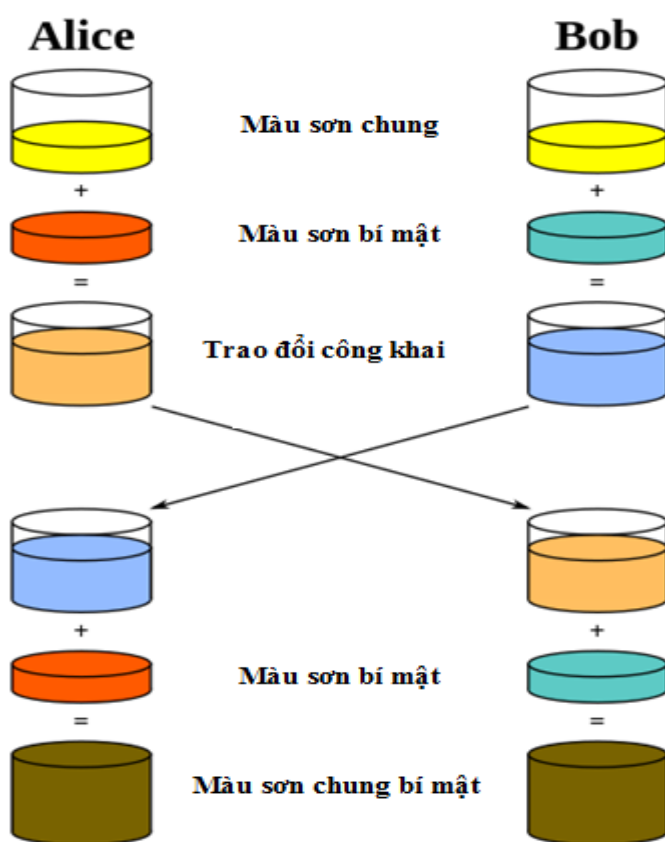
#### 5.4.2. Giao thức Diffie-Hellman

Một trong những giao thức phân phối và thỏa thuận khóa kết hợp là giao thức Diffie-Hellman. Trong tài liệu sẽ tập trung mô tả Diffie-Hellman để làm rõ nguyên tắc phân phối và thỏa thuận khóa của giao thức này. Điểm cần lưu ý là trong một số tài liệu, giao thức Diffie-Hellman được coi là giao thức quản lý và phân phối khóa công khai. Tuy nhiên, trong quy trình quản lý và phân phối khóa, giao thức này có sử dụng khóa công



khai để trao đổi khóa bí mật. Chính vì vậy, có thể coi giao thức Diffie-Hellman là một giao thức phân phối và thỏa thuận khóa kết hợp.

Giao thức Diffie-Hellman được công bố lần đầu tiên bởi *Whitfield Diffie* và *Martin Hellman* vào năm 1976, dù rằng trước đó vài năm nó đã được phát minh một cách độc lập trong cơ quan tình báo Anh, bởi James H. Ellis, Clifford Cocks và Malcolm J. Williamson nhưng được giữ bí mật. Năm 2002, Hellman đề xuất thuật toán nên được gọi là *trao đổi khóa Diffie–Hellman–Merkle* để ghi nhận sự đóng góp của Ralph Merkle trong việc phát minh lĩnh vực mật mã hóa khóa công khai.



Hình 5. 14. Sơ đồ ví dụ về việc trao đổi khóa

Hình 5.14 minh họa ý tưởng cơ bản của việc trao đổi khóa thông qua ví dụ về màu sơn. Điểm chủ chốt của ý tưởng này là Alice và Bob trao đổi màu sơn bí mật thông qua hỗn hợp sơn. Ý tưởng trao đổi giữa Alice và Bob (hình 5.14) thực hiện cụ thể như sau:

- Đầu tiên Alice và Bob trộn màu đã biết chung (màu vàng) với màu bí mật riêng của mỗi người. Sau đó, mỗi người chuyển hỗn hợp của mình tới người kia thông qua một kênh vận chuyển công cộng;

Khi nhận được hỗn hợp của người kia, mỗi người sẽ trộn thêm với màu bí mật của riêng mình và nhận được hỗn hợp cuối cùng. Hỗn hợp sơn cuối cùng là hoàn toàn giống nhau cho cả hai người và chỉ có riêng hai người biết. Mấu chốt ở đây là đối với một người ngoài sẽ rất khó (về mặt tính toán) cho họ để tìm ra được hỗn hợp bí mật chung của hai người (nghĩa là hỗn hợp cuối cùng). Alice và Bob sẽ sử dụng hỗn hợp bí mật chung này để mã hóa và giải mã dữ liệu truyền trên kênh công cộng.

**Lưu ý:** Màu sơn đầu tiên (màu vàng) có thể tùy ý lựa chọn, nhưng được thỏa thuận trước giữa Alice và Bob. Màu sơn này cũng có thể được giả sử là không bí mật đối với người thứ ba mà không làm lộ bí mật chung cuối cùng của Alice và Bob. Giao thức được diễn giải dưới dạng toán học như trong bảng 5.3.

**Bảng 5. 2. Bảng diễn giải giao thức Diffie-Hellman**

Alice				Bob		
Bí mật	Công khai	Tính	Gửi	Tính	Công khai	Bí mật
$a$	$p, g$		$p, g \rightarrow$			$b$
$a$	$p, g, A$	$A = g^a \bmod p$	$A \rightarrow$		$p, g$	$b$
$a$	$p, g, A$		$\leftarrow B$	$B = g^b \bmod p$	$p, g, A, B$	$b$
$a, s$	$p, g, A, B$	$s = B^a \bmod p$		$s = A^b \bmod p$	$p, g, A, B$	$B, s$

Giao thức sử dụng *nhóm nhân số nguyên modulo  $p$* , trong đó  $p$  là số nguyên tố, và  $g$  là *căn nguyên thủy modulo  $p$* .

Các bước thực hiện:

- Đầu tiên, Alice và Bob thỏa thuận dùng chung một số nguyên tố  $p$  và một số căn nguyên thủy  $g$ ;
- Alice chọn số ngẫu nhiên bí mật  $a$  ( $0 \leq a \leq p-2$ ), tính:

$$A = g^a \bmod p$$

Alice gửi cho Bob giá trị  $A$

- Bob chọn số ngẫu nhiên bí mật  $b$  ( $0 \leq b \leq p-2$ ), tính:

$$B = g^b \bmod p$$

Bob gửi cho Alice giá trị B

- Alice tính  $s = B^a \bmod p$
- Bob tính  $s = A^b \bmod p$
- Alice và Bob có khóa chung là  $s$  được sử dụng để trao đổi thông tin.

Trong thực tế, để giao thức được an toàn, người ta sử dụng giá trị rất lớn cho  $a$ ,  $b$ , và  $p$  (trong ví dụ trên chỉ có tổng cộng 23 kết quả khác nhau cho  $n \bmod 23$ ; do đó kẻ tấn công chỉ cần thử hết 23 trường hợp là tìm ra khóa bí mật). Nếu số nguyên tố  $p$  có ít nhất 300 chữ số, còn  $a$  và  $b$  có ít nhất 100 chữ số, thì ngay cả những máy tính hiện đại nhất hiện nay cũng không thể tìm được  $a$  nếu chỉ biết  $g$ ,  $p$ ,  $g^b \bmod p$  và  $g^a \bmod p$ . Đây gọi là bài toán *Lôgarit rời rạc*, hiện chưa có cách giải hiệu quả bằng máy tính (vì vậy nó được sử dụng để tạo khóa công khai). Tham số  $g$  không cần thiết là một căn nguyên thủy có giá trị lớn. Trong thực tế người ta hay sử dụng các giá trị 2, 3 hoặc 5.

*Ví dụ về quy trình trao đổi khóa giữa Alice và Bob:*

- Alice và Bob thỏa thuận sử dụng chung một số nguyên tố  $p=23$  và căn nguyên thủy  $g=5$ .

- Alice chọn một số nguyên bí mật  $a=6$ , và gửi cho Bob giá trị A

$$A = 5^6 \bmod 23 = 8$$

- Bob chọn một số nguyên bí mật  $b=15$ , và gửi cho Alice giá trị B

$$B = 5^{15} \bmod 23 = 19$$

- Alice tính  $s$ :

$$s = 19^6 \bmod 23 = 2$$

- Bob tính  $s$ :

$$s = 8^{15} \bmod 23 = 2$$

- Như vậy, Alice và Bob cùng chia sẻ khóa bí mật chung là  $s=2$ .

Một điểm chú ý trong giao thức Diffie-Hellman là giao thức này không giới hạn việc thỏa thuận khóa chỉ cho hai bên tham gia. Bất kỳ số lượng người sử dụng nào cũng có thể tham gia vào giao thức để tạo khóa bí mật chung bằng cách thực hiện lặp lại các

bước trao đổi thông tin và tính toán trong giao thức. Ví dụ dưới đây sẽ làm rõ hơn về nhận định này.

Alice, Bob và Carol cùng tham gia trao đổi thông tin và sử dụng giao thức Diffie-Hellman để thỏa thuận khóa. Quá trình trao đổi khóa diễn ra như sau (lưu ý: tất cả tính toán dưới đây dựa trên *modulo*  $p$ ):

- Các bên thỏa thuận trước về các tham số  $p$  và  $g$ ;
- Mỗi bên tự tạo khóa bí mật, lần lượt là  $a, b, c$ ;
- Alice tính  $g^a \bmod p$  rồi gửi cho Bob;
- Bob tính  $(g^a)^b \bmod p = g^{ab} \bmod p$  và gửi cho Carol;
- Carol tính  $(g^{ab})^c \bmod p = g^{abc} \bmod p$  và giữ bí mật giá trị này. Đây chính là khóa bí mật chia sẻ, Carol sẽ dùng để trao đổi thông tin với Alice và Bob;
- Bob tính  $g^b \bmod p$  và gửi cho Carol;
- Carol tính  $(g^b)^c \bmod p = g^{bc} \bmod p$  và gửi cho Alice;
- Alice tính  $(g^{bc})^a \bmod p = g^{bca} \bmod p$  và giữ bí mật giá trị này. Đây chính là khóa bí mật chia sẻ, Alice sẽ dùng để trao đổi thông tin với Bob và Carol;
- Carol tính  $g^c \bmod p$  rồi gửi cho Alice;
- Alice tính  $(g^c)^a \bmod p = g^{ca} \bmod p$  và gửi cho Bob;
- Bob tính  $(g^{ca})^b \bmod p = g^{cab} \bmod p$  và giữ bí mật giá trị này. Đây chính là khóa bí mật chia sẻ, Bob sẽ dùng để trao đổi thông tin với Alice và Carol.

Trong thực tế, khi một nhóm nhiều hơn hai người tiến hành trao đổi khóa, sẽ phát sinh một vấn đề, đó là: thứ tự trao đổi khóa giữa các thành viên trong nhóm sẽ diễn ra như thế nào để đảm bảo được tất cả mọi người đều tham gia tạo khóa và biết được khóa chung bí mật?. Để giải quyết vấn đề này, người ta đưa ra 2 phương pháp như sau:

- Phương pháp vòng tròn: Là phương pháp đơn giản và dễ hiểu nhất. Cách thức làm việc của phương pháp này như sau:
  - $N$  người tham gia được sắp xếp theo vòng tròn;
  - $N$  khóa được chuyển theo vòng tròn cho tới khi mỗi khóa được chuyển tới tất cả  $N$  người tham gia (kết thúc với người sở hữu khóa đó);

- Mỗi người tham gia phải thực hiện  $N$  phép tính modulo lũy thừa.
- Phương pháp chia đề trị: Giúp giảm số lượng phép tính modulo lũy thừa của mỗi người xuống, chỉ còn  $\log_2(N) + 1$ . Dưới đây trình bày ví dụ trao đổi khóa trong một hệ thống gồm 8 người ( $N=8$ ):

- Mỗi người A, B, C, D thực hiện một phép tính lũy thừa để tính  $g^{abcd} \bmod p$ . Giá trị này được gửi đến E, F, G, H. Ngược lại, E, F, G, H cũng thực hiện tính  $g^{efgh} \bmod p$  để gửi đến A, B, C, D.
- A và B, mỗi người thực hiện một phép tính lũy thừa để tính  $g^{efghab} \bmod p$  và gửi cho C, D. Ngược lại, C và D cũng làm tương tự để gửi cho A và B giá trị  $g^{efghcd} \bmod p$ . Tương tự, E và F tính  $g^{abcdef} \bmod p$  gửi cho G và H; G và H tính  $g^{abcdgh} \bmod p$  gửi cho E và F
- A thực hiện tính  $g^{efghcda} \bmod p$  để gửi cho B. B cũng tính  $g^{efghcdb} \bmod p$  để gửi cho A. C và D, E và F, G và H cũng thực hiện công việc tương tự.
- A thực hiện phép tính cuối cùng tính  $g^{efghcdba} \bmod p = g^{abcdefgh} \bmod p$  và giữ bí mật giá trị này.

$$B \text{ tính } g^{efghcdab} \bmod p = g^{abcdefgh} \bmod p$$

$$C \text{ tính } g^{efghabcd} \bmod p = g^{abcdefgh} \bmod p$$

$$D \text{ tính } g^{efghabdc} \bmod p = g^{abcdefgh} \bmod p$$

$$E \text{ tính } g^{abcdghfe} \bmod p = g^{abcdefgh} \bmod p$$

$$F \text{ tính } g^{abcdghef} \bmod p = g^{abcdefgh} \bmod p$$

$$G \text{ tính } g^{abcdefhg} \bmod p = g^{abcdefgh} \bmod p$$

$$H \text{ tính } g^{abcdefgh} \bmod p = g^{abcdefgh} \bmod p$$

- Như vậy,  $g^{abcdefgh} \bmod p$  chính là khóa bí mật được chia sẻ giữa 8 người. Mỗi người chỉ cần thực hiện 4 phép tính lũy thừa thay vì 8 phép tính lũy thừa như phương pháp vòng tròn.

## 5.5. CÂU HỎI VÀ BÀI TẬP

- 1) Hãy trình bày khái niệm về quản lý khóa?. Quản lý khóa cung cấp các tính năng gì?. Các mối đe dọa mà quản lý khóa có thể gặp phải?.
- 2) Hãy cho biết đặc điểm chung của các kỹ thuật phân phối và thỏa thuận khóa bí mật?.

- 3) Hãy trình bày về KDC và KTC?. Hãy so sánh 2 kỹ thuật này?.
- 4) Hãy nêu các bước chính trong quá trình trao đổi khóa của giao thức Needham-Schroeder?.
- 5) Hãy nêu các bước chính trong quá trình trao đổi khóa của giao thức Otway-Rees?.
- 6) Hãy trình bày về giao thức không khóa Shamir?.
- 7) Hãy nêu các bước hoạt động chính của giao thức Kerberos?.
- 8) Hãy cho biết đặc điểm chung của các kỹ thuật phân phối khóa công khai?.
- 9) Hãy trình bày về chứng chỉ xác thực khóa công khai X.509?.
- 10) Hãy trình bày về kỹ thuật phân phối và thỏa thuận khóa kết hợp?.
- 11) Hãy nêu các bước chính trong giao thức thỏa thuận khóa Diffie-Hellman?. Các điểm cần lưu ý khi sử dụng giao thức này?.

### TÀI LIỆU THAM KHẢO

- [1] Nguyễn Bình, Ngô Đức Thiện. *Cơ sở mật mã học*. Học Viện Công Nghệ Bưu Chính Viễn Thông, 2013. 237 trang.
- [2] Behrouz A. Forouzan. *Introduction to cryptography and network security*. Moscow, 2010, 784p.
- [3] Nguyễn Khanh Văn. *Cơ sở an toàn thông tin*. Đại học Bách khoa Hà Nội, 2014. 230 trang.
- [4] William Stallings, *Cryptography and Network Security*, Prentice Hall, 2010.
- [5] Jonathan Katz and Yehuda Lindell, *Introduction to Modern Cryptography*, Chapman & Hall/ CRC, 2007.
- [6] Bruce Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, Wiley, 1996.
- [7] Clifford Neuman, Tom Yu, Sam Hartman, Kenneth Raeburn, *The Kerberos Network Authentication Service (V5)*, 2005.
- [8] Bc. Tomáš Král, *Advanced authentication in Java applications using Kerberos protocol*, 2011.
- [9] Refik Molva, Gene Tsudik, Els Van Herreweghenz, Stefano Zattiz, *KryptoKnight Authentication and Key Distribution System*, 1993.
- [10] Ray Bird, Inder Gopal, Amir Herzberg, Phil Janson, Shay Kuttan, Refik Molva, Moti Yung, *The KryptoKnight Family of Light-Weight Protocols for Authentication and Key Distribution*, 1993.

