CHƯƠNG 8
BẢO VỆ DỮ LIỆU - MÃ HÓA

ThS. Trần Bá Nhiệm
Website:
sites.google.com/site/tranbanhiem
Email: tranbanhiem@gmail.com

Nội dung

- Giới thiệu
- Phân tích mã
- Các thuật ngữ
- Mã hóa bất đối xứng – khóa công khai
- RSA
- Mã hóa đối xứng
- Chống tấn công

Giới thiệu

- Nếu không có mã hóa thì bất kỳ ai cũng có thể dễ dàng truy cập vào đường truyền dữ liệu giữa các máy tính để xem, sửa chữa,…
- Bảo mật là vấn đề hết sức quan trọng trong giao dịch thương mại và nhiều kiểu trao đổi thông tin khác

Giới thiệu

- Nội dung trình bày của chương được chia làm 3 phần:
  - Mô tả các phương pháp bẻ khóa bảo mật và chỉ ra bảo mật như thế nào là yếu
  - Mã hóa bất đối xứng: phương pháp được ứng dụng nhiều nhất
  - Mã hóa đối xứng: phương pháp bổ sung, kết hợp với các kiểu khác để tăng cường hiệu quả

Các thuật ngữ

- Plain text: thông tin số chưa được mã hóa
- Cipher text: thông tin số đã được mã hóa
- Key: một mảnh dữ liệu số được dùng bởi chương trình máy tính để mã hóa hoặc giải mã
- Cryptographic algorithm hoặc Cipher: giải thuật để mã hóa hoặc giải mã
- Strength: đo độ khó khi bẻ khóa

An ninh hệ mã hóa

- An ninh vô điều kiện
  - Bản mã không chứa đủ thông tin để xác định duy nhất nguyên bản tương ứng, bất kể với số lượng bao nhiêu và tốc độ máy tính thế nào
- An ninh tính toán
  - Thỏa mãn một trong hai điều kiện
    - Chi phí phá mã vượt quá giá trị thông tin
    - Thời gian phá mã vượt quá tuổi thọ thông tin
  - Thực tế thỏa mãn hai điều kiện
    - Không có nhược điểm
    - Khóa có quá nhiều giá trị không thể thử hết

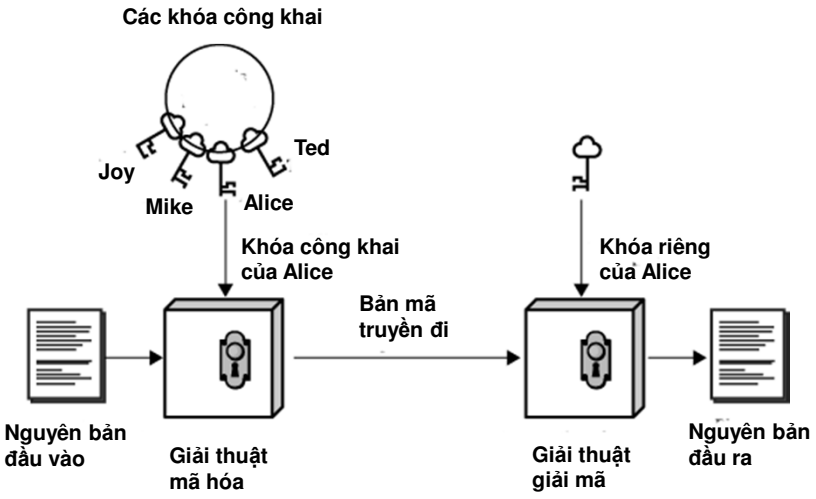
Mã hóa bất đối xứng

- Những hạn chế của mật mã đối xứng
  - Vấn đề phân phối khóa
    - Khó đảm bảo chia sẻ mà không làm lộ khóa bí mật
    - Trung tâm phân phối khóa có thể bị tấn công
  - Không thích hợp cho chữ ký số
    - Bên nhận có thể làm giả thông báo nói nhận được từ bên gửi
- Mật mã khóa công khai đề xuất bởi Whitfield Diffie và Martin Hellman vào năm 1976
  - Khắc phục những hạn chế của mật mã đối xứng
  - Có thể coi là bước đột phá quan trọng nhất trong lịch sử của ngành mật mã
  - Bổ sung chứ không thay thế mật mã đối xứng

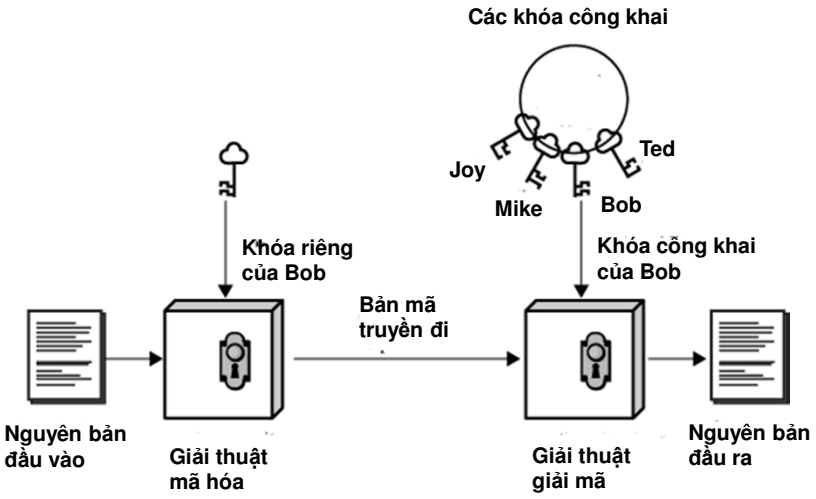
Đặc điểm mật mã khóa công khai

- Còn gọi là mật mã hai khóa hay bất đối xứng
- Các giải thuật khóa công khai sử dụng 2 khóa
  - Một khóa công khai
    - Ai cũng có thể biết
    - Dùng để mã hóa thông báo và thẩm tra chữ ký
  - Một khóa riêng
    - Chỉ nơi giữ được biết
    - Dùng để giải mã thông báo và ký (tạo ra) chữ ký
- Có tính bất đối xứng
  - Bên mã hóa không thể giải mã thông báo
  - Bên thẩm tra không thể tạo chữ ký

Mã hóa khóa công khai



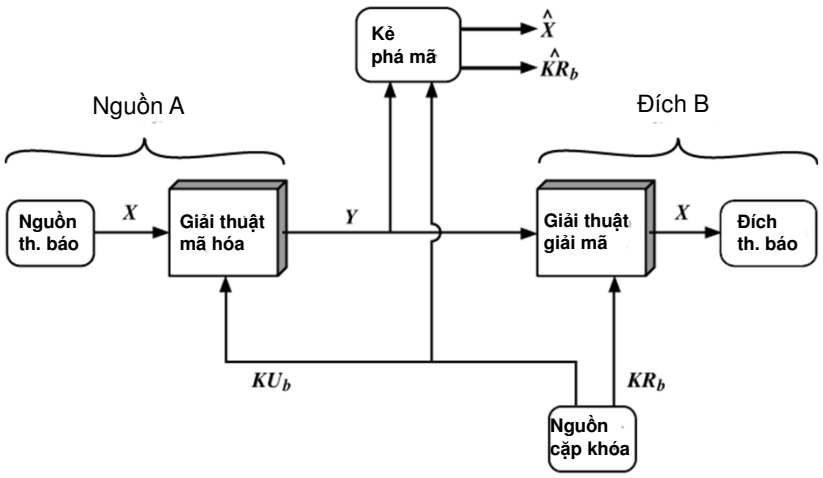
Xác thực



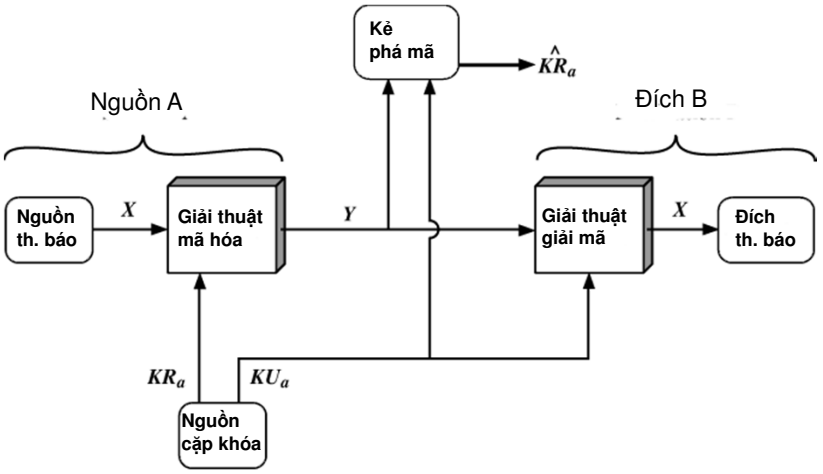
Ứng dụng mật mã khóa công khai

- Có thể phân ra 3 loại ứng dụng
  - Mã hóa/giải mã
    - Đảm bảo sự bí mật của thông tin
  - Chữ ký số
    - Hỗ trợ xác thực văn bản
  - Trao đổi khóa
    - Cho phép chia sẻ khóa phiên trong mã hóa đối xứng
- Một số giải thuật khóa công khai thích hợp cho cả 3 loại ứng dụng; một số khác chỉ có thể dùng cho 1 hay 2 loại

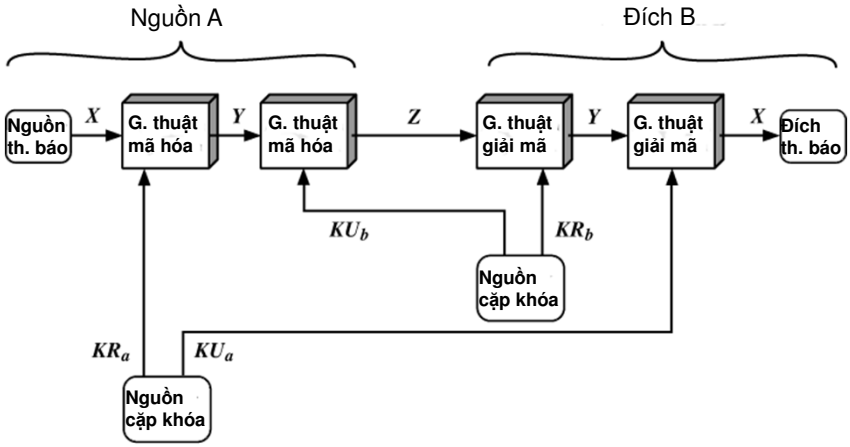
Mô hình đảm bảo bí mật



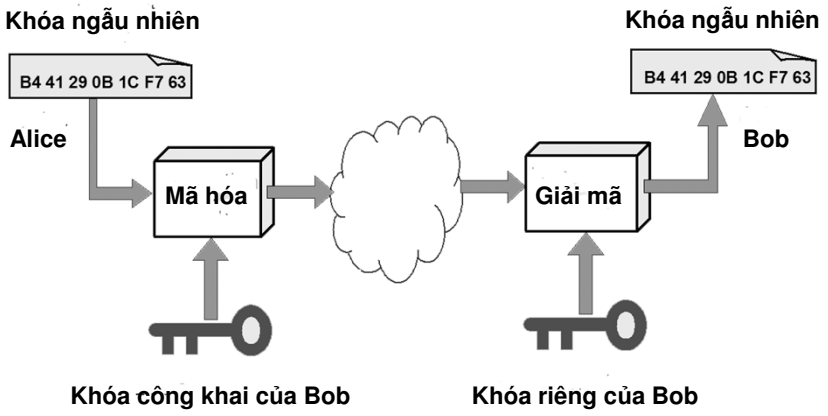
Mô hình xác thực



Mô hình kết hợp



Trao đổi khóa



Các điều kiện cần thiết

- Bên B dễ dàng tạo ra được cặp (KU_b, KR_b)
- Bên A dễ dàng tạo ra được $C = E_{KU_b}(M)$
- Bên B dễ dàng giải mã $M = D_{KR_b}(C)$
- Đối thủ không thể xác định được KR_b khi biết KU_b
- Đối thủ không thể xác định được M khi biết KU_b và C
- Một trong hai khóa có thể dùng mã hóa trong khi khóa kia có thể dùng giải mã
  - $M = D_{KR_b}(E_{KU_b}(M)) = D_{KU_b}(E_{KR_b}(M))$
  - Không thực sự cần thiết

Hệ mã hóa RSA

- Đề xuất bởi Ron Rivest, Adi Shamir và Len Adleman (MIT) vào năm 1977
- Hệ mã hóa khóa công khai phổ dụng nhất
- Mã hóa khối với mỗi khối là một số nguyên < n
  - Thường kích cỡ n là 1024 bit ≈ 309 chữ số thập phân
- Đăng ký bản quyền năm 1983, hết hạn năm 2000
- An ninh vì chi phí phân tích thừa số của một số nguyên lớn là rất lớn

Tạo khóa RSA

- Mỗi bên tự tạo ra một cặp khóa công khai - khóa riêng theo các bước sau:
  - Chọn ngẫu nhiên 2 số nguyên tố đủ lớn $p \neq q$
  - Tính $n = pq$
  - Tính $\Phi(n) = (p-1)(q-1)$
  - Chọn ngẫu nhiên khóa mã hóa e sao cho $1 < e < \Phi(n)$ và $\gcd(e, \Phi(n)) = 1$
  - Tìm khóa giải mã $d \leq n$ thỏa mãn $e.d \equiv 1 \bmod \Phi(n)$
- Công bố khóa mã hóa công khai KU = {e, n}
- Giữ bí mật khóa giải mã riêng KR = {d, n}
  - Các giá trị bí mật p và q bị hủy bỏ

Thực hiện RSA

- Để mã hóa 1 thông báo nguyên bản M, bên gửi thực hiện
  - Lấy khóa công khai của bên nhận KU = {e, n}
  - Tính $C = M^e \bmod n$
- Để giải mã bản mã C nhận được, bên nhận thực hiện
  - Sử dụng khóa riêng KR = {d, n}
  - Tính $M = C^d \bmod n$
- Lưu ý là thông báo M phải nhỏ hơn n
  - Phân thành nhiều khối nếu cần

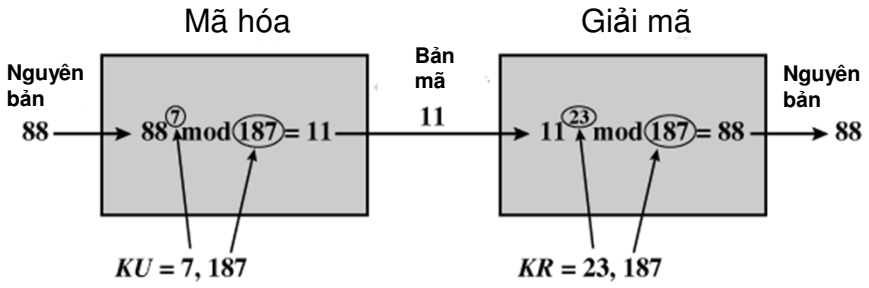
Vì sao RSA khả thi

- Theo định lý Euler
  - $\forall$ a, n: gcd(a, n) = 1 $\Rightarrow a^{\Phi(n)} \bmod n = 1$
  - $\Phi(n)$ là số các số nguyên dương nhỏ hơn n và nguyên tố cùng nhau với n
- Đối với RSA có
  - n = pq với p và q là các số nguyên tố
  - $\Phi(n) = (p - 1)(q - 1)$
  - $ed \equiv 1 \bmod \Phi(n) \Rightarrow \exists$ số nguyên k: $ed = k\Phi(n) + 1$
  - M < n
- Có thể suy ra
  - $C^d \bmod n = M^{ed} \bmod n = M^{k\Phi(n) + 1} \bmod n = M \bmod n = M$

Ví dụ tạo khóa RSA

- Chọn 2 số nguyên tố p = 17 và q = 11
- Tính n = pq = 17 × 11 = 187
- Tính Φ(n) = (p - 1)(q - 1) = 16 × 10 = 160
- Chọn e: gcd(e, 160) = 1 và 1 < e < 160; lấy e = 7
- Xác định d: de ≡ 1 mod 160 và d ≤ 187
  Giá trị d = 23 vì 23 × 7 = 161 = 1 × 160 + 1
- Công bố khóa công khai KU = {7, 187}
- Giữ bí mật khóa riêng KR = {23, 187}
  - Hủy bỏ các giá trị bí mật p = 17 và q = 11

Ví dụ thực hiện RSA



Chọn tham số RSA

- Cần chọn p và q đủ lớn
- Thường chọn e nhỏ
- Thường có thể chọn cùng giá trị của e cho tất cả người dùng
- Trước đây khuyến nghị giá trị của e là 3, nhưng hiện nay được coi là quá nhỏ
- Thường chọn e = $2^{16} - 1 = 65535$
- Giá trị của d sẽ lớn và khó đoán

An ninh của RSA

- Khóa 128 bit là một số giữa 1 và một số rất lớn
  340.282.366.920.938.000.000.000.000.000.000.000.000
- Có bao nhiêu số nguyên tố giữa 1 và số này
  ≈ $n / \ln(n) = 2^{128} / \ln(2^{128})$ ≈
  3.835.341.275.459.350.000.000.000.000.000.000.000
- Cần bao nhiêu thời gian nếu mỗi giây có thể tính được 10^{12} số
  Hơn 121.617.874.031.562.000 năm (khoảng 10 triệu lần tuổi của vũ trụ)
- An ninh nhưng cần đề phòng những điểm yếu

Phá mã RSA

- Phương pháp vét cạn
  - Thử tất cả các khóa riêng có thể
    - Phụ thuộc vào độ dài khóa
- Phương pháp phân tích toán học
  - Phân n thành tích 2 số nguyên tố p và q
  - Xác định trực tiếp $\Phi(n)$ không thông qua p và q
  - Xác định trực tiếp d không thông qua $\Phi(n)$
- Phương pháp phân tích thời gian
  - Dựa trên việc đo thời gian giải mã
  - Có thể ngăn ngừa bằng cách làm nhiễu

Phân tích thừa số RSA

- An ninh của RSA dựa trên độ phức tạp của việc phân tích thừa số n
- Thời gian cần thiết để phân tích thừa số một số lớn tăng theo hàm mũ với số bit của số đó
  - Mất nhiều năm khi số chữ số thập phân của n vượt quá 100 (giả sử làm 1 phép tính nhị phân mất 1 ηs)
- Kích thước khóa lớn đảm bảo an ninh cho RSA
  - Từ 1024 bit trở lên
  - Gần đây nhất năm 1999 đã phá mã được 512 bit (155 chữ số thập phân)

Minh họa RSA

- Với .NET thì RSA được hiện thực trong lớp RSACryptoServiceProvider, nó có thể sinh ra khóa riêng và khóa chung, mã hóa và giải mã bằng các phương thức Encrypt và Decrypt. Các khóa được lưu trữ dưới dạng thức XML
- Khai báo thư viện:
  - using System;
  - using System.Security.Cryptography;

Minh họa RSA

```
public class clsCryptography
{
    private RSACryptoServiceProvider RSA;
    public string PublicKey;
    public string PrivateKey;
    public byte[] Encrypt(byte[] Data, string PublicKeyIn)
    {
        RSA.FromXmlString(PublicKeyIn);
        return RSA.Encrypt(Data, false);
    }
    public byte[] Decrypt(byte[] Data, string PrivateKeyIn)
    {
        RSA.FromXmlString(PrivateKeyIn);
        return RSA.Decrypt(Data, false);
    }
```

Minh họa RSA

```
    public clsCryptography()
    {
      CspParameters cspParams = new CspParameters();
      cspParams.Flags =
    CspProviderFlags.UseMachineKeyStore;
      RSA = new RSACryptoServiceProvider(cspParams);
      PublicKey = RSA.ToXmlString(false);
      PrivateKey = RSA.ToXmlString(true);
    }
}
```

- Toàn bộ công việc mã hóa và giải mã được đóng gói trong lớp này

Minh họa RSA

- Thiết kế form minh họa có 2 textbox và 2 button.
- Khai báo biến form:

```
private rsa.clsCryptography clsRSA = new
rsa.clsCryptography();
private byte[] Decrypted;
private byte[] Encrypted;
```

Minh họa RSA

```
private void Form1_Load(object sender,
EventArgs e)
        {
            tbStatus.Text += "Private key is:\r\n"
+ clsRSA.PrivateKey + "\r\n";
            tbStatus.Text += "Public key is:\r\n" +
clsRSA.PublicKey + "\r\n";
        }
```

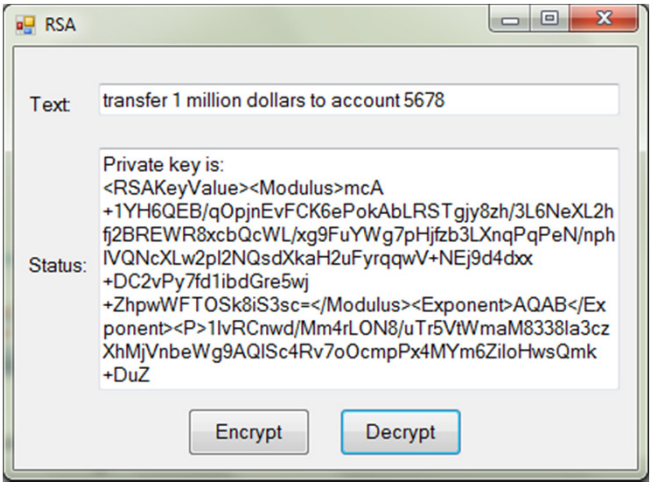

Minh họa RSA

```
private void btnEncrypt_Click(object sender,
EventArgs e)
{
        byte[] PlainText =
System.Text.Encoding.ASCII.GetBytes(tbWorking.Te
xt);
        Encrypted = clsRSA.Encrypt(PlainText,
clsRSA.PublicKey);
        tbWorking.Text =
System.Text.Encoding.ASCII.GetString(Encrypted);
}
```

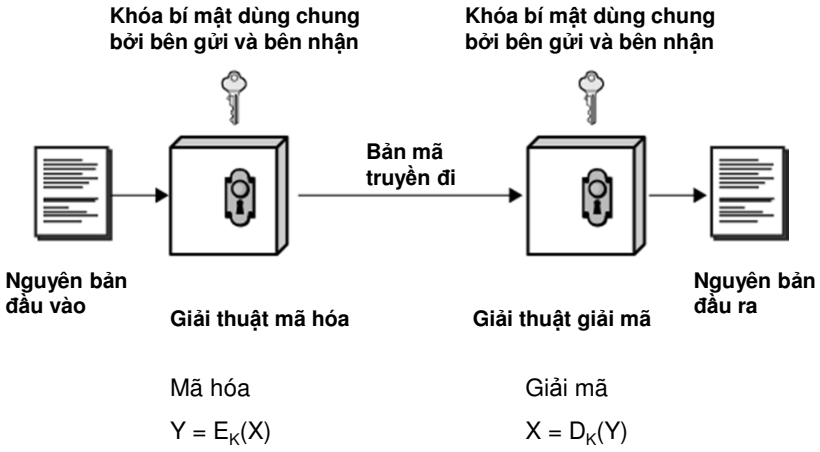
Minh họa RSA

```
private void btnDecrypt_Click(object sender,
EventArgs e)
{
        Decrypted = clsRSA.Decrypt(Encrypted,
clsRSA.PrivateKey);
        tbWorking.Text =
System.Text.Encoding.ASCII.GetString(Decrypt
ed);
}
```


Minh họa RSA



Mô hình hệ mã hóa đối xứng



Mô hình hệ mã hóa đối xứng

- Gồm có 5 thành phần
  - Nguyên bản
  - Giải thuật mã hóa
  - Khóa bí mật
  - Bản mã
  - Giải thuật giải mã
- An ninh phụ thuộc vào sự bí mật của khóa, không phụ thuộc vào sự bí mật của giải thuật

Phá mã

- Là nỗ lực giải mã văn bản đã được mã hóa không biết trước khóa bí mật
- Có hai phương pháp phá mã
  - Vét cạn
    - Thử tất cả các khóa có thể
  - Thám mã
    - Khai thác những nhược điểm của giải thuật
    - Dựa trên những đặc trưng chung của nguyên bản hoặc một số cặp nguyên bản - bản mã mẫu

Phương pháp phá mã vét cạn

- Về lý thuyết có thể thử tất cả các giá trị khóa cho đến khi tìm thấy nguyên bản từ bản mã
- Dựa trên giả thiết có thể nhận biết được nguyên bản cần tìm
- Tính trung bình cần thử một nửa tổng số các trường hợp có thể
- Thực tế không khả khi nếu độ dài khóa lớn

Thời gian tìm kiếm trung bình

Kích thước khóa (bit)	Số lượng khóa	Thời gian cần thiết (1 giải mã/μs)	Thời gian cần thiết (10^6 giải mã/μs)
32	$2^{32} = 4{,}3 \times 10^9$	2^{31} μs = 35,8 phút	2,15 ms
56	$2^{56} = 7{,}2 \times 10^{16}$	2^{55} μs = 1142 năm	10,01 giờ
128	$2^{128} = 3{,}4 \times 10^{38}$	2^{127} μs = $5{,}4 \times 10^{24}$ năm	$5{,}4 \times 10^{18}$ năm
168	$2^{168} = 3{,}7 \times 10^{50}$	2^{167} μs = $5{,}9 \times 10^{36}$ năm	$5{,}9 \times 10^{30}$ năm
26 ký tự (hoán vị)	$26! = 4 \times 10^{26}$	2×10^{26} μs = $6{,}4 \times 10^{12}$ năm	$6{,}4 \times 10^6$ năm

Khóa DES dài 56 bit
Khóa AES dài 128+ bit
Khóa 3DES dài 168 bit

Tuổi vũ trụ: ~ 10^{10} năm

Các kỹ thuật thám mã

- Chỉ có bản mã
  - Chỉ biết giải thuật mã hóa và bản mã hiện có
- Biết nguyên bản
  - Biết thêm một số cặp nguyên bản - bản mã
- Chọn nguyên bản
  - Chọn 1 nguyên bản, biết bản mã tương ứng
- Chọn bản mã
  - Chọn 1 bản mã, biết nguyên bản tương ứng
- Chọn văn bản
  - Kết hợp chọn nguyên bản và chọn bản mã

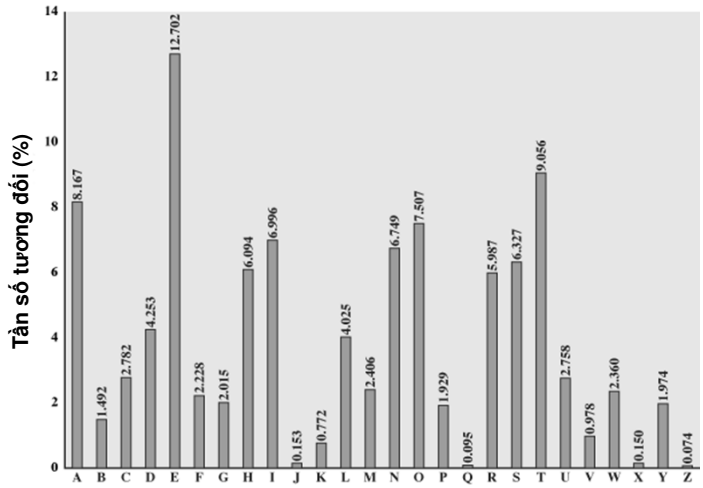
Phân tích mã

- Để nhận thức đầy đủ mã hóa là gì, chúng ta cần hiểu sự khác biệt giữa các phương pháp mã hóa tốt và không tốt
- Khi dùng kỹ thuật mã hóa không tốt, kết quả còn tệ hơn không mã hóa vì hệ thống sẽ tin nhầm vào việc mã hóa đó, trong khi thực sự nó chẳng có bảo mật gì cả

Phân tích mã

- Bất kỳ giải thuật mã hóa nào thay thế một ký tự bằng ký tự khác đều có thể bị bẻ mà không cần biết khóa hoặc thậm chí cơ chế mà văn bản được mã hóa. Quá trình này gọi là phân tích tần số.
- Ký tự được dùng nhiều nhất trong tiếng Anh là khoảng trắng (mã ASCII là 32), tiếp theo là “e”, “t”,…, “z”

Các tần số chữ cái tiếng Anh



Phân tích mã

- Với phương pháp thay thế, tần số xuất hiện của ký tự thay thế vẫn giống với ký tự gốc → dễ dàng suy đoán
- Ví dụ: cho đoạn văn bản đã được mã hóa

  v`z/bnv/a`{/c`na/}ja{/cjn|j/cjak/`}/`{gj}xf|j/{}na|ij}/{g j/`{gj}/bjkfzb/{`/na`{gj}/z|j}/jwlj {/n|/ n}{/`i/{gj/ j}bnaja{/{}na|ij}/n|/ }`yfkjk/nm`yj/`i/{gj/|`i{xn}j/ }`kzl{

Phân tích mã

- Ký tự có xác suất xuất hiện cao nhất là "/", tạm gán cho " ", kế đến "j" gán cho "e", tương tự như thế cho đến "z". Kết quả sơ chế lần thứ 1 được:

  fou cif not moin aent meise mend oa otheagwse tainsrea the othea cedwuc to inothea usea ebpelt is liat or the leacinent tainsrea is laoywded ivoye or the sortgiae laodupt

Phân tích mã

- Nhìn vào kết quả bước 1, có một số từ đã tạm chấp nhận, một số cần tinh chế lần 2.
  - othea → other: a = r, r = ?
  - o? → on, of: giả sử "not" là đúng, nên r = f, f = ?
  - ?ou → you: f = y, "y" không có trong văn bản mã hóa
- Tiến trình trên có thể lặp lại vài lần

Phân tích mã

- Kết quả sơ chế lần sau được:

  you ciy not moin rent meise mend or othergwse trinsfer the other cedwuc to inother user ebpelt is lirt of the lercinent trinsfer is lroywded ivoye of the softgire lrodupt

- Tiếp tục tinh chế:
  - trinsfer → transfer: i = a
  - softgare → software: g = w, w = ?
  - otherw?se → otherwise: w = l

Phân tích mã

- Kết quả sơ chế lần sau được:

  you cay not moan rent mease mend or otherwise transfer the other cediuc to another user ebpelt as lart of the lercanent transfer as lroyided avoye of the software lrodupt

- Tiếp tục tinh chế:
  - cediuc → medium : c = m
  - ?ermanent → permanent : l = p, p =?
  - mease → lease : m = l

Phân tích mã

- Kết quả sơ chế lần sau được:

  you may not loan rent lease lend or otherwise transfer the other medium to another user eb?ept as part of the permanent transfer as proyided avoye of the software produ?t

- Tiếp tục tinh chế:
  - produ?t → product : p = c
  - ebcept → except : b = x
  - proyided → provided : y = v
  - avove → above: v = b

Phân tích mã

- Kết quả sơ chế lần sau được:

  you may not loan rent lease lend or otherwise transfer the other medium to another user except as part of the permanent transfer as provided above of the software product

- Phần mềm kiểu phân tích tần số có thể chạy mà không cần sự can thiệp của con người và dễ dàng ghi nhận, giải mã các file cũng như dữ liệu trên mạng

Mã hóa thay thế cổ điển

- Các chữ cái của nguyên bản được thay thế bởi các chữ cái khác, hoặc các số, hoặc các ký hiệu
- Nếu nguyên bản được coi như một chuỗi bit thì thay thế các mẫu bit trong nguyên bản bằng các mẫu bit của bản mã

Hệ mã hóa Caesar

- Là hệ mã hóa thay thế xuất hiện sớm nhất và đơn giản nhất
- Sử dụng đầu tiên bởi Julius Caesar vào mục đích quân sự
- Dịch chuyển xoay vòng theo thứ tự chữ cái
  - Khóa k là số bước dịch chuyển
  - Với mỗi chữ cái của văn bản
    - Đặt p = 0 nếu chữ cái là a, p = 1 nếu chữ cái là b,...
    - Mã hóa: C = E(p) = (p + k) mod 26
    - Giải mã: p = D(C) = (C - k) mod 26
- Ví dụ: Mã hóa "meet me after class" với k = 3. Kết quả: "phhw ph diwhu fodvv"

Phá mã hệ mã hóa Caesar

- Phương pháp vét cạn
  - Khóa chỉ là một chữ cái (hay một số giữa 1 và 25)
  - Thử tất cả 25 khóa có thể
  - Dễ dàng thực hiện
- Ba yếu tố quan trọng
  - Biết trước các giải thuật mã hóa và giải mã
  - Chỉ có 25 khóa để thử
  - Biết và có thể dễ dàng nhận ra được ngôn ngữ của nguyên bản
- Ví dụ: Phá mã "GCUA VQ DTGCM"

Hệ mã hóa đơn bảng

- Thay một chữ cái này bằng một chữ cái khác theo trật tự bất kỳ sao cho mỗi chữ cái chỉ có một thay thế duy nhất và ngược lại
- Khóa dài 26 chữ cái
- Ví dụ
  - Khóa

    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

    M N B V C X Z A S D F G H J K L P O I U Y T R E W Q
  - Nguyên bản: i love you
  - Mã hóa thành: s gktc wky

Phá mã hệ mã hóa đơn bảng

- Phương pháp vét cạn
  - Khóa dài 26 ký tự
  - Số lượng khóa có thể = 26! = 4 x 10^{26}
  - Rất khó thực hiện
- Khai thác những nhược điểm của giải thuật
  - Biết rõ tần số các chữ cái tiếng Anh
    - Có thể suy ra các cặp chữ cái nguyên bản - chữ cái bản mã
    - Ví dụ: chữ cái xuất hiện nhiều nhất có thể tương ứng với 'e'
  - Có thể nhận ra các bộ đôi và bộ ba chữ cái
    - Ví dụ bộ đôi: 'th', 'an', 'ed'
    - Ví dụ bộ ba: 'ing', 'the', 'est'

Ví dụ phá mã hệ đơn bảng

- Cho bản mã
  UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
  VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
  EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
- Tính tần số chữ cái tương đối
- Đoán P là e, Z là t
- Đoán ZW là th và ZWP là the
- Tiếp tục đoán và thử, cuối cùng được
  it was disclosed yesterday that several informal but
  direct contacts have been made with political
  representatives of the viet cong in moscow

Hệ mã hóa Playfair (1)

- Là một hệ mã hóa nhiều chữ
  - Giảm bớt tương quan cấu trúc giữa bản mã và nguyên bản bằng cách mã hóa đồng thời nhiều chữ cái của nguyên bản
- Phát minh bởi Charles Wheatstone vào năm 1854, lấy tên người bạn Baron Playfair
- Sử dụng 1 ma trận chữ cái 5x5 xây dựng trên cơ sở 1 từ khóa
  - Điền các chữ cái của từ khóa (bỏ các chữ trùng)
  - Điền nốt ma trận với các chữ khác của bảng chữ cái
  - I và J chiếm cùng một ô của ma trận

Hệ mã hóa Playfair (2)

- Ví dụ ma trận với từ khóa MONARCHY

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 2 chữ cái một lúc
  - Nếu 2 chữ giống nhau, tách ra bởi 1 chữ điền thêm
  - Nếu 2 chữ nằm cùng hàng, thay bởi các chữ bên phải
  - Nếu 2 chữ nằm cùng cột, thay bởi các chữ bên dưới
  - Các trường hợp khác, mỗi chữ cái được thay bởi chữ cái khác cùng hàng, trên cột chữ cái cùng cặp

Phá mã hệ mã hóa Playfair

- An ninh đảm bảo hơn nhiều hệ mã hóa đơn chữ
- Có 26 x 26 = 676 cặp chữ cái
  - Việc giải mã từng cặp khó khăn hơn
  - Cần phân tích 676 tần số xuất hiện thay vì 26
- Từng được quân đội Anh, Mỹ sử dụng rộng rãi
- Bản mã vẫn còn lưu lại nhiều cấu trúc của nguyên bản
- Vẫn có thể phá mã được vì chỉ có vài trăm cặp chữ cái cần giải mã

Hệ mã hóa Vigenère

- Là một hệ mã hóa đa bảng
  - Sử dụng nhiều bảng mã hóa
  - Khóa giúp chọn bảng tương ứng với mỗi chữ cái
- Kết hợp 26 hệ Ceasar (bước dịch chuyển 0 - 25)
  - Khóa K = $k_1k_2...k_d$ gồm d chữ cái sử dụng lặp đi lặp lại với các chữ cái của văn bản
  - Chữ cái thứ i tương ứng với hệ Ceasar bước chuyển i
- Ví dụ
  - Khóa: deceptivedeceptivedeceptive
  - Nguyên bản: wearediscoveredsaveyourself
  - Bản mã: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Phá mã hệ mã hóa Vigenère

- Phương pháp vét cạn
  - Khó thực hiện, nhất là nếu khóa gồm nhiều chữ cái
- Khai thác những nhược điểm của giải thuật
  - Cấu trúc của nguyên bản được che đậy tốt hơn hệ Playfair nhưng không hoàn toàn biến mất
  - Chỉ việc tìm độ dài khóa sau đó phá mã từng hệ Ceasar
  - Cách tìm độ dài khóa
    - Nếu độ dài khóa nhỏ so với độ dài văn bản, có thể phát hiện 1 dãy văn bản lặp lại nhiều lần
    - Khoảng cách giữa 2 dãy văn bản lặp là 1 bội số của độ dài khóa
    - Từ đó suy ra độ dài khóa

Hệ mã hóa khóa tự động

- Vigenère đề xuất từ khóa không lặp lại mà được gắn vào đầu nguyên bản
  - Nếu biết từ khóa sẽ giải mã được các chữ cái đầu tiên
  - Sử dụng các chữ cái này làm khóa để giải mã các chữ cái tiếp theo,...
- Ví dụ:
  - Khóa: deceptivewearediscoveredsav
  - Nguyên bản: wearediscoveredsaveyourself
  - Mã hóa: ZICVTWQNGKZEIIGASXSTSLVVWLA
- Vẫn có thể sử dụng kỹ thuật thống kê để phá mã
  - Khóa và nguyên bản có cùng tần số các chữ cái

Độn một lần

- Là hệ mã hóa thay thế không thể phá được
- Đề xuất bởi Joseph Mauborgne
- Khóa ngẫu nhiên, độ dài bằng độ dài văn bản, chỉ sử dụng một lần
- Giữa nguyên bản và bản mã không có bất kỳ quan hệ nào về thống kê
- Với bất kỳ nguyên bản và bản mã nào cũng tồn tại một khóa tương ứng
- Khó khăn ở việc tạo khóa và đảm bảo phân phối khóa an ninh

Mã hóa hoán vị cổ điển

- Che đậy nội dung văn bản bằng cách sắp xếp lại trật tự các chữ cái
- Không thay đổi các chữ cái của nguyên bản
- Bản mã có tần số xuất hiện các chữ cái giống như nguyên bản

Hệ mã hóa hàng rào

- Viết các chữ cái theo đường chéo trên một số hàng nhất định
- Sau đó đọc theo từng hàng một
- Ví dụ
  - Nguyên bản: attack at midnight
  - Mã hóa với độ cao hàng rào là 2

    a t c a m d i h

    t a k t i n g t
  - Bản mã: ATCAMDIHTAKTINGT

Hệ mã hóa hàng

- Viết các chữ cái theo hàng vào 1 số cột nhất định
- Sau đó hoán vị các cột trước khi đọc theo cột
- Khóa là thứ tự đọc các cột
- Ví dụ
  - Khóa: 4 3 1 2 5 6 7
  - Nguyên bản: a t t a c k p

    o s t p o n e

    d u n t i l t

    w o a m x y z
  - Bản mã:

    TTNAAPTMTSUOAODWCOIXKNLYPETZ

Mã hóa tích hợp

- Các hệ mã hóa thay thế và hoán vị không an toàn vì những đặc điểm của ngôn ngữ
- Kết hợp sử dụng nhiều hệ mã hóa sẽ khiến việc phá mã khó hơn
  - Hai thay thế tạo nên một thay thế phức tạp hơn
  - Hai hoán vị tạo nên một hoán vị phức tạp hơn
  - Một thay thế với một hoán vị tạo nên một hệ mã hóa phức tạp hơn nhiều
- Là cầu nối từ các hệ mã hóa cổ điển đến các hệ mã hóa hiện đại

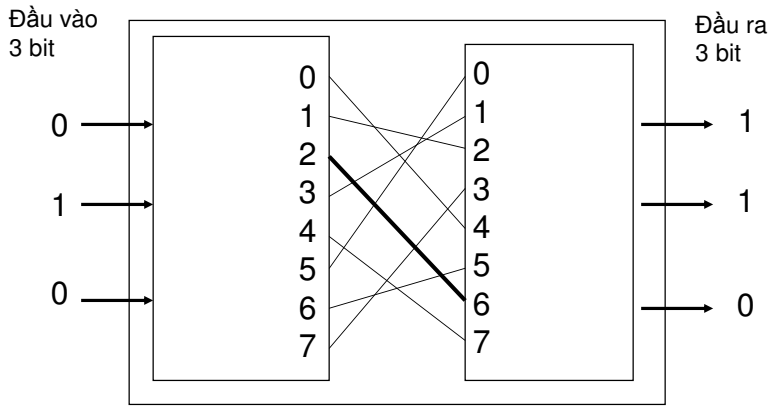
Mã hóa khối

- So với mã hóa luồng
  - Mã hóa khối xử lý thông báo theo từng khối
  - Mã hóa luồng xử lý thông báo 1 bit hoặc 1 byte mỗi lần
- Giống như thay thế các ký tự rất lớn ($\geq$ 64 bit)
  - Bảng mã hóa gồm 2^n đầu vào (n là độ dài khối)
  - Mỗi khối đầu vào ứng với một khối mã hóa duy nhất
    - Tính thuận nghịch
  - Độ dài khóa là n x 2^n bit quá lớn
- Xây dựng từ các khối nhỏ hơn
- Hầu hết các hệ mã hóa khối đối xứng dựa trên cấu trúc hệ mã hóa Feistel

Mạng S-P

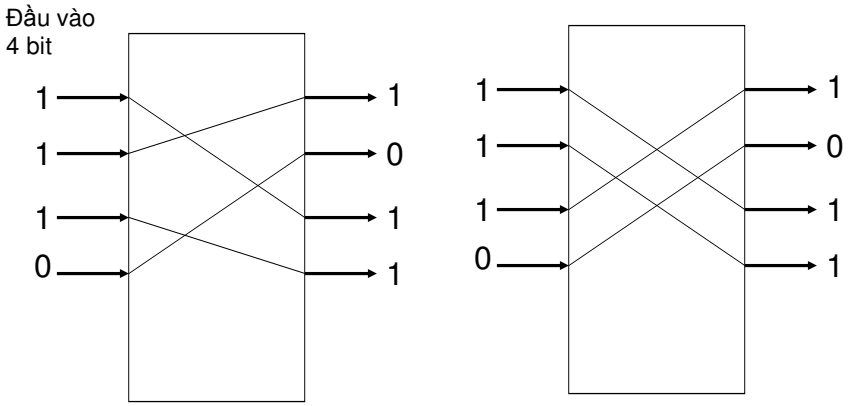
- Mạng thay thế (S) - hoán vị (P) đề xuất bởi Claude Shannon vào năm 1949
- Là cơ sở của các hệ mã hóa khối hiện đại
- Dựa trên 2 phép mã hóa cổ điển
  - Phép thay thế: Hộp S
  - Phép hoán vị: Hộp P
- Đan xen các chức năng
  - Khuếch tán: Hộp P (kết hợp với hộp S)
    - Phát tỏa cấu trúc thống kê của nguyên bản khắp bản mã
  - Gây lẫn: Hộp S
    - Làm phức tạp hóa mối quan hệ giữa bản mã và khóa

Hộp S



Lưu ý: Hộp S có tính thuận nghịch

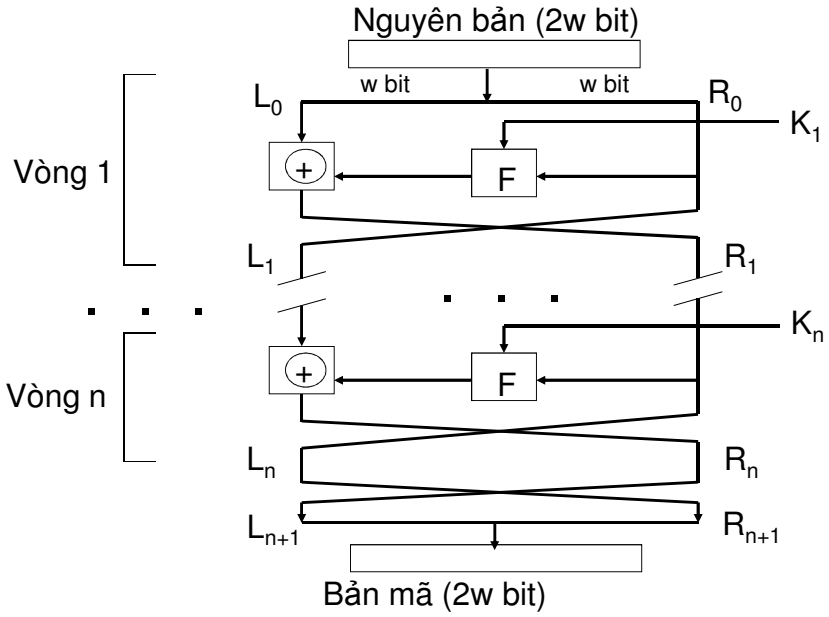
Hộp P



Lưu ý: Hộp P có tính thuận nghịch

Mã hóa Feistel

- Đề xuất bởi Horst Feistel dựa trên khái niệm hệ mã hóa tích hợp thuận nghịch của Shannon
- Phân mỗi khối dài 2w bit thành 2 nửa L_0 và R_0
- Xử lý qua n vòng
- Chia khóa K thành n khóa con K_1, K_2,..., K_n
- Tại mỗi vòng i
  - Thực hiện thay thế ở nửa bên trái L_{i-1} bằng cách XOR nó với $F(K_i, R_{i-1})$
  - F thường gọi là hàm chuyển đổi hay hàm vòng
  - Hoán vị hai nửa L_i và R_i



Các đặc trưng hệ Feistel

- Độ dài khối
  - Khối càng lớn càng an ninh (thường 64 bit)
- Độ dài khóa
  - Khóa càng dài càng an ninh (thường 128 bit)
- Số vòng
  - Càng nhiều vòng càng an ninh (thường 16 vòng)
- Giải thuật sinh mã con
  - Càng phức tạp càng khó phá mã
- Hàm vòng
  - Càng phức tạp càng khó phá mã
- Ảnh hưởng đến cài đặt và phân tích

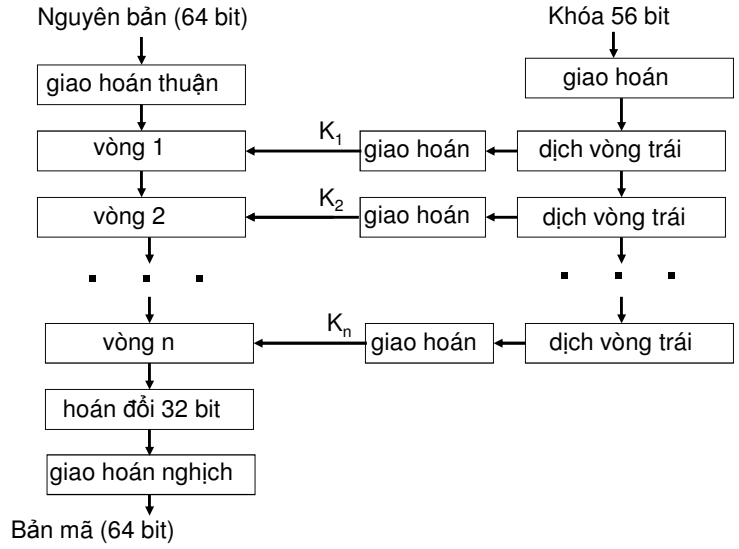
Giải mã Feistel

- Giống giải thuật mã hóa, chỉ khác
  - Bản mã là dữ liệu đầu vào
  - Các khóa con được dùng theo thứ tự ngược lại
- Tại mỗi vòng kết quả đầu ra chính là các dữ liệu đầu vào của quá trình mã hóa
  - Đối với quá trình mã hóa
    - $L_i = R_{i-1}$
    - $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$
  - Đối với quá trình giải mã
    - $R_{i-1} = L_i$
    - $L_{i-1} = R_i \oplus F(L_i, K_i)$

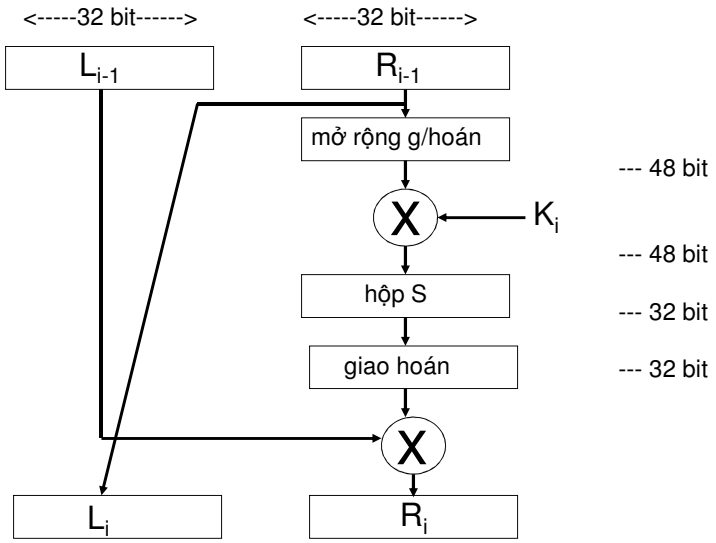
Chuẩn mã hóa dữ liệu

- DES (Data Encryption Standard) được công nhận chuẩn năm 1977
- Phương thức mã hóa được sử dụng rộng rãi nhất
- Tên giải thuật là DEA (Data Encryption Algorithm)
- Là một biến thể của hệ mã hóa Feistel, bổ sung thêm các hoán vị đầu và cuối
- Kích thước khối: 64 bit
- Kích thước khóa: 56 bit
- Số vòng: 16
- Từng gây nhiều tranh cãi về độ an ninh

Giải thuật mã hóa DES



Một vòng DES



Phá mã DES

- Khóa 56 bit có $2^{56} = 7{,}2 \times 10^{16}$ giá trị có thể
- Phương pháp vét cạn tỏ ra không thực tế
- Tốc độ tính toán cao có thể phá được khóa
  - 1997: 70000 máy tính phá mã DES trong 96 ngày
  - 1998: Electronic Frontier Foundation (EFF) phá mã DES bằng máy chuyên dụng (250000$) trong < 3 ngày
  - 1999: 100000 máy tính phá mã trong 22 giờ
- Vấn đề còn phải nhận biết được nguyên bản
- Thực tế DES vẫn được sử dụng không có vấn đề
- Nếu cần an ninh hơn: 3DES hay chuẩn mới AES

Hệ mã hóa 3DES

- Sử dụng 3 khóa và chạy 3 lần giải thuật DES
  - Mã hóa: $C = E_{K_3}[D_{K_2}[E_{K_1}[p]]]$
  - Giải mã: $p = D_{K_1}[E_{K_2}[D_{K_3}[C]]]$
- Độ dài khóa thực tế là 168 bit
  - Không tồn tại $K_4 = 56$ sao cho $C = E_{K_4}(p)$
- Vì sao 3 lần? tránh tấn công "gặp nhau ở giữa"
  - $C = E_{K_2}(E_{K_1}(p)) \Rightarrow X = E_{K_1}(p) = D_{K_2}(C)$
  - Nếu biết một cặp (p, C)
    - Mã hóa p với 2^{56} khóa và giải mã C với 2^{56} khóa
    - So sánh tìm ra K_1 và K_2 tương ứng
    - Kiểm tra lại với 1 cặp (p, C) mới; nếu OK thì K_1 và K_2 là khóa

Chuẩn mã hóa tiên tiến

- AES (Advanced Encryption Standard) được công nhận chuẩn mới năm 2001
- Tên giải thuật là Rijndael (Rijmen + Daemen)
- An ninh hơn và nhanh hơn 3DES
- Kích thước khối: 128 bit
- Kích thước khóa: 128/192/256 bit
- Số vòng: 10/12/14
- Cấu trúc mạng S-P, nhưng không theo hệ Feistel
  - Không chia mỗi khối làm đôi

Các hệ mã hóa khối khác (1)

- IDEA (International Data Encryption Algorithm)
  - Khối 64 bit, khóa 128 bit, 8 vòng
  - Theo cấu trúc mạng S-P, nhưng không theo hệ Feistel
    - Mỗi khối chia làm 4
  - Rất an ninh
  - Bản quyền bởi Ascom nhưng dùng miễn phí
- Blowfish
  - Khối 64 bit, khóa 32-448 bit (ngầm định 128 bit), 16 vòng
  - Theo cấu trúc hệ Feistel
  - An ninh, khá nhanh và gọn nhẹ
  - Tự do sử dụng

Các hệ mã hóa khối khác (2)

- RC5
  - Phát triển bởi Ron Rivest
  - Khối 32/64/128 bit, khóa 0-2040 bit, 0-255 vòng
  - Đơn giản, thích hợp các bộ xử lý có độ rộng khác nhau
  - Theo cấu trúc hệ Feistel
- CAST-128
  - Phát triển bởi Carlisle Adams và Stafford Tavares
  - Khối 64 bit, khóa 40-128 bit, 12/16 vòng
  - Có 3 loại hàm vòng dùng xen kẽ
  - Theo cấu trúc hệ Feistel
  - Bản quyền bởi Entrust nhưng dùng miễn phí

Các phương thức mã hóa khối

- ECB (Electronic Codebook)
  - Mã hóa từng khối riêng rẽ
- CBC (Cipher Block Chaining)
  - Khối nguyên bản hiện thời được XOR với khối bản mã trước đó
- CFB (Cipher Feedback)
  - Mô phỏng mã hóa luồng (đơn vị s bit)
    - s bit mã hóa trước được đưa vào thanh ghi đầu vào hiện thời
- OFB (Output Feeback)
    - s bit trái đầu ra trước được đưa vào thanh ghi đầu vào hiện thời
- CTR (Counter)
  - XOR mỗi khối nguyên bản với 1 giá trị thanh đếm mã hóa

Bố trí công cụ mã hóa

- Giải pháp hữu hiệu và phổ biến nhất chống lại các mối đe dọa đến an ninh mạng là mã hóa
- Để thực hiện mã hóa, cần xác định
  - Mã hóa những gì
  - Thực hiện mã hóa ở đâu
- Có 2 phương án cơ bản
  - Mã hóa liên kết
  - Mã hóa đầu cuối

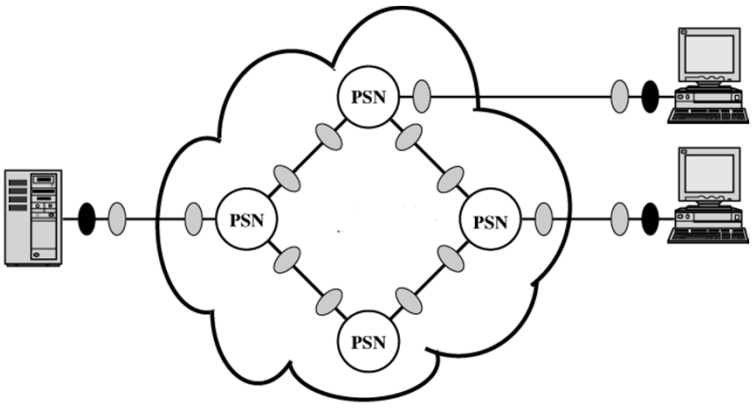
Mã hóa liên kết

- Công cụ mã hóa được sắp đặt ở 2 đầu của mọi liên kết có nguy cơ bị tấn công
- Đảm bảo an ninh việc lưu chuyển thông tin trên tất cả các liên kết mạng
- Các mạng lớn cần đến rất nhiều công cụ mã hóa
- Cần cung cấp rất nhiều khóa
- Nguy cơ bị tấn công tại mỗi chuyển mạch
  - Các gói tin cần được mã hóa mỗi khi đi vào một chuyển mạch gói để đọc được địa chỉ ở phần đầu
- Thực hiện ở tầng vật lý hoặc tầng liên kết

Mã hóa đầu cuối

- Quá trình mã hóa được thực hiện ở 2 hệ thống đầu cuối
- Đảm bảo an ninh dữ liệu người dùng
- Chỉ cần một khóa cho 2 đầu cuối
- Đảm bảo xác thực ở mức độ nhất định
- Mẫu lưu chuyển thông tin không được bảo vệ
  - Các phần đầu gói tin cần được truyền tải tường minh
- Thực hiện ở tầng mạng trở lên
  - Càng lên cao càng ít thông tin cần mã hóa và càng an ninh nhưng càng phức tạp với nhiều thực thể và khóa

Kết hợp các phương án mã hóa



Công cụ mã hóa đầu cuối

Công cụ mã hóa liên kết

PSN: Packet-switching node

Quản lý khóa bí mật

- Vấn đề đối với mã hóa đối xứng là làm sao phân phối khóa an ninh đến các bên truyền tin
  - Thường hệ thống mất an ninh là do không quản lý tốt việc phân phối khóa bí mật
- Phân cấp khóa
  - Khóa phiên (tạm thời)
    - Dùng mã hóa dữ liệu trong một phiên kết nối
    - Hủy bỏ khi hết phiên
  - Khóa chủ (lâu dài)
    - Dùng để mã hóa các khóa phiên, đảm bảo phân phối chúng một cách an ninh

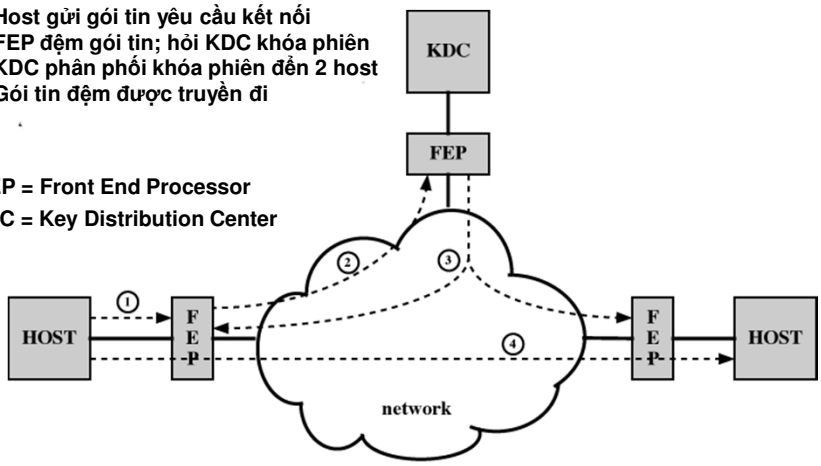
Các cách phân phối khóa

- Khóa có thể được chọn bởi bên A và gửi theo đường vật lý đến bên B
- Khóa có thể được chọn bởi một bên thứ ba, sau đó gửi theo đường vật lý đến A và B
- Nếu A và B đã có một khóa dùng chung thì một bên có thể gửi khóa mới đến bên kia, sử dụng khóa cũ để mã hóa khóa mới
- Nếu mỗi bên A và B đều có một kênh mã hóa đến một bên thứ ba C thì C có thể gửi khóa theo các kênh mã hóa đó đến A và B

Phân phối khóa tự động

1. Host gửi gói tin yêu cầu kết nối
2. FEP đệm gói tin; hỏi KDC khóa phiên
3. KDC phân phối khóa phiên đến 2 host
4. Gói tin đệm được truyền đi

FEP = Front End Processor
KDC = Key Distribution Center



Minh họa DES

- Thiết kế form minh họa có 2 textbox và 3 button.
- Khai báo thư viện:
  - using System.Security.Cryptography;
  - using System.IO;

Minh họa DES

```
private void btnEncrypt_Click(object sender, EventArgs e)
      {
          string encFile = tbFile.Text + ".enc";
          FileStream fs = new FileStream(encFile,
FileMode.Create, FileAccess.Write);
          StreamReader sr = new StreamReader(tbFile.Text);
          string strinput = (sr).ReadToEnd();
          sr.Close();
          byte[] bytearrayinput =
Encoding.Default.GetBytes(strinput);
          des = new DESCryptoServiceProvider();
          ICryptoTransform desencrypt = des.CreateEncryptor();
          CryptoStream cryptostream = new CryptoStream(fs,
desencrypt, CryptoStreamMode.Write);
```


Minh họa DES

```
 cryptostream.Write(bytearrayinput, 0, bytearrayinput.Length);
cryptostream.Close();
          fs.Close();
          fs = new FileStream(encFile, FileMode.Open,
FileAccess.Read);
          sr = new StreamReader(encFile);
          String str = "";
          str = (sr).ReadToEnd();
          tbContent.Text = str;
          fs.Close();
          MessageBox.Show("encrypted");
          btnDecrypt.Enabled = true;
      }
```

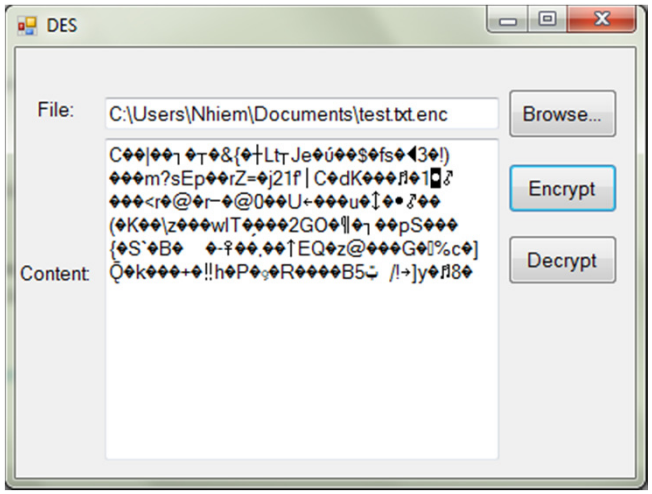
Minh họa DES

```
private void btnDecrypt_Click(object sender, EventArgs
e)
{
        FileStream fsread = new FileStream(tbFile.Text,
FileMode.Open, FileAccess.Read);
        ICryptoTransform desdecrypt =
des.CreateDecryptor();
        CryptoStream cryptostreamDecr = new
CryptoStream(fsread, desdecrypt,
CryptoStreamMode.Read);
        string decryptedFile = new
StreamReader(cryptostreamDecr).ReadToEnd();
        FileInfo fi = new FileInfo(tbFile.Text);
```


Minh họa DES

```
        string origionalFile =
tbFile.Text.Substring(0, tbFile.Text.Length -
fi.Extension.Length);
        StreamWriter fileWriter = new
StreamWriter(origionalFile);
        fileWriter.Write(decryptedFile);
        fileWriter.Close();
        tbContent.Text = decryptedFile;
        MessageBox.Show("decrypted");
}
```

Minh họa DES



Minh họa ứng dụng mã hóa

- Không nên lưu trữ password thô vào cơ sở dữ liệu, nếu hacker tấn công thành công thì nguy cơ bị chiếm hoàn toàn hệ thống rõ ràng xảy ra

Results | Messages

	USERNAME	PASSWORD	HOTEN	DIENTHOAI	CHUCVU	MAKHOA
1	admin	21232F297A57A5A743894AE4A801FC3	Trần	NULL	NULL	0
2	admin1	E0CF25AD42683B3DF678C61F42C6BDA	NULL	NULL	NULL	0
3	CNPM	747FC2989F7EADC7ED4F490BC9FC65A	NULL	NULL	NULL	CNPM
4	HTTT	8823247E8CA93837D44ABD6548B74ECA	NULL	NULL	NULL	HTTT
5	KHMT	39CB116725EA22A3AB6824DF7954E29	NULL	NULL	NULL	KHMT
6	KTMT	76789B92E773FB61FD67715B048769D	NULL	NULL	NULL	KTMT
7	MMT	3272CA43DFEC08F03EB95CE1D2F4D9	NULL	NULL	NULL	MMT

Query executed successfully. VP\SQL2008 (10.0 RTM)

Bảo vệ tác quyền phần mềm

- Dạng thường thấy bảo vệ tác quyền phần mềm là một CD-ROM với mã bản quyền
- Chỉ có 1 cách bảo đảm rằng mỗi mã tác quyền (license code) chỉ được sử dụng trên 1 máy là theo dõi code đó từ server trung tâm
- Phương pháp tổng quát sinh ra license code là chọn một số ngẫu nhiên lớn a

Bảo vệ tác quyền phần mềm

- Tăng số a đó bằng cách nhân với một số ngẫu nhiên b
- Số này được mã hóa vì thế không dễ nhớ
- Key người dùng nhập vào c là hợp lệ nếu (c - a) mod b = 0
- Ta có thể quảng bá key này trên mạng cục bộ hoặc server trung tâm để bảo đảm key duy nhất

Bảo vệ tác quyền phần mềm

- Kẻ tấn công rất khó khăn tìm được key hợp lệ thứ hai từ c nếu a và b đủ lớn
- Phương pháp khác là giả sử phần mềm sinh ra một số ngẫu nhiên lớn n tại thời điểm mua. Số này có thể được mã hóa bởi khóa riêng để sinh ra số thứ hai là m và trả về cho phần mềm
- Nếu m giải mã với khóa chung cho ra n thì key là hợp lệ

Bảo vệ tác quyền phần mềm

- Hacker có thể dùng phần mềm tự động lặp hàng triệu lần các tổ hợp key để giả lập người dùng nhập thông tin vào cửa sổ “enter license key”
- Vì vậy phần mềm của chúng ta nên kết thúc nếu sau 3 lần thử bị sai và xóa chính nó nếu sai 100 lần.

Bài tập

- Cài đặt các chương trình đã minh họa trong bài giảng của chương bằng ngôn ngữ C# hoặc VB.NET