Nghiên cứu phối hợp hai phương pháp nén và mã hóa thông tin

Nguyễn Quý Hào

Trường Đại học Công nghệ
Luận văn Thạc sĩ ngành: Truyền số liệu và Mạng máy tính; Mã số: 60.48.15
Người hướng dẫn: PGS.TS. Trịnh Nhật Tiến
Năm bảo vệ: 2012

Abstract: Trình bày cơ sở toán học được sử dụng trong quá trình nén và mã hoá thông tin gồm: các khái niệm, các định lý, định nghĩa và một số thuật toán cơ bản. Tìm hiểu các thuật toán mã hoá: AES, RSA và các kỹ thuật có liên quan được sử dụng trong quá trình mã hoá. Nghiên cứu các phương pháp nén: Fanno, Shanon, Huffman, Lzw … Phân tích hướng nghiên cứu phối hợp các phương pháp nén và mã hoá thông tin. Giải pháp thực hiện và đánh giá mô hình nghiên cứu. Ngoài ra còn trình bày về quá trình cài đặt chương trình thử nghiệm mô hình phối hợp bằng ngôn ngữ lập trình C#.Net.

Keywords: Nén thông tin; Mã hóa thông tin; Kỹ thuật giấu tin

Content

LỜI MỞ ĐẦU

Quá trình lưu trữ và truyền tải thông tin luôn luôn có 2 yếu tố được quan tâm hàng đầu là: tính an toàn bảo mật và kích thước của tệp tin.

Đã có rất nhiều các phần mềm, các chương trình được viết để giải quyết hai vấn đề được đặt ra. Tuy nhiên nếu chỉ nén dữ liệu kích thước tệp tin được giảm nhưng lại không bảo đảm tính an toàn thông tin. Ngược lại nếu chỉ mã hoá chỉ đảm bảo tính an toàn nhưng không giải quyết được vấn đề giảm dung lượng lưu trữ hơn thế mã hoá tệp tin lớn tốn nhiều thời gian và băng thông để truyền tải cũng tăng theo.

Trong khi đó nếu phối hợp cả hai quá trình trên sẽ đem lại rất nhiều lợi ích: giảm dung lượng lưu trữ, giảm băng thông truyền tải, giảm thời gian mã hoá, tăng tính bảo mật cho tệp tin so với tệp tin chỉ mã hoá đơn thuần.

Từ ý nghĩa thực tiễn quan trọng nêu trên là động lực để tôi nghiên cứu đề tài: "Nghiên cứu phối hợp hai phương pháp nén và mã hoá thông tin".

Trong luận văn sẽ đề xuất mô hình và giải pháp phối hợp hai phương pháp nén và mã hoá thông tin: sử dụng các thuật toán nén để nén dữ liệu sau đó dùng phương pháp mã hoá đối xứng để mã hoá tệp tin sau, cuối cùng là dùng mã khoá khoá bất đối xứng RSA để mã hoá khoá chung của AES.

Luận văn được trình bày theo cấu trúc sau:

- Chương 1: trình bày cơ sở toán học được sử dụng trong quá trình nén và mã hoá thông tin gồm: các khái niệm, các định lý, định nghĩa và một số thuật toán cơ bản
- Chương 2: trình bày về các thuật toán mã hoá: AES, RSA và các kỹ thuật có liên quan được sử dụng trong quá trình mã hoá.
- Chương 3: trình bày về các phương pháp nén: Fanno, Shanon, Huffman, Lzw…

- Chương 4: trình bày về hướng nghiên cứu phối hợp các phương pháp nén và mã hoá thông tin. Giải pháp thực hiện và đánh giá mô hình nghiên cứu. Ngoài ra còn trình bày về quá trình cài đặt chương trình thử nghiệm mô hình phối hợp bằng ngôn ngữ lập trình C#.Net.

Chương 1: MỘT SỐ KHÁI NIỆM CƠ BẢN

1.1 CÁC ĐỊNH LÝ QUAN TRỌNG

1.1.1 Định lý Euler:

Cho $a \in Z, m \in N, m > 1$. Nếu UCLN (a, m) = 1 thì $a^{\varphi(m)} \equiv 1 \pmod m$

1.1.2 Định lý Fermat (hệ quả của định lý Euler)

Cho $a \in Z$ và k là một số nguyên tố khi đó $a^k \equiv a \pmod k$

Nếu UCLN(a, k) = 1 thì $a^{k-1} \equiv 1 \pmod k$

1.1.3 Định lý đồng dư Trung Quốc

Cho $m_1, m_2, \ldots, m_r$ là các số nguyên tố cùng nhau từng đôi một nghĩa là UCLN(m_1, m_2) = 1 $\forall$ i, j = 1, 2, …, r ; $i \neq j$. Giả sử $a_1, a_2, \ldots a_r \in Z$ khi đó hệ phương trình đồng dư

$$\begin{cases} x \equiv a_1 (\mathrm{mod} m_1) \\ x \equiv a_2 (\mathrm{mod} m_2) \\ .... \\ x \equiv a_r (\mathrm{mod} m_r) \end{cases}$$

Có nghiệm duy nhất theo modulo $M = m_1 m_2, \ldots m_r$ là $x = \sum_{i=1}^{r} a_i M_i y_i$ trong đó $M_i = \frac{M}{m_i}$ và $y_i = M_i^{-1} \bmod m_i$

1.1.4 Định lý Bezout

Cho a, b ∈ N, a > b ≥ 1; ta có:

+ Tồn tại x, y ∈ Z sao cho ax + by = UCLN(a, b)
+ Nếu a, b nguyên tố cùng nhau thì tồn tại x,y ∈ Z | ax + by = 1
+ a, b nguyên tố cùng nhau khi và chỉ khi tồn tại x, y ∈ Z | ax + by = 1

1.2 MỘT SỐ THUẬT TOÁN

1.2.1 Thuật toán Euclidean (Tìm ước chung lớn nhất của hai số)

INPUT: $r_0, r_1 \in N, r_0 > r_1 \geq 0$

OUPUT: $d = UCLN(r_0, r_1)$

Thuật toán:

Bước 1: Nếu $r_1 = 0$ trả về $d := r_0$, kết thúc thuật toán

Nếu $r_1 \neq$ chuyển sang bước 2

Bước 2: $r := r_0 \bmod r_1$; $r_0 := r_1$; $r_1 = r$; quay lại bước 1

1.2.2 Thuật toán Euclidean mở rộng

Với mọi j, $0 \leq j \leq 2$ ta có $r_j \equiv t_j r_1 \pmod{r_0}$

Hệ quả: Nếu UCLN (r_0, r_1) = 1 thì $t_m = r_1^{-1} \bmod r_0$

1.2.3 Thuật toán bình phương và nhân

INPUT: $x \in Z_n$, $b \in Z$, $0 < b < n$,

b: viết dưới dạng nhị phân với k chữ số nhị phân $b = b_1b_2\ldots b_k$; $b_i = \{0, 1\}$

OUTPUT: Số $a = x^b \bmod n$

1.2.4 Thuật toán xác suất kiểm tra số nguyên tố

Thuật toán Miller – Rabin

Kiểm tra Miller:

Giả sử n là một số nguyên dương lẻ, khi đó ta biểu diễn được $n - 1 = 2^st$ với s là một số nguyên không âm, t là một số nguyên dương lẻ. Ta nói n vượt qua được kiểm tra Miller cơ sở a ($a \in Z, a > 0$) nếu $a^t \equiv 1 \pmod n$ hoặc $a^{2^kt} \equiv -1 \pmod n$ với k nào đó $0 \le k < s$.

Mệnh đề:

Nếu n là một số nguyên tố thì n vượt qua kiểm tra Miller cơ sở $\forall a, 0 < a < n$

Định nghĩa:

Nếu n vượt qua Miller cơ sở a thì n được gọi là số nguyên tố giả cơ sở a

Số nguyên dương $n > 1$ được gọi là số giả nguyên tố mạnh cơ sở a nếu nó là hợp số và vượt qua được kiểm tra Miller cơ sở a

Thuật toán Miller – Rabin

INPUT: $n \in N$, lẻ, $N > 1$

OUTPUT: "n là nguyên tố" hoặc "n là hợp số"

Mệnh đề: Thuật toán Miller – Rabin cho bài toán hợp số là thuật toán Monter – Carlo định hướng có. Nghĩa là câu trả lời "b là hợp số" luôn đúng. Câu trả lời "n là số nguyên tố" có xác suất sai không quá ¼

Miller – Rabin – Test(n)

Thuật toán Miller – Rabin với t lần thực hiện kiểm tra Miller:

INPUT: $n \in N$, lẻ, $n > 1$, $t \in N$, số lần kiểm tra

OUTPUT: "n là nguyên tố" hoặc "n là hợp số"

Xác suất sai khi trả lời n là số nguyên tố không vượt quá $(1/4)^t$

1.3 KHÁI NIỆM ENTROPY

1.3.1 Định nghĩa Entropy

Sự đoán nhận trạng thái xuất hiện của đồng xu là một việc bất định. Bất định không phải ở chỗ không thể đoán nhận được là mặt nào sẽ xẩy ra mà là ở chỗ không thể đưa ra một khẳng định chắc chắn.

1.3.2 Tính chất của Entropy

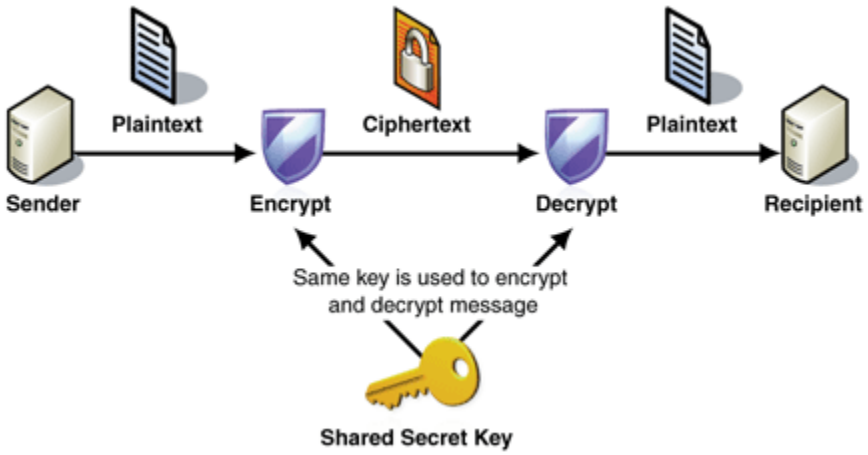
Tính chất 1: Tính đơn điệu tăng

Tính chất 2: Tính chất cộng

Chương 2: PHƯƠNG PHÁP MÃ HOÁ

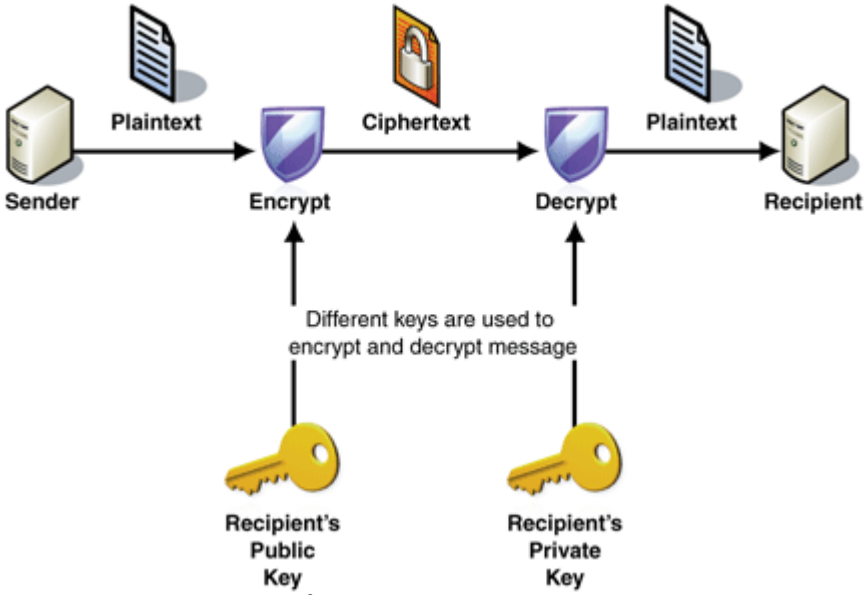
2.1. CÁC KHÁI NIỆM CƠ BẢN

2.1.1. Hệ mã hoá khoá đối xứng



Hình 2.2: Mô hình truyền thông sử dụng hệ mã hoá khoá đối xứng

2.1.2. Hệ mã hoá khoá phi đối xứng



Hình 2.2: Mô hình truyền thông sử dụng hệ mã hoá khoá công khai

2.1.3. Hệ mã hoá RSA

2.1.3.1 Lịch sử hình thành hệ mã hoá RSA

2.1.3.2 Hệ mã hoá đầu tiên

Hệ mã hoá RSA(Ron Rivers, Adi Shamir Len Adleman)

Thuật toán RSA xây dựng dựa trên độ khó của bài toán phân tích số nguyên lớn ra thừa số nguyên tố

2.1.3.3 Định nghĩa hệ mã hoá RSA

Cho bộ số RSA (n, p, q, a, b), $x \in Z_n$.

Đặt $y = x^b \bmod n \Rightarrow x = y^a \bmod n$

Trong đó: *(n, b) là thành phần khoá công khai*

(p, q, a) là khoá bí mật, p, q là số nguyên tố

2.2. CÁC CHUẨN KỸ THUẬT TRONG PKCS

2.2.1. Tổng quan về PKCS và PKCS#1 v2.1

2.2.1.1. PKCS

PKCS (Publick – Key Cryptography Standards) là một tập hợp các chuẩn trong mã hoá hóa công khai, được đánh thứ tự PKCS#1 tới PKCS #15

2.2.1.2. PKCS#1 v2.1

Trong PKCS# v2.1 sử dụng:

Thuật toán mã hoá RSA đa nguyên tố (Multi Prime RSA)
Cơ sở chuyển đổi I2OSP và OS2
Sử dụng hàm băm SHA1: đầu vào thông điệp $2^{64} - 1$ => Thông điệp 160 bit, Hàm sinh mặt lạ MGF1

2.2.2. Các ký hiệu trong PKCS#1 v2.1

2.2.3. Các kiểu khóa

2.2.3.1. Khóa công khai RSA

Một khóa công khai RSA bao gồm hai thành phần:
+ n modulo RSA, một số nguyên dương
+ e số mũ công khai RSA, một số nguyên dương
Trong một khóa công khai RSA hợp lệ thì:
+ Modulo RSA n là tích của u số nguyên tố lẻ phân biệt r_i, i = 1, 2, 3,...,u; u ≥ 2
+ số mũ công khai RSA e là số nguyên thỏa $3 \le e \le n - 1$, UCLN(e, λ(n)) = 1

2.2.3.2. Khóa bí mật RSA

Có thể sử dụng một trong hai cách biểu diễn khóa bí mật RSA
Dạng 1: theo cách biểu diễn này thì một khóa bí mật RSA bao gồm hai thành phần:
+ n modulo RSA, một số nguyên dương
+ d số mũ bí mật RSA, một số nguyên dương
Dạng 2: trong cách biểu diễn này thì một khóa bí mật RSA bao gồm một bộ năm số (p, q, dP, dQ, qInv) và các bộ ba (khuyết nếu u = 2) (r_i, d_i, t_i).
Trong một khóa bí mật RSA hợp lệ ở dạng biểu diễn thứ nhất thì:
+ n mang ý nghĩa như trong khóa công khai RSA
+ d số mũ bí mật RSA, là một số nguyên dương nhỏ hơn n và thỏa mãn $e.d \equiv 1 \pmod{\lambda(n)}$.
Một khóa bí mật RSA hợp lệ ở dạng biểu diễn thứ hai thì phải thỏa mãn:
+ p, q là hai thừa số nguyên tố đầu tiên của modulo RSA n. Các số mũ CRT dP, dQ là những số nguyên dương tương ứng nhỏ hơn p và q thỏa mãn:
$e.dP \equiv 1 \pmod{(p - 1)}$
$e.dQ \equiv 1 \pmod{(q - 1)}$
+ Hệ số CRT đầu tiên qInv là một số nguyên dương nhỏ hơn p, thỏa mãn $q.qInv \equiv 1 \pmod{p}$
+ Nếu u > 2 thì trong khóa bí mật RSA dạng hai bao gồm các bộ ba (r_i, d_i, t_i), I = 3, 4, … u.trong đó r_i là thừa số nguyên tố thứ I của modulo RSA n. Với số mũ CRT d_I (i = 3, 4,…,u) phải thỏa mãn $e.d_i \equiv 1 \pmod{(r_i - 1)}$, $d_i < r_i$. Mỗi hệ số CRT t_i (i = 3, 4, …, u) là một số nguyên dương nhỏ hơn r_i và thỏa mãn $R_i . t_i \equiv 1 \pmod{r_i}$, $R_i = r_1.r_2...r_{i-1}$

2.2.4. Cơ sở chuyển đổi dữ liệu I2OSP và OS2IP

2.2.4.1. Chuyển đổi dữ liệu I2OSP

Integer – to – Octet – String Primitive: chuyển đổi từ dạng số nguyên sang chuỗi octet (dãy 8 bit)

2.2.4.2. Chuyển đổi dữ liệu OS2IP

Octet – String – to – Integer Primitive: chuyển đổi từ dạng chuỗi octet (dãy 8 bit) sang dạng số nguyên

2.2.5. Cơ sở của hệ mật mã

2.2.5.1.Cơ sở hệ mã hóa RSAEP

RSAEP((n, e), m)
INPUT:
(n, e) khóa công khai RSA
m biểu diễn thông điệp, một số ngyên giữa 0 và n – 1
OUTPUT:
c biểu diễn bản mã, một số nguyên giữa 0 và n – 1

Lỗi: "biểu diễn thông điệp ngoài giớ hạn"
Giả định: khóa công khai RSA(n, e) hợp lệ

2.2.5.2.Cơ sở hệ giải mã hóa – RSADP

RSADP(K, c)
INPUT: K: khóa bí mật RSA, trong đó K thuộc một trong hai hình thức sau:
+ Một cặp (n, d)
+ Một bộ năm (p, q, dP, dQ, qInv) và một dãy các bộ ba (có thể khuyết)
(r_i, d_i, t_i), i = 3, …, u
c: biểu diễn bản mã, một số nguyên nằm giữa 0 và n – 1
OUTPUT: m: biểu diễn thông điệp, một số nguyên nằm giữa 0 và n – 1
Error: "Biểu diễn thông điệp nằm ngoài giới hạn"

2.2.6. Lược đồ mã hoá

2.2.6.1. Tổng quan về lược đồ mã hoá

2.2.6.2. Các kỹ thuật hỗ trợ

1. Hàm băm SHA1: SHA1 là một hàm băm mã hoá được thiết kế bởi National Security Agency (NSA) và được công bố bởi NIST theo chuẩn FIPS PUBS 180 – 1 ngày 17/4/1995. Hàm băm SHA1 nhận đầu vào là một thông điệp có chiều dài tối đa là $2^{64} – 1$, và sản xuất ra một thông điệp rút rọn với chieuf dài 160bit
2. Hàm sinh mặt nạ MGF1
*Input:*mgf Seed hạt giống để sinh mặt nạ, một chuỗi octet
Output: Mask mặt nạ, một chuỗi octet với chiều dài maskLen
Error:"mask too long"

2.2.6.3. Lược đồ RSAES – OAEP

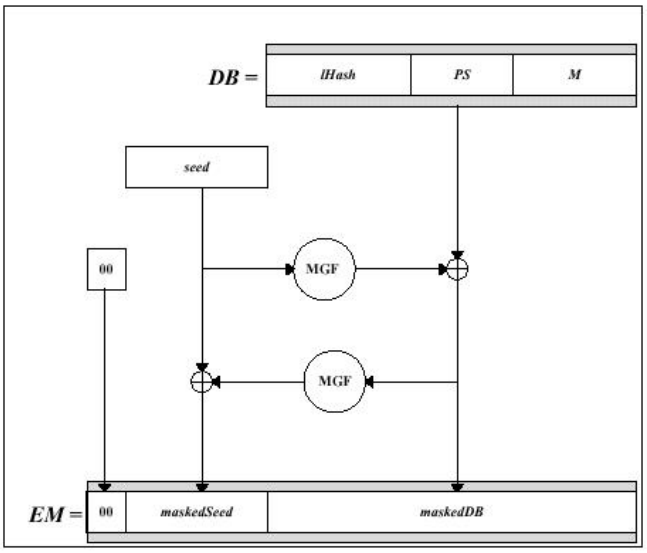
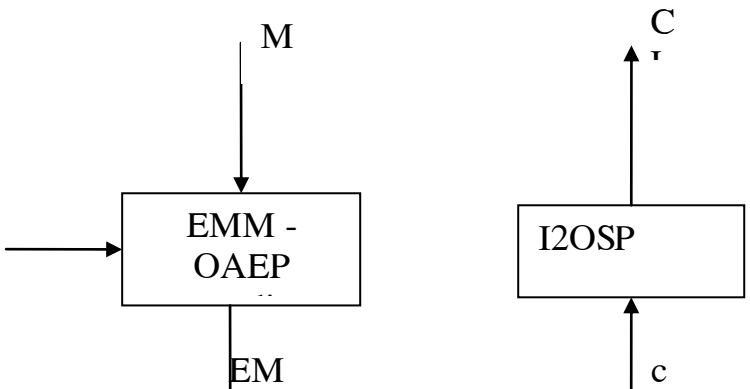
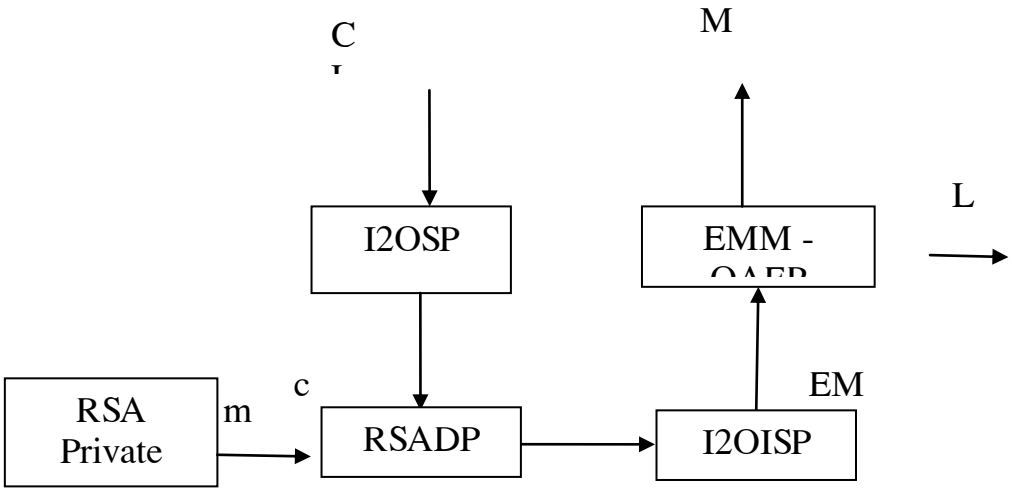


Figure 1: EME-OAEP encoding operation. *lHash* is the hash of the optional label *L*. Decoding operation follows reverse steps to recover *M* and verify *lHash* and *PS*.

Hình 2.4: Sơ đồ mã hoá EME – OAEP [13]



Hình 2.5: Tóm lược quy trình xử lý RSAES – OAEP - ENCRYPT



Hình 2.6: Tóm lược quy trình xử lý RSAES – OAEP - DECRYPT

2.2.7. Ý nghĩa của việc áp dụng EME - OAEP trước khi mã hóa RSA

Nếu không thực hiện việc độn thông điệp M, quá trình chuyển đổi M sang dạng số nguyên m sẽ có thể rơi vào một số trường hợp sau:

1. m = 0 hoặc m = 1 thì RSADP sẽ tạo ra các bản mã có giá trị tương ứng (với khóa bất kỳ)
2. Khi mã hóa với số mũ nhỏ (chẳng bạn e = 3) và m cũng có giá trị nhỏ ($m < n^{1/e}$) khi đó phép tính modulo là không có ý nghĩa, như vậy đối phương sẽ dễ tìm ra bản rõ m bằng việc khai căn bậc e của c

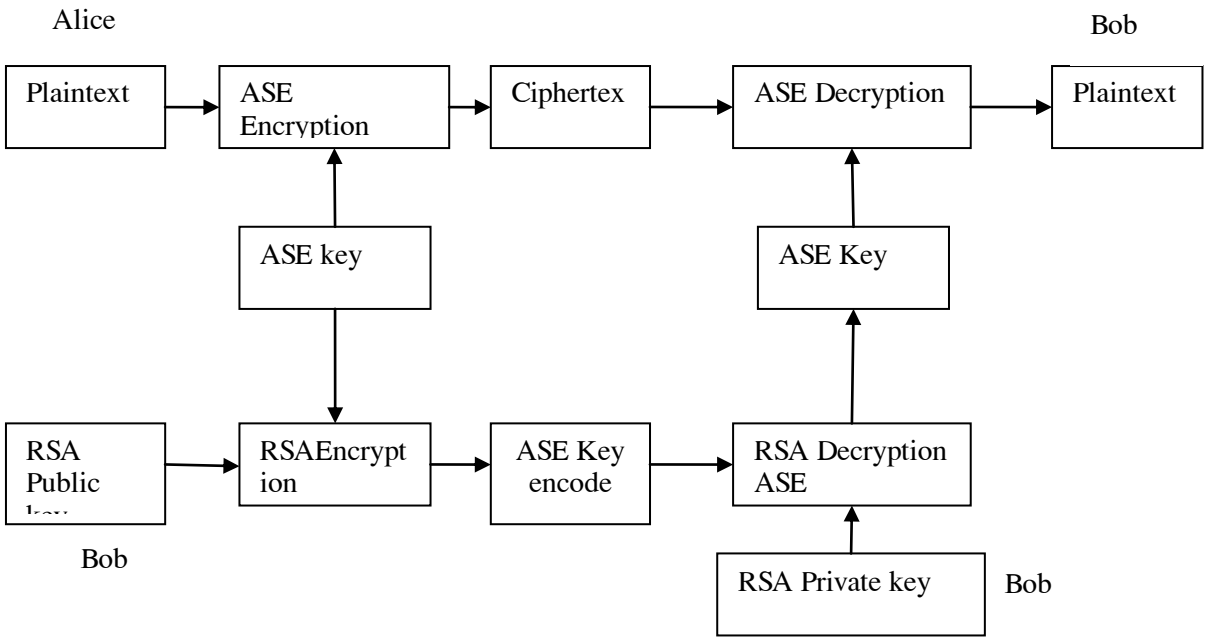
Mặt khác, RSA là thuật toán mã hóa tất định (xác định) nghĩa là với một bản rõ ban đầu, cùng một khóa công khai được sử dụng để mã hóa thì một bản mã duy nhất được tạo ra. Do đó đối phương có thể tấn công hệ mật bằng phương pháp tấn công lựa chọn bản rõ. Việc áp dụng EME - OAEP hạn chế tối đa được sự tấn công này

2.2.8 Vấn đề sinh khóa RSA

Vấn đề sinh khóa RSA trên các số nguyên tố lớn theo thuật toán kiểm tra số nguyên tố Rabin - Miller

2.3. CHUẨN MÃ HÓA DỮ LIỆU TIÊN TIẾN – AES

2.3.1. Mục đích nghiên cứu chuẩn AES



Hình 2.7: Sơ đồ kết hợp RSA và AES

Việc phải tính toán trên các số nguyên lớn làm cho tốc độ của thuật toán RSA chậm đáng kể so với các thuật toán mã hóa khóa đối xứng (như DES, AES) vì lý do này người ta thường không sử dụng thuật toán RSA để xử lý (mã hóa và giải mã) trực tiếp các thông điệp dài. Thuật toán RSA thường được sử dụng kết hợp với một thuật toán mã hóa đối xứng

2.3.2. Tổng quan

AES (viết tắt của Advanced Encryption Standard) là một thuật toán mã hoá được sử dụng để bảo vệ dữ liệu điện tử. Có cách gọi khác nhau về thuật toán AES: AES – 128, AES – 129, AES – 256

2.3.3. Các khái niệm cơ sở

2.3.3.1. Input, Output, Key

Cho thuật toán AES đều là các chuỗi 128 bit (tương ứng 16 byte). Một chuỗi như thế thường được gọi là một khối (block),

2.3.3.2.Byte

Đơn vị cơ sở được xử lý trong thuật toán AES

2.3.3.3.Ma trận trạng thái (State Matrix)

Thuật toán AES bao gồm một tập các toán tử thực thi và làm thay đổi một ma trận hai chiều được gọi là ma trận trạng thái. Ma trận trạng thái là ma trận cỡ 4x4 chứa 16 byte

2.3.3.4. Hộp thay thế S – Box và InvS – Box

Bảng các giá trị xác định sử dụng trong quá trình mã hoá và giải mã AES

2.3.4. Đặc tả thuật toán

2.3.4.1.Sinh khóa con: Ban đầu khóa K = $k_0k_1…k_{15}$ (16 byte) được đưa vào. Từ khóa K sinh ra 44 khóa con W[0], W[1], … W[43], với mỗi khóa còn là một word 4 byte

2.3.4.2. Hoạt động mã hóa

(Thuật toán mã hoá và các thủ tục sử dụng trong mã hoá AES)

Sau khi dữ liệu từ Input được nạp vào ma trận trạng thái, quá trình mã hóa diễn ra dưới sự tham gia của 4 hàm (toán tử) thao tác và biến đổi ma trận trạng thái:

+ SubBytes(): thay thế tất cả các phần từ (byte) của ma trận trạng thái bằng việc áp dụng S – Box lên phần tử đó

+ ShiftRows(): hoán vị vòng dòng thứ i của ma trận trạng thái về bên trái i bước, mỗi bước dịch 1 byte

+ MixColums(): thực hiện phép nhân ma trận cố định với ma trận trạng thái, trong đó phép nhân ma trận dựa trên các phép toán cộng và nhân byte đã định nghĩa

$$\text{State} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \text{ x State}$$

+ AddRoudKey(): Tác động vào ma trận trạng thái 11 vòng (r = 0, 1, …, 10), tại vòng lặp r, 4 khóa con liên tiếp được sử dụng: W[4r], W[4r + 1], W[4r + 2], W[4r + 3]. Trong đó mỗi khóa con được xem là một cột bốn byte, được tạo thành ma trận 4x4, đem ma trận này XOR với ma trận trạng thái làm biến đổi ma trận trạng thái…

2.3.4.3.Hoạt động giải mã

(thuật toán giải mã và các thủ tục trong quá trình giải mã AES)

Hoạt động giải mã là quá trình ngược lại với hoạt động mã hóa. Tương ứng với 4 hàm sử dụng trong quá trình mã hóa ta có 4 hàm sử dụng trong quá trình giải mã:

+ InvSubBytes(): Thay thế tất cả phần từ của ma trận trạng thái bằng việc áp dụng InvS – Box lên từng phần đó

+ InvSiftRows(): Hoán vị dòng thứ i của ma trận trạng thái i bước về bên phải, mỗi bước dịch một byte

+ InvMixColumns():Thực hiện phép nhân ma trận cố định (nghịch đảo của ma trận cố định sử dụng trong MixColumns()) với ma trận trạng thái, trong đó phép nhân ma trận cũng dựa trên các phép toán cộng và nhân byte đã định nghĩa ở trên

$$\text{State} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \text{ x State}$$

+ Toán tử ngược AddRowKey() là bản thân nó (áp dụng hai lần liên tiếp toán tử này thì ma trận trạng thái không thay đổi

Chương 3: PHƯƠNG PHÁP NÉN DỮ LIỆU

3.1. TỔNG QUAN VỀ NÉN DỮ LIỆU

3.1.1. Mã nén dữ liệu

3.1.1.1. Nén dữ liệu, bit trung bình

- Nén dữ liệu: Quá trình giảm dung lượng cần thiết để lưu trữ một lượng thông tin cho trước
- Bít trung bình: Tỉ số giữa độ dài văn bản cho số chữ cái trong văn bản

3.1.1.2. Mã tổng và mã phân tách

3.1.2. Định lý Shannon

Một văn bản thực ra thì chỉ có thể nén đến một giới hạn nhất định, giới hạn ấy chúng ta gọi là lượng tin của văn bản. Mọi thuật toán đều không thể nén một văn bản đến một file nhỏ hơn lượng tin mà văn bản có

3.2. MÔ HÌNH THỐNG KÊ

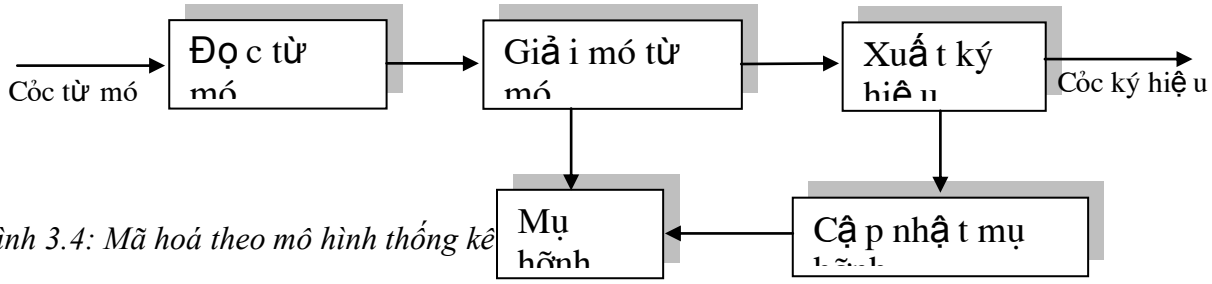
Nén văn bản dựa trên thống kê tỉ lệ xuất hiện các chữ cái trong văn bản

3.2.1. Mô hình thống kê tĩnh

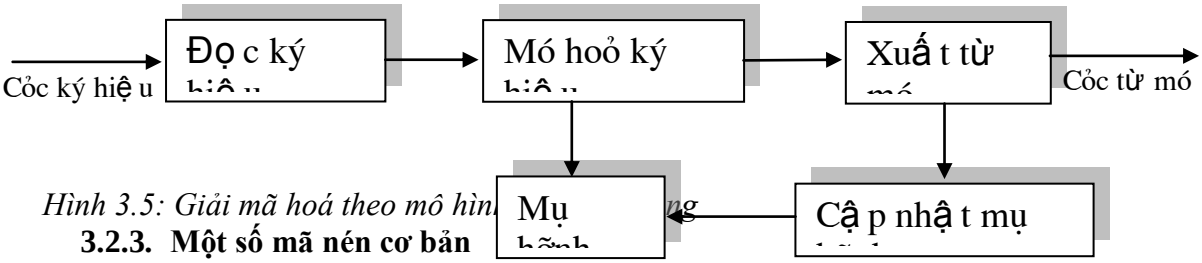
Dạng đơn giản nhất của mô hình thống kê tĩnh là một bảng tĩnh liệt kê các giá trị xác suất theo cách tính thông thường

3.2.2. Mô hình thống kê động

Số liệu thống kê đối với dữ liệu cần mã hoá không phải lưu trữ trước mà liên tục được tích luỹ và sửa đổi trong suốt quá trình mã hoá và giải mã



Hình 3.4: Mã hoá theo mô hình thống kê



Hình 3.5: Giải mã hoá theo mô hình thống kê

3.2.3. Một số mã nén cơ bản

3.2.3.1. Mã Fano - Mã Shannon

a. Mã Fano

Giả sử a_i với i = 1..n là các chữ cái của một bảng chữ cái nào đó và a_i xuất hiện với xác suất tương ứng là p_i. Chú ý $p_1 + p_2 + ... + p_n = 1$.

Bước 1: Bằng cách sắp xếp và ký hiệu lại chúng ta có thể coi các chữ cái $a_1, a_2, ..., a_n$ có xác suất là $p_1 \geq p_2 \geq ... \geq p_n$ (theo thứ tự giảm dần).

Bước 2: Chia các chữ cái ra làm 2 nửa, nửa trên và nửa dưới, sao cho chúng có tổng gần bằng nhau nhất. Nửa trên nhận mã là 0, nửa dưới nhận là 1.

Bước 3: Lặp lại công việc cho từng nửa và cứ tiếp tục với các nửa mới sinh ra cho tới khi trong mỗi nửa chỉ có một chữ cái. Dãy các số 0, 1 được tạo ra là mã của các chữ cái.

b. Mã Shannon

Giả sử a_i với i = 1..n là các chữ cái của một bảng chữ cái nào đó và a_i xuất hiện với xác suất tương ứng là p_i. Chú ý $p_1 + p_2 + ... + p_n = 1$.

Bước 1: Bằng cách sắp xếp và ký hiệu lại chúng ta có thể coi các chữ cái $a_1, a_2, ..., a_n$ có xác suất là $p_1 \geq p_2 \geq ... \geq p_n$ (theo thứ tự giảm dần).

Bước 2: Chia các chữ cái ra làm 2 nửa, nửa trên và nửa dưới, sao cho chúng có tổng gần bằng nhau nhất. Nửa trên nhận mã là 0, nửa dưới nhận là 1.

Bước 3: Lặp lại công việc cho từng nửa và cứ tiếp tục với các nửa mới sinh ra cho tới khi trong mỗi nửa chỉ có một chữ cái. Dãy các số 0, 1 được tạo ra là mã của các chữ cái.

3.2.3.2. Mã Huffman

1. Bắt đầu với một rừng các cây. Tất cả các cây có một nút, cùng với trọng lượng của cây bằng trọng lượng của mỗi kí tự trong nút. Các kí tự xuất hiện thường xuyên nhất có trọng lượng lớn nhất. Các kí tự ít xuất hiện nhất có trọng lượng nhỏ nhất.
2. Lặp bước này cho đến khi chỉ còn một cây:
3. Chọn hai cây có trọng lượng nhỏ nhất, gọi là cây T_1 và T_2. Tạo ra một cây mới mà gốc của cây này có trọng lượng bằng tổng trọng lượng của cây $T_1 + T_2$ và cây con trái là T_1 và cây con phải là T_2.
4. Một cây đơn thu được sau bước 2 là cây mã hoá tối ưu

3.2.3.3. Lưu đồ giải mã Fanon, Shannon, Huffman

3.3. MÔ HÌNH TỪ ĐIỂN

3.3.1. Giới thiệu

Mô hình từ điển: đọc dữ liệu vào rồi tìm một nhóm ký hiệu tương hợp hiện có trong từ điển. Nếu tìm thấy chúng xuất ra một con trỏ đến nhóm ký hiệu đó

3.3.2. Kỹ thuật từ điển

3.3.2.1. Nguyên lý LZ

- Mã với từ điển tĩnh: từ điển cố đinh, không khả thi
- Mã với từ điển động: từ điển liên tục được cập nhật trong quá trình nén và giải nén

3.3.2.2. Các thuật toán LZ

a. Thuật toán LZ77

Nguyên tắc nén như sau:

Bước 1: F – 1 kí tự đầu tiên giữ nguyên.

Bước 2: Cho buffer gồm F kí tự trong tương lai dịch dần về quá khứ N lần. Mỗi lần đếm số kí tự trùng nhau liên tiếp kể từ đầu của buffer với quá khứ và ghi nhớ con số ấy tương ứng với các vị trí quá khứ mà buffer dịch đến. Tìm số lớn nhất trong tất cả các con số ấy.

Bước 3: Ghi ra mã gồm [i, j, w] trong đó i là khoảng cách từ vị trí hiện tại đến vị trí tương ứng với số lớn nhất vừa tìm được, j là số các kí tự trùng nhau hay chính là số lớn nhất vừa tìm được, w là kí tự ở vị trí thứ j + 1 trong buffer.

Bước 4: Lặp lại bước 2 cho đến khi hết văn bản.

Quá trình giải nén

Bước 1: Đọc F – 1 kí tự đầu tiên của bản mã ghi ra bản giải mã.

Bước 2: Đọc một bộ mã [i, j, w]. Lùi về đoạn văn bản đã giải mã i vị trí và copy j kí tự từ vị trí thứ i đó để ghi ra bản giải mã, sau đó ghi ra w.

Bước 3: Nếu còn bộ mã thì quay lại bước 2. Nếu không còn thì kết thúc.

b. Thuật toán LZ78

Thuật toán nén.

Bước 1: Khởi tạo từ điển có một phần tử là xâu rỗng có chỉ số 0. Xâu trung gian P rỗng.

Bước 2: Đọc kí tự tiếp theo trong văn bản vào C.

Bước 3: Tìm khúc (P + C) đã có trong từ điển chưa?

Nếu có thì P ← P + C và quay lại bước 2.

Nếu không thì:

Cập nhật P + C vào trong từ điển.

W ← chỉ số của P trong từ điển.

Ghi cặp (W, C) ra bản mã.

P ← rỗng.

Bước 4: Còn kí tự trong văn bản vào?
Nếu còn thì quay lại bước 2.
Nếu không còn thì ghi ra cặp (W, C) với C là rỗng

Thuật toán giải nén.

Bước 1: Khởi tạo từ điển có một phần tử là xâu rỗng có chỉ số 0.
Bước 2: Đọc một cặp (W, C).
Bước 3: Ghi xâu (.W + C) ra bản giải mã. Cập nhật xâu này vào từ điển.
Bước 4: Còn kí tự trong bản mã không?
Nếu còn thì đọc cặp (W, C) tiếp theo và quay lại bước 2.
Nếu không thì kết thúc.

c. Thuật toán LZW

Thuật toán nén

Bước 1: Khởi tạo từ điển gồm các kí tự trong bảng chữ cái của văn bản xuất hiện trong văn bản được đánh chỉ số từ 0. Xâu trung gian P rỗng.
Bước 2: Đọc kí tự tiếp theo trong văn bản vào C.
Bước 3: Xâu P + C đã có trong từ điển chưa?
Nếu có thì P ← P + C
Nếu không thì:
Ghi ra bản mã chỉ số của P trong từ điển.
Thêm xâu P + C vào từ điển.
P ← C (lúc này P chứa một kí tự).
Bước 4: Còn kí tự nào trong văn bản không?
Nếu có thì quay lại bước 2.
Nếu không thì:
Ghi ra bản mã chỉ số của P trong từ điển.
Kết thúc

Thuật toán giải nén

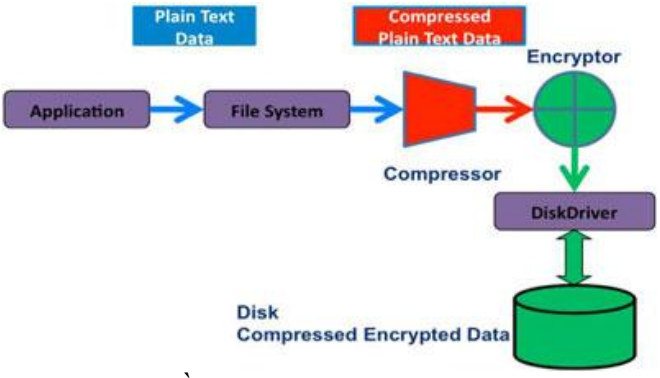
Bước 1: Khởi tạo từ điển ban đầu gồm các kí tự trong bảng chữ cái của văn bản xuất hiện trong văn bản, được đánh chỉ số từ 0.
Bước 2: cW ← từ mã đầu tiên trong bản mã.
Bước 3: Ghi đoạn copy .cW ra bản giải mã.
Bước 4: pW ← cW.
Bước 5: cW ← từ mã tiếp theo trong bản mã.
Bước 6: Đoạn copy .cW có trong từ điển không?
Nếu có thì
Ghi đoạn copy .cW ra bản giải mã.
p ← .pW
c ← kí tự đầu tiên của đoạn copy .cW
Thêm đoạn copy p + c vào từ điển.
Nếu không thì
p ← .pW
c ← kí tự đầu tiên của đoạn copy .pW
Ghi đoạn copy p + c ra bản giải mã và thêm nó vào từ điển.
Bước 7: Còn từ mã nào trong bản mã không?
Nếu có thì quay về bước 4.
Nếu không thì kết thúc

Chương 4: PHỐI HỢP CÁC PHƯƠNG PHÁP NÉN VÀ MÃ HOÁ THÔNG TIN

MÔ HÌNH PHỐI HỢP HAI PHƯƠNG PHÁP NÉN VÀ MÃ HOÁ THÔNG TIN

Về không gian lưu trữ

Các phương pháp nén về cơ bản phụ thuộc vào việc loại bỏ dư thừa trong tập hợp các dữ liệu. Thông thường, điều này được thực hiện bằng cách tìm kiếm cho dữ liệu lặp đi lặp lại mô hình, giữ lại một ví dụ của mỗi mẫu và thay thế các bản sao và được đánh dấu trên bản nén. Ngược lại, phương pháp mã hóa tốt là làm cho dữ liệu đầu vào (đặc biệt là dư thừa dữ liệu) xuất hiện ngẫu nhiên, do đó loại bỏ tất cả các dư thừa trong dữ liệu, các mẫu lặp đi lặp lại. Kết quả là không thể được nén, dữ liệu đã được mã hoá. Vì vậy, khi cả hai mã hóa và nén được triển khai, công đoạn nén phải được thực hiện trước khi mã hóa.



Hình 4.1 : Luồng xử lý nén và mã hoá

Vấn đề an ninh

Nén có thể làm giảm hiệu quả của một số cuộc tấn công tệp nén bằng cách giảm sự dư thừa trong dữ liệu. Một phương pháp giải mã phổ biến là dựa trên phân tích bản mã, dựa trên tìm kiếm dữ liệu lặp đi lặp lại. Tệp tin sau khi đã nén sẽ làm giảm hiệu quả của phương pháp tấn công này

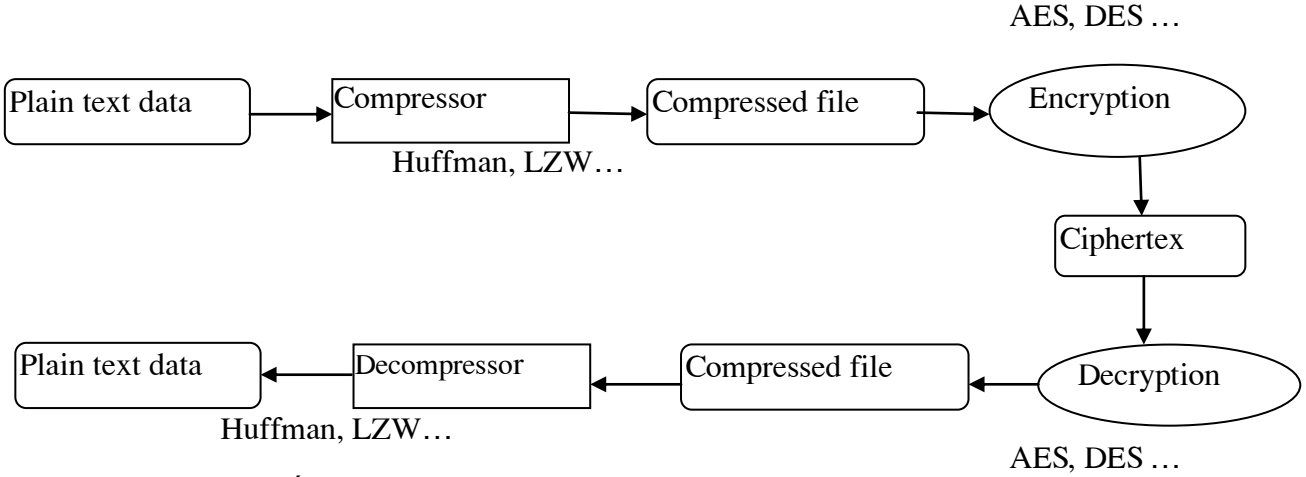
Vấn đề thời gian xử lý dữ liệu

Các phương pháp mã hoá hiện nay thường kích thước tệp tin nén lớn hơn so với tệp tin gốc hơn nữa dữ liệu ngẫu nhiên làm tăng thời gian nén tệp tin

Nén dữ liệu trước kích thước tệp tin sẽ được thu nhỏ, các dữ liệu trùng lặp sẽ được loại bỏ. Thời gian mã hoá tệp tin sẽ giảm đi đáng kể so với việc thực hiện nén tệp tin trước khi mã hoá

Mô hình phối hợp hai phương pháp nén và mã hoá dữ liệu

Như đã trình bày trong mục 4.1.1 quá trình nén và mã hoá sẽ được thực hiện như hình 4.2



Hình 4.2: Mô hình phối hợp hai phương pháp nén và mã hoá thông tin

CHƯƠNG TRÌNH THỬ NGHIỆM

Mô tả chung

Theo *Hình 4.2* để gửi đi file đã nén và mã hoá người dùng cần phải gửi kèm theo file bản mã của khoá AES đã được sử dụng để mã hoá tệp tin nén. Khi người nhận tệp tin nén cần:

- Dùng khoá bí mật RSA của mình để giải mã khoá AES nhận được
- Dùng khoá AES để giải mã tệp tin nén
- Giải nén tệp tin
- Ý tưởng cài đặt chương trình cho qui trình trên như sau:

Khi mã hoá, chương trình chỉ nhận file bản rõ, khoá AES và khoá RSA là tham số đầu vào, khoá AES sau khi được mã hoá bằng thuật toán RSA sẽ được lưu ra một tệp, tệp tin nén sau khi mã hoá sẽ được lưu trên một tệp (***tệp cuối***).

Giải mã, chương trình nhận bản mã của khoá AES, khoá bí mật RSA và tệp nén làm tham số đầu vào, chương trình sẽ đọc bản mã của khoá AES giải mã để lấy khoá, sau đó khoá được sử dụng để giải mã ***tệp cuối***. Tệp sau giải mã được giải nén để có tệp tin ban đầu

Ý tưởng cài đặt

Ngôn ngữ lập trình

Cấu trúc chương trình

1. Xây dựng lớp *ConvertStringTobyte.cs* gồm các phương thức:

- ***readfileTostring:*** Đọc tệp có định dạng tuỳ ý trả về một xâu ký tự
- ***Createfile***: Từ xâu ký tự ghi thành tệp gốc (giữ nguyên nội dung và

2. Nén dữ liệu bằng phương pháp LZW

a. Nén dữ liệu:

Input: đường dẫn tệp tin cần nén

Output: Tệp tin đã nén (với tên: tên file gốc + phần mở rộng lzw)

Tệp tin nén sẽ được đặt cùng thư mục với tệp tin gốc

Trong quá trình nén sử dụng một bảng băm để lưu từ điển nén theo thuật toán LZW đã trình bày trong phần 3.3.2.2 mục 3.

b. Giải nén dữ liệu:

+ Input: đường dẫn tệp cần tin nén;

+ Output: tệp tin giải nén (có tên: tên của tập tin ban đầu)

3. Nén bằng phương pháp GZIP – Xây dựng lớp: *Gzip.cs*

- Sử dụng thư viện có sẵn của C#.Net: System.Io.Compression
- Sử dụng lớp có sẵn trong thư viện: Gzipstream gồm hai phương thức CompressMode và DecompressMode
- Phương thức **Compress**

+ Input: đường dẫn tệp tin cần nén; đường dẫn tệp tin sau khi nén

+ Output: tệp tin nén (tên file nén = tên tệp ban đầu + phần mở rộng ***.gzip***)

- Phương thức **Decompress**

+ Input: đường tệp tin nén, đường dẫn tệp tin giải nén

+ Output: tệp tin nén

4. Huffman – xây dựng hai lớp: *HuffmanTree.cs và Node.cs*

a. Lớp HuffmanTree.cs

Phương thức: ***Build (string)***

Phương thức: ***BitArray Encode (string);***

Phương thức String ***Decode (BitArray);***

b. Node.cs

Gồm có các thuộc tính:

char Symbol: ký tự của node

int Frequency: trọng số của node

Node Right: cây con trái

Node Left: cây con phải
Phương thức: Traverse(char symbol, List<bool> data): trả về dãy bit để mã hoá ký tự symbol.

5. Mã hoá tệp tin (AES Encryption)

- Thuật toán mã hoá AES được đặt trong lớp **AES.cs**
- Trong C#. Net có thư viện System.Security.Cryptography; hỗ trợ mã hoá tệp tin bằng thuật toán AES

a. Mã hoá - thủ tục *Encrypt*

Input: bản rõ, khoá AES, (độ dài khoá)
Output: bản mã

b. Giải mã - thủ tục *Desecrypt*

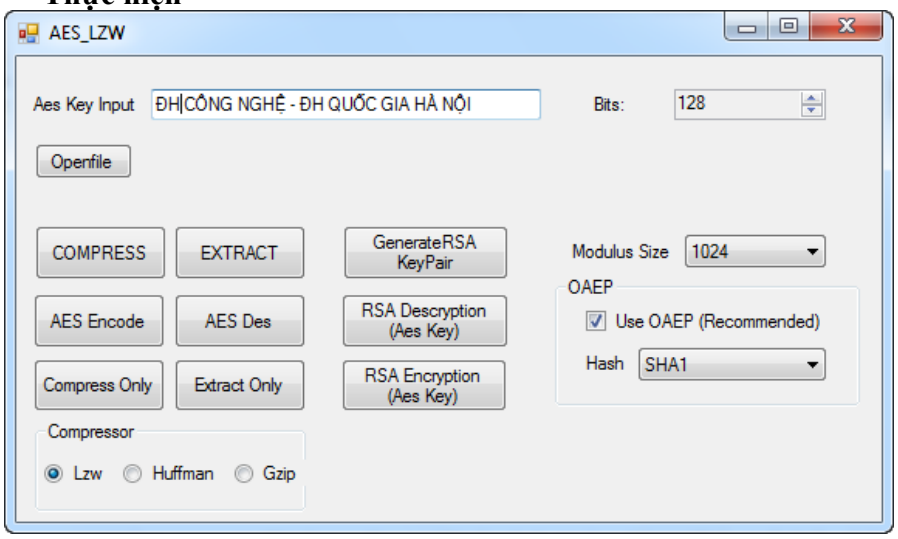
- Quá trình giải mã gồm các bước
- Tạo khoá hợp lệ cho quá trình giải mã:
- Chuyển văn bản mã hoá về dãy byte
- Giải mã dãy byte với cặp khoá hợp lệ vừa tạo ra và trả về xâu ký tự.

6. Mã hoá khoá công khai RSA

Sử dụng thư viện System.Security.Cryptography; của C#.Net. Trong thư viện này sử dụng lớp RSACryptoServiceProvider:
Tạo khoá bí mật RSA RSAProvider.ToXmlString(true)
Tạo khoá công khai RSA RSAProvider.ToXmlString(false)

- Mã hoá từng khối dữ liệu với đầu vào là một dãy byte và trả về dãy byte tương ứng, sau đó dãy byte này được chuyển về một xâu kí tự của bản mã
- Giải mã từng từng khối dữ liệu với đầu vào là một dãy byte và trả về dãy byte tương ứng sau đó dãy byte này được chuyển thành xâu kí tự trong bản rõ

Thực hiện



Hình 4.8: Giao diện chương trình thử nghiệm

Đánh giá

a. Một số kết quả thực hiện chương trình

- Mục tiêu:

+ Thử nghiệm các chức năng thực hiện của chương trình
+ So sánh hiệu quả nén giữa các thuật toán nén và kích thước tệp vừa nén, vừa mã hoá với tệp tin chỉ mã hoá hoặc mã hoá trước nén sau
+ Đánh giá về mặt thời gian thực hiện các thuật toán

b. Đánh giá

Hiệu quả nén của mỗi thuật toán phụ thuộc vào nhiều yếu tố: loại tệp tin, nội dung của mỗi loại tệp tin… Với tệp văn bản Huffman không hiệu quả bằng LZW, Gzip, tuy nhiên với tệp hình ảnh cho thấy Huffman có hiệu quả hơn so với LZW, Gzip.

Ngoài ra, hiệu quả nén còn phụ thuộc vào hiệu quả của cách xây dựng từ điển và cấu trúc dữ liệu sử dụng để cài đặt chương trình, định dạng của tệp, kích thước, độ lặp của dữ liệu, kiểu file lưu trữ…

- Loại tệp: tệp wordocument (Luanvan.doc) dung lượng: **1505 KB**

STT	Chương trình	Dung lượng sau khi thực hiện	Tỉ lệ
1.	LZW	1251 KB	16.9 %
2.	Huff	1332 KB	11.5 %
3.	Gzip	966 KB	35.8 %
4.	Lzw + AES	1776 KB	-10.8 %
5.	Huff + AES	1668 KB	-18 %
6.	Gzip + AES	1288 KB	14.4 %
7.	AES	2007 KB	-33.4 %
8.	GPG	849 KB	43.6 %

- Loại tệp: JPG (Picture.JPG) dung lượng: **3656KB**

STT	Chương trình	Dung lượng sau khi thực hiện	Tỉ lệ
1.	LZW	3958 KB	-0.1 %
2.	Huff	3655 KB	0 %
3.	Gzip	3659 KB	-8.3 %
4.	GPG	3656 KB	0 %
5.	Winzar	3656 KB	0 %

Bảng 4.1: Bảng kết quả thử nghiệm đánh giá hiệu quả nén

c. Các nghiên cứu có liên quan

Đã có những nghiên cứu phối hợp các phương pháp nén và mã hoá thông tin như GPG, PGP. Các ứng dụng PGP giờ đây bao gồm: thư điện tử, 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.Một điểm rất mạnh của GPG là cho phép quản lý, phân phối và thu hồi khoá thông qua các server, cho phép lưu các khoá trên các mail riêng của người dùng.

KẾT LUẬN

Luận văn có các kết quả chính sau:

1) Nghiên cứu tài liệu để hệ thống lại các vấn đề sau:

Về nén dữ liệu:

- Trình bày những kiến thức cơ bản nhất về mã nén
- Trình bày hai mô hình nén dữ liệu không tổn hao và các mã nén cơ bản trong hai mô hình đó (Fanno, Shanon, Huffman, LZ78, LZW)

Về mã hoá dữ liệu:

- Trình bày chuẩn mã hoá AES
- Trình bày cơ sở lý thuyết số học và các thuật toán quan trọng trong mã hoá RSA
- Trình bày các chuẩn kỹ thuật cơ bản trong PKCS#1 v2.1

Đề xuất giải pháp phối hợp quá trình nén và mã hoá tệp tin, phân tích các quá trình thực nghiệm của giải pháp.

2) Chương trình thử nghiệm

- Sử dụng ngôn ngữ lập trình C#.Net để cài đặt chương trình gồm:
- Nén tập tin bằng các thuật toán: LZW, Huffman, Gzip.
- Mã hoá tập tin nén bằng chuẩn mã hoá AES
- Sử dụng mã hoá RSA chuẩn PKCS #1 bảo vệ khoá của chuẩn mã hoá AES

3) Hướng phát triển

- Phát triển chương trình cho phép tích hợp trên các ứng dụng, cho phép quản lý và phân phối khoá trên mạng Internet.
- Phát triển như một module bảo mật trên hệ điều hành Windows.

References

TIẾNG VIỆT

1. Nguyễn Văn Hộ (2008), *Xác suất thống kê*, NXB Giáo dục

2. Nguyễn Thuý Vân (2001), *Lý thuyết mã*, NXB Khoa học và kỹ thuật

3. Bùi Minh Tiêu (1979), *Lý thuyết truyền tin*

TIẾNG ANH

4. Mark Nelson, *The Data Compression Book*

5. RSA laboratories (2002), PKCS#1 v2.1, *RSA Cryptography Standard*

6. *Federal Information Processing Standards Publication 197*

7. Micheal Malenkov, Christopher J.Dtra, Marco T. Morazan, *A New Bignum Multiplication Algorithm,* Seton Hall University, Department of Mathematics and Coputer Science

8. *Source programming Cookbook for C & C++*

9. http://dummy.codespeech.com/types-of-cryptography

10. http://www.rsa.com/rsalabs/node.asp?id=2125

11. http://en.wikipedia.org/wiki/Rijndael_S-box

12. http://vi.wikipedia.org/wiki/AES_(mahoa)

13. http://www.fadden.com/techmisc/hdc/index.htm

14. http://brskari.wordpress.com/2010/03/21/compression-before-encryption/