ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

MẬT MÃ & AN NINH MẠNG (CO3069)

Bài báo cáo

Bài tập lớn 1

GVHD: Nguyễn Hữu Hiếu
SV thực hiện: Trần Công Khanh 1511503
Đỗ Đức Hoài 1511093
Hoàng Hồng Khang 1511466

Tp. Hồ Chí Minh, Tháng 04/2018

Mục lục

Tóm tắt **3**

1 Giới thiệu **3**

2 Nội dung chính **3**
2.1 Tổng quan về chương trình 3
2.2 Cơ sở lý thuyết 4
2.2.1 Giải thuật DES 4
2.2.2 Giải thuật AES 8
2.2.3 Giải thuật RSA 11
2.2.4 Các giải thuật Hash 14
2.3 Tính năng cơ bản 15
2.4 Tính năng nâng cao 16

3 Phân tích và kết luận **16**

4 Hướng phát triển **17**

5 Tham khảo **18**

Phụ lục 1: Bảng đánh giá **19**

Phụ lục 2: Hướng dẫn sử dụng chương trình **20**

LỜI NÓI ĐẦU

Như chúng ta đã biết ngay từ xa xưa con người đã tìm hiểu và tiếp cận với các phương thức truyền thông. Trong một thời đại không có điện thoại cũng chẳng có thư điện tử, bất kỳ ai muốn truyền thông tin riêng của mình đến một người nào khác ở nơi xa chỉ có một cách là viết nó lên mặt giấy rồi giao phó cho một người khác mang thư đi. Nếu chẳng may người mang thư đó ngờ là trong thư có những thông tin quý giá, anh ta có thể bán cho kẻ thù để kiếm được nhiều tiền hơn là đưa nó đến đúng địa chỉ. Từ đó, nhiều bộ óc trong lịch sử đã phát minh ra các phương pháp sử dụng mật mã để giải quyết những thách thức trong việc bảo vệ dữ liệu. Julius Caesar phát minh ra cách viết mật mã mà chúng ta hay gọi là hộp Caesar. Mary - Nữ hoàng Scotland - đã tạo ra mật mã thay thế và gửi đi những thông báo bí mật từ nhà tù. Nhà khoa học xuất sắc người A-rập Abu Yusuf Ismail al-Kindi đã bảo vệ được những bí mật của mình bằng một mật mã thay thế tài tình sử dụng nhiều chữ cái khác nhau,...

Ngày nay, khi khoa học phát triển, mật mã cũng được sử dụng trong những hệ thống cần độ bảo mật cao như: hệ thống cơ sở dữ liệu của ngân hàng, bưu chính,...Với sự phát triển của các thiết bị thông minh mà điển hình là những chiếc máy tính có khả năng xử lý, tính toán tốc độ cao, những thuật toán mã hóa được phát triển ngày càng nhiều và có độ bảo mật cao hơn.

Tóm tắt

Mục tiêu chính của báo cáo là nhằm giúp mang lại cái nhìn tổng quan về mã hóa, đi sâu vào trình bày cụ thể ba giải thuật mã hóa mà nhóm sử dụng (DES, AES, và RSA), và ứng dụng nó vào trong chương trình hỗ trợ mã hóa và giải mã các tập tin cho người dùng. Bài báo cáo sẽ gồm 5 phần chính bao gồm:

Phần 1: Giới thiệu. Nội dung trong phần này sẽ được trình bày bao gồm: giới thiệu ngắn gọn về mã hóa, tổng quan các công việc, và phạm vi đề tài của nhóm.

Phần 2: Nội dung công việc đã làm. Nội dung trong phần này sẽ được trình bày bao gồm: tổng quan về chương trình mà nhóm đã hiện thực, lí do chọn giải thuật mã hóa, cơ sở lý thuyết tổng quan cũng như chứng minh độ an toàn của giải thuật được chọn, trình bày cụ thể các tính năng cơ bản và nâng cao của chương trình.

Phần 3: Phân tích và kết luận. Nội dung trong phần này sẽ được trình bày bao gồm: tổng kết lại kết quả đạt được, đánh giá kết quả và mặt hạn chế.

Phần 4: Hướng phát triển. Nội dung trong phần này sẽ được trình bày bao gồm: nêu các công việc chưa đươc giải quyết và hướng phát triển trong tương lai.

Phần 5: Tham khảo. Nội dung trong phần này sẽ được trình bày bao gồm: danh sách các tài liệu tham khảo: sách, báo, đường dẫn Internet,...

1 Giới thiệu

Mã hóa là công việc được thực hiện với mục đích làm cho dữ liệu không thể đọc được bởi bất cứ ai, ngoại trừ những ai được cho phép đọc. Mã hóa sử dụng thuật toán và khóa để biến đổi dữ liệu từ hình thức đơn giản rõ ràng (plaintext hay cleartext) sang hình thức mật mã vô nghĩa (code hay ciphertext). Chỉ có những ai có thông tin giải mã thì mới có thể giải mã và đọc được dữ liệu. Trong phạm vị bài tập lớn này, nhóm chúng em xin được trình bày ba giải thuật mã hóa đó là DES, AES, và RSA, cũng như việc ứng dụng các thư viện lập trình để xây dựng một chương trình mã hóa và giải mã các tập tin và thư mục sử dụng ba giải thuật mã hóa trên.

2 Nội dung chính

2.1 Tổng quan về chương trình

Chương trình mã hóa và giải mã được trình bày trong bài tập lớn này sẽ giúp bảo vệ dữ liệu khỏi những truy cập không mong muốn bằng cách khóa chúng bằng một chìa khóa (encryption key) có chức năng tương đương với một mật khẩu do chính người dùng tạo ra, hoặc nhờ chương trình sinh mã tự động. Nếu người dùng quên chìa khóa

(encryption key) sẽ mất khả năng truy cập những dữ liệu đó. Chương trình sử dụng việc mã hóa để bảo vệ thông tin của người dùng. Không chỉ mã hóa một số tệp riêng biệt, chương trình còn có khả năng mã hóa và giải mã toàn bộ tập tin bên trong thư mục được chọn.

Chương trình được viết bằng ngôn ngữ C#, và sử dụng thư viện lập trình mã hóa *Cryptography* của ngôn ngữ này, với giao diện dễ nhìn, chương trình sẽ dễ dàng sử dụng với người dùng ngay từ lần đầu tiên tiếp xúc.

Chương trình có khả năng mã hóa một tập tin bất kì ví dụ như: hình ảnh, âm thanh, video, pdf, doc, docx, ppt, zip,...và với bất kỳ dung lượng nào, thành một tập tin với định dạng *".encrypted"* và trong khoảng thời gian tương đối ngắn.

Chương trình hỗ trợ ba giải thuật mã hóa, trong đó bao gồm hai giải thuật mã hóa đối xứng là DES, AES, và một giải thuật mã hóa bất đối xứng là RSA. Ngoài ra, chương trình còn cung cấp các giải thuật Hash (MD5, SHA-1, SHA-256, SHA-384, SHA-512, RIPEMD-160) giúp người dùng có thể kiểm tra tính toàn vẹn của một tập tin.

2.2 Cơ sở lý thuyết

2.2.1 Giải thuật DES

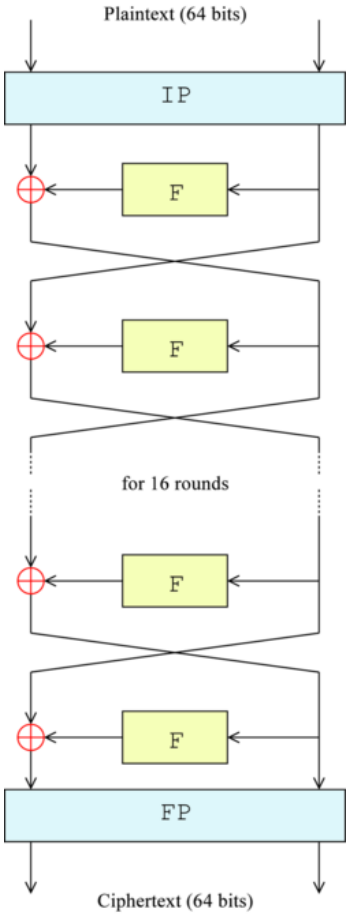
Mô tả thuật toán

DES là thuật toán mã hóa khối: nó xử lý từng khối thông tin của bản rõ có độ dài xác định và biến đổi theo những quá trình phức tạp để trở thành khối thông tin của bản mã có độ dài không thay đổi. Trong trường hợp của DES, độ dài mỗi khối là 64 bit. DES cũng sử dụng khóa để cá biệt hóa quá trình chuyển đổi. Nhờ vậy, chỉ khi biết khóa mới có thể giải mã được văn bản mã. Khóa dùng trong DES có độ dài toàn bộ là 64 bit. Tuy nhiên chỉ có 56 bit thực sự được sử dụng; 8 bit còn lại chỉ dùng cho việc kiểm tra. Vì thế, độ dài thực tế của khóa chỉ là 56 bit.

Cấu trúc tổng thể của thuật toán được thể hiện ở Hình [1]: có 16 chu trình giống nhau trong quá trình xử lý. Ngoài ra còn có hai lần hoán vị đầu và cuối (initial and final permutation - IP & EP). Hai quá trình này có tính chất đối nhau (trong quá trình mã hóa thì IP trước EP, khi giải mã thì ngược lại). IP và EP không có vai trò xét về mật mã học và việc sử dụng chúng chỉ có ý nghĩa đáp ứng cho quá trình đưa thông tin vào và lấy thông tin ra từ các khối phần cứng có từ thập niên 1970. Trước khi đi vào 16 chu trình chính, khối thông tin 64 bit được tách làm hai phần 32 bit và mỗi phần sẽ được xử lý tuần tự (quá trình này còn được gọi là mạng Feistel).

Cấu trúc của thuật toán (mạng Feistel) đảm bảo rằng quá trình mã hóa và giải mã diễn ra tương tự. Điểm khác nhau chỉ ở chỗ các khóa con được sử dụng theo trình tự ngược nhau. Điều này giúp cho việc thực hiện thuật toán trở nên đơn giản, đặc biệt là

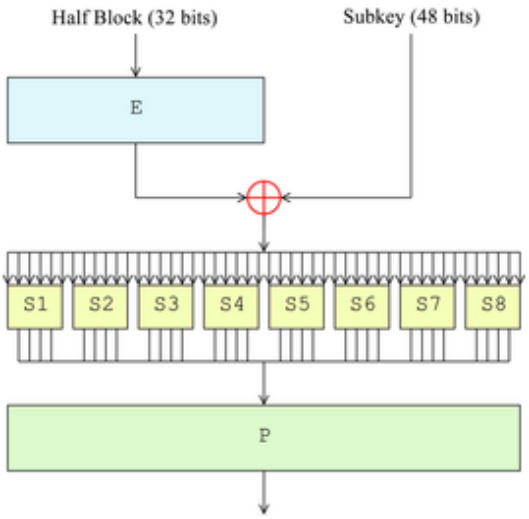
khi thực hiện bằng phần cứng.



Hình 1: *Cấu trúc thuật toán Feistel dùng trong DES*

Hàm F, như được miêu tả ở Hình [2], hoạt động trên khối 32 bit và bao gồm bốn giai đoạn:

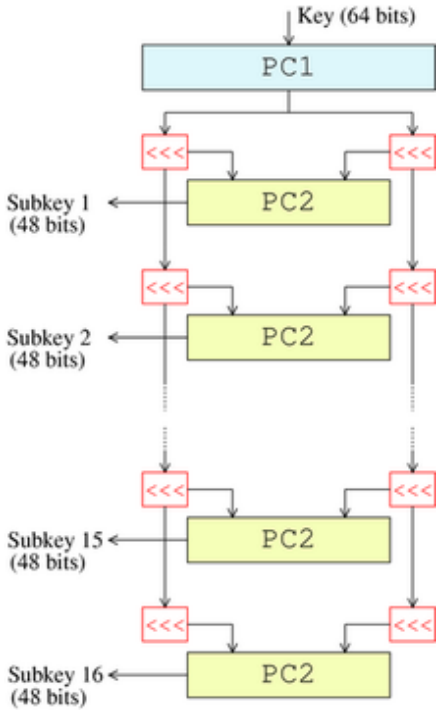
- *Mở rộng:* 32 bit đầu vào được mở rộng thành 48 bit sử dụng thuật toán hoán vị mở rộng (expansion permutation) với việc nhân đôi một số bit. Giai đoạn này được ký hiệu là E trong sơ đồ.
- *Trộn khóa:* 48 bit thu được sau quá trình mở rộng được XOR với khóa con. Mười sáu khóa con 48 bit được tạo ra từ khóa chính 56 bit theo một chu trình tạo khóa con (key schedule) miêu tả ở phần sau.
- *Thay thế:* 48 bit sau khi trộn được chia làm 8 khối con 6 bit và được xử lý qua hộp thay thế S-box. Đầu ra của mỗi khối 6 bit là một khối 4 bit theo một chuyển đổi phi tuyến được thực hiện bằng một bảng tra. Khối S-box đảm bảo phần quan trọng cho độ an toàn của DES. Nếu không có S-box thì quá trình sẽ là tuyến tính và việc thám mã sẽ rất đơn giản.



Hình 2: *Hàm F (F-function) dùng trong DES*

- *Hoán vị:* Cuối cùng, 32 bit thu được sau S-box sẽ được sắp xếp lại theo một thứ tự cho trước (còn gọi là P-box).

  Quá trình luân phiên sử dụng S-box và sự hoán vị các bit cũng như quá trình mở rộng đã thực hiện được tính chất gọi là sự xáo trộn và khuyếch tán (*confusion and diffusion*).



Hình 3: *Quá trình tạo khóa con trong DES*

Quá trình tạo khóa con: Hình [3] mô tả thuật toán tạo khóa con cho các chu trình. Đầu tiên, từ 64 bit ban đầu của khóa, 56 bit được chọn (Permuted Choice 1, hay PC-1); 8 bit còn lại bị loại bỏ. 56 bit thu được được chia làm hai phần bằng nhau, mỗi phần được xử lý độc lập. Sau mỗi chu trình, mỗi phần được dịch đi 1 hoặc 2 bit (tùy thuộc từng chu trình). Các khóa con 48 bit được tạo thành bởi thuật toán lựa chọn 2 (Permuted Choice 2, hay PC-2) gồm 24 bit từ mỗi phần. Quá trình dịch bit (được ký hiệu là "<<<" trong sơ đồ) khiến cho các khóa con sử dụng các bit khác nhau của khóa chính; mỗi bit được sử dụng trung bình 14 trong tổng số 16 khóa con.

Quá trình tạo khóa con khi thực hiện giải mã cũng diễn ra tương tự nhưng các khóa con được tạo theo thứ tự ngược lại. Ngoài ra sau mỗi chu trình, khóa sẽ được dịch phải thay vì dịch trái như khi mã hóa.

An toàn và sự giải mã

Tấn công kiểu duyệt toàn bộ: Đối với bất cứ phương pháp mã hóa nào, kiểu tấn công cơ bản và đơn giản nhất là tấn công kiểu duyệt toàn bộ: thử lần lượt tất cả các khóa có thể cho đến khi tìm ra khóa đúng. Độ dài của khóa sẽ xác định số lượng phép thử tối đa cần thực hiện và do đó thể hiện tính khả thi của phương pháp. Trong trường hợp của DES, nghi ngờ về độ an toàn của nó đã được đặt ra ngay từ khi nó chưa trở thành tiêu chuẩn.

Các kiểu tấn công khác:

- *Phá mã vi sai:* được Eli Biham và Adi Shamir tìm ra vào cuối những năm 1980 mặc dù nó đã được IBM và NSA biết đến trước đó. Để phá mã DES với đủ 16 chu trình, phá mã vi sai cần đến 2^{47} văn bản rõ. DES đã được thiết kế để chống lại tấn công dạng này.
- *Phá mã tuyến tính:* được tìm ra bởi Mitsuru Matsui và nó đòi hỏi 2^{43} văn bản rõ (Matsui, 1993). Phương pháp này đã được Matsui thực hiện và là thực nghiệm phá mã đầu tiên được công bố. Không có bằng chứng chứng tỏ DES có khả năng chống lại tấn công dạng này.
- *Phá mã Davies:* trong khi phá mã vi sai và phá mã tuyến tính là các kỹ thuật phá mã tổng quát, có thể áp dụng cho các thuật toán khác nhau, phá mã Davies là một kỹ thuật dành riêng cho DES. Dạng tấn công này được đề xuất lần đầu bởi Davies vào cuối những năm 1980 và cải tiến bởi Biham và Biryukov (1997). Dạng tấn công mạnh nhất đòi hỏi 2^{50} văn bản rõ, độ phức tạp là 2^{50} và có tỷ lệ thành công là 51%.

Một vài đặc điểm về cách giải mã

DES có tính chất bù:

$$E_K(P) = C \Leftrightarrow E_{\overline{K}}(\overline{P}) = \overline{C}$$

trong đó $\overline{x}$ là phần bù của x theo từng bit (1 thay bằng 0 và ngược lại). E_K là bản mã hóa của E với khóa K. P và C là văn bản rõ (trước khi mã hóa) và văn bản mã (sau khi mã hóa). Do tính bù, ta có thể giảm độ phức tạp của tấn công duyệt toàn bộ xuống 2 lần (tương ứng với 1 bit) với điều kiện là ta có thể lựa chọn bản rõ.

Ngoài ra DES còn có 4 khóa yếu (weak keys). Khi sử dụng khóa yếu thì mã hóa (E) và giải mã (D) sẽ cho ra cùng kết quả:

$$E_K(E_K(P)) = P \Leftrightarrow E_K = D_K$$

Bên cạnh đó, còn có 6 cặp khóa nửa yếu (semi-weak keys). Mã hóa với một khóa trong cặp, K_1, tương đương với giải mã với khóa còn lại, K_2:

$$E_{K_1}(E_{K_2}(P)) = P \Leftrightarrow E_{K_1} = D_{K_2}$$

Tuy nhiên có thể dễ dàng tránh được những khóa này khi thực hiện thuật toán, có thể bằng cách thử hoặc chọn khóa một cách ngẫu nhiên. Khi đó khả năng chọn phải khóa yếu là rất nhỏ.

Lý do chọn giải thuật DES: Tuy độ bảo mật của giải thuật DES không cao, nhưng ngày này giải thuật DES vẫn thường được sử dụng phổ biến. Mặt khác giải thuật DES đã từng được chọn làm chuẩn chính thức và chuẩn này được sử dụng rộng rãi trên phạm vi thế giới. Do đó, độ thông dụng của giải thuật này tương đối lớn, và được rất nhiều người biết đến.

2.2.2 Giải thuật AES

Mô tả thuật toán

Mặc dù hai tên AES và Rijndael vẫn thường được gọi thay thế cho nhau nhưng trên thực tế thì hai thuật toán không hoàn toàn giống nhau. AES chỉ làm việc với các khối dữ liệu (đầu vào và đầu ra) 128 bit và khóa có độ dài 128, 192 hoặc 256 bit trong khi Rijndael có thể làm việc với dữ liệu và khóa có độ dài bất kỳ là bội số của 32 bit nằm trong khoảng từ 128 tới 256 bit. Các khóa con sử dụng trong các chu trình được tạo ra bởi quá trình tạo khóa con Rijndael. Mỗi khóa con cũng là một cột gồm 4 byte. Hầu hết các phép toán trong thuật toán AES đều thực hiện trong một trường hữu hạn của các byte. Mỗi khối dữ liệu 128 bit đầu vào được chia thành 16 byte (mỗi byte 8 bit), có thể xếp thành 4 cột, mỗi cột 4 phần tử hay là một ma trận 4 x 4 của các byte, nó được gọi là ma trận trạng thái, hay vắn tắt là trạng thái (tiếng Anh: state, trang thái trong Rijndael có thể có thêm cột). Trong quá trình thực hiện thuật toán các toán tử tác động để biến đổi ma trận trạng thái này.

Quá trình mã hóa

Bao gồm các bước:

1. Khởi động vòng lặp
    + AddRoundKey — Mỗi cột của trạng thái đầu tiên lần lượt được kết hợp với một khóa con theo thứ tự từ đầu dãy khóa.
2. Vòng lặp
    + SubBytes — đây là phép thế (phi tuyến) trong đó mỗi byte trong trạng thái sẽ được thế bằng một byte khác theo bảng tra (Rijndael S-box).
    + ShiftRows — dịch chuyển, các hàng trong trạng thái được dịch vòng theo số bước khác nhau.
    + MixColumns — quá trình trộn làm việc theo các cột trong khối theo một phép biến đổi tuyến tính.
    + AddRoundKey
3. Vòng lặp cuối
    + SubBytes
    + ShiftRows
    + AddRoundKey

Tại chu trình cuối thì bước MixColumns không thực hiện.

Bước SubBytes: Các byte được thế thông qua bảng tra S-box. Đây chính là quá trình phi tuyến của thuật toán. Hộp S-box này được tạo ra từ một phép biến đổi khả nghịch trong trường hữu hạn GF(2^8) có tính chất phi tuyến. Để chống lại các tấn công dựa trên các đặc tính đại số, hộp S-box này được tạo nên bằng cách kết hợp phép nghịch đảo với một phép biến đổi affine khả nghịch. Hộp S-box này cũng được chọn để tránh các điểm bất động (fixed point).

Bước ShiftRows: Các hàng được dịch vòng một số bước nhất định. Đối với AES, hàng đầu được giữ nguyên. Mỗi byte của hàng thứ 2 được dịch vòng trái một vị trí. Tương tự, các hàng thứ 3 và 4 được dịch vòng 2 và 3 vị trí. Do vậy, mỗi cột khối đầu ra của bước này sẽ bao gồm các byte ở đủ 4 cột khối đầu vào. Đối với Rijndael với độ dài khối khác nhau thì số vị trí dịch chuyển cũng khác nhau.

Bước MixColumns: Bốn byte trong từng cột được kết hợp lại theo một phép biến đổi tuyến tính khả nghịch. Mỗi khối 4 byte đầu vào sẽ cho một khối 4 byte ở đầu ra với tính chất là mỗi byte ở đầu vào đều ảnh hưởng tới cả bốn byte đầu ra. Cùng với bước ShiftRows, MixColumns đã tạo ra tính chất khuếch tán cho thuật toán.

Mỗi cột được xem như một đa thức trong trường hữu hạn và được nhân với đa thức $c(x) = 3x^3 + x^2 + x + 2$ $(modulo\ x^4 + 1)$. Vì thế, bước này có thể được xem là phép nhân ma trận trong trường hữu hạn.

Bước AddRoundKey: Tại bước này, khóa con được kết hợp với các khối. Khóa con trong mỗi chu trình được tạo ra từ khóa chính với quá trình tạo khóa con Rijndael; mỗi khóa con có độ dài giống như các khối. Quá trình kết hợp được thực hiện bằng cách XOR từng bit của khóa con với khối dữ liệu.

Độ an toàn của giải thuật

Vào thời điểm năm 2006, dạng tấn công lên AES duy nhất thành công là tấn công kênh bên (side channel attack).

Phương pháp thường dùng nhất để tấn công các dạng mã hóa khối là thử các kiểu tấn công lên phiên bản có số chu trình thu gọn. Đối với khóa 128 bit, 192 bit và 256 bit, AES có tương ứng 10, 12 và 14 chu trình. Tại thời điểm năm 2006, những tấn công thành công được biết đến là 7 chu trình đối với khóa 128 bit, 8 chu trình với khóa 192 bit và 9 chu trình với khóa 256 bit.

Một số nhà khoa học trong lĩnh vực mật mã lo ngại về an ninh của AES. Họ cho rằng ranh giới giữa số chu trình của thuật toán và số chu trình bị phá vỡ quá nhỏ. Nếu các kỹ thuật tấn công được cải thiện thì AES có thể bị phá vỡ. Ở đây, phá vỡ có nghĩa chỉ bất cứ phương pháp tấn công nào nhanh hơn tấn công kiểu duyệt toàn bộ. Vì thế một tấn công cần thực hiện 2^{120} cũng được coi là thành công mặc dù tấn công này chưa thể thực hiện trong thực tế. Tại thời điểm hiện nay, nguy cơ này không thực sự nguy hiểm và có thể bỏ qua. Tấn công kiểu duyệt toàn bộ quy mô nhất đã từng thực hiện là do distributed.net thực hiện lên hệ thống 64 bit RC5 vào năm 2002.

Một vấn đề khác nữa là cấu trúc toán học của AES. Không giống với các thuật toán mã hóa khác, AES có mô tả toán học khá đơn giản. Tuy điều này chưa dẫn đến mối nguy hiểm nào nhưng một số nhà nghiên cứu sợ rằng sẽ có người lợi dụng được cấu trúc này trong tương lai.

Lý do chọn giải thuật AES: Thuật toán AES thực hiện việc xử lý rất nhanh. Mã chương trình ngắn gọn, thao tác xử lý sử dụng ít bộ nhớ. Tất cả các bước xử lý của việc mã hóa và giải mã đều được thiết kế thích hợp với cơ chế xử lý song song. Yêu cầu đơn giản trong việc thiết kế cùng tính linh hoạt trong xử lý được đáp ứng nên nhóm em quyết định chọn giải thuật này để thực hiện.

2.2.3 Giải thuật RSA

Mô tả thuật toán

Thuật toán RSA có hai khóa: khóa công khai và khóa bí mật. Mỗi khóa là những giá trị cố định sử dụng trong quá trình mã hóa và giải mã. Khóa công khai được công bố rộng rãi cho mọi người và được dùng để mã hóa. Những thông tin được mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa bí mật tương ứng. Nói cách khác, mọi người đều có thể mã hóa nhưng chỉ có người biết khóa cá nhân (bí mật) mới có thể giải mã được.

Ta có thể mô phỏng trực quan một hệ mật mã khoá công khai như sau: Bob muốn gửi cho Alice một thông tin mật mà Bob muốn duy nhất Alice có thể đọc được. Để làm được điều này, Alice gửi cho Bob một chiếc hộp có khóa đã mở sẵn và giữ lại chìa khóa. Bob nhận chiếc hộp, cho vào đó một tờ giấy viết thư bình thường và khóa lại (như loại khoá thông thường chỉ cần sập chốt lại, sau khi sập chốt khóa ngay cả Bob cũng không thể mở lại được - không đọc lại hay sửa thông tin trong thư được nữa). Sau đó Bob gửi chiếc hộp lại cho Alice. Alice mở hộp với chìa khóa của mình và đọc thông tin trong thư. Trong ví dụ này, chiếc hộp với khóa mở đóng vai trò khóa công khai, chiếc chìa khóa chính là khóa bí mật.

Sinh khóa

Mấu chốt cơ bản của việc sinh khóa trong RSA là tìm được bộ 3 số tự nhiên e, d và n sao cho:

$$m^{ed} \equiv m \; mod \; n$$

và một điểm không thể bỏ qua là cần bảo mật cho d sao cho dù biết e, n hay thậm chí cả m cũng không thể tìm ra d được.

Cụ thể, khóa của RSA được sinh như sau:

- Chọn 2 số nguyên tố p và q.
- Tính $n = pq$. Sau này, n sẽ được dùng làm modulus trong cả public key và private key.
- Tính một số giả nguyên tố bằng *phi hàm Carmichael* như sau: $\lambda(n) = BCNN(\lambda(p), \lambda(q)) = BCNN(p-1, q-1)$. Giá trị này sẽ được giữ bí mật.
- Chọn một số tự nhiên e trong khoảng $(1, \lambda(n))$ sao cho $ƯCLN(e, \lambda(n)) = 1$, tức là e và $\lambda(n)$ là hai số nguyên tố cùng nhau.
- Tính toán số d sao cho $d \equiv 1/e \; (mod \; \lambda(n))$ hay viết dễ hiểu hơn thì $de \equiv 1 \; (mod \; \lambda(n))$. Số d được gọi là số nghịch đảo modulo của e (theo modulo $mod \; \lambda(n)$).

Public key sẽ là bộ số (n, e), và private key sẽ là bộ số (n, d). Chúng ta cần giữ private key thật cẩn thận cũng như các số nguyên tố p và q vì từ đó có thể tính toán các khóa rất dễ dàng.

Trong thực hành, chúng ta thường chọn e tương đối nhỏ để việc mã hóa và giải mã nhanh chóng hơn. Giá trị thường được sử dụng là $e = 65537$. Ngoài ra, chúng ta có thể tính số giả nguyên tố bằng *phi hàm Euler* $\phi(n) = (p-1)(q-1)$ và dùng nó như $\lambda(n)$. Vì $\phi(n)$ là bội của $\lambda(n)$ nên các số d thỏa mãn $de \equiv 1 \ (mod \ \phi(n))$ cũng sẽ thỏa mãn $d \equiv 1/e \ (mod \ \lambda(n))$. Tuy nhiên, một số tác dụng phụ của việc này là d thường sẽ trở nên lớn hơn mức cần thiết.

Mã hóa và giải mã

Trong phần này, chúng ta sẽ tìm hiểu cách mã hóa với public key (n, e) và giải mã với private key (n, d).

Nếu chúng ta có bản rõ M, chúng ta cần chuyển nó thành một số tự nhiên m trong khoảng (0, n) sao cho m, n là hai số nguyên tố cùng nhau. Việc này rất dễ dàng thực hiện bằng cách thêm một trong các kỹ thuật padding. Tiếp theo, chúng ta sẽ mã hóa m, thành c như sau:

$$c \equiv m^e \ mod \ n$$

Sau đó giá trị c sẽ được chuyển cho người nhận.

Ở phía người nhận, họ sẽ giải mã từ c để lấy được m như sau:

$$c^d \equiv m^{de} \equiv m \ mod \ n$$

Từ m có thể lấy lại được bản tin bằng cách đảo ngược padding. Chúng ta lấy một ví dụ đơn giản như sau:

$$p = 5, q = 7$$

$$\Rightarrow n = pq = 35$$

$$\Rightarrow \phi(n) = 24$$

Chúng ta chọn e = 5 vì ƯCLN(5, 24) = 1, cuối cùng chọn d = 29 vì ed - 1 = 29 x 5 - 1 chia hết cho 24.

Giả sử m = 32 (dấu cách), chúng ta sẽ mã hóa m và thu được:

$$c = 32^5 \ \% \ 35 = 2$$

Giải mã c để thu được m:

$$m = 2^{29} \ \% \ 35 = 32$$

Đây chính là m ban đầu. Bạn có thể thử m với các giá trị khác nhau để thấy thuật toán hoàn toàn chính xác. Việc chứng minh RSA hoạt động chính xác là một vấn đề toán học khá phức tạp mà bài viết này sẽ không đi vào chi tiết của nó.

Mức độ bảo mật của RSA phụ thuộc rất lớn vào khả năng phân tích thừa số nguyên tố của các số lớn. Bởi vì chúng ta cung cấp public key một cách rộng rãi, nếu việc phân tích thừa số nguyên tố đơn giản, thì việc bị lộ private key là không thể tránh khỏi.

Vì vậy, khi sinh khóa, chúng ta cần chọn các số nguyên tố p và q một cách ngẫu nhiên. Bản thân hai số nguyên tố này cũng rất lớn, và để việc phân tích thừa số nguyên tố khó khăn hơn, hai số nguyên tố này sẽ không có cùng độ dài. Trong tương lai gần, có lẽ vẫn chưa có một phương pháp hiệu quả nào cho phép thực hiện điều này với các máy tính cá nhân.

Tuy nhiên, với sự phát triển của công nghệ, các siêu máy tính xuất hiện ngày càng nhiều. Cùng với đó, máy tính lượng tử cho phép tính toán với tốc độ cao hơn rất nhiều có thể sẽ phá vỡ sự bảo mật của RSA.

Ngay từ năm 1993, thuật toán Shor đã được phát triển và chỉ ra rằng máy tính lượng tử có thể giải bài toán phân tích ra thừa số trong thời gian đa thức. Rất may là những điều này mới chỉ là lý thuyết vì đến thời điểm hiện tại và trong vài năm tới, máy tính lượng tử vẫn chưa được hoàn thiện.

Một câu hỏi khá thú vị là có thể đảo vai trò của public key và private key hay không? Tức là chúng ta dùng private key để mã hóa và dùng public key để giải mã hay không? Câu trả lời là có. Tuy nhiên, không ai làm như vậy cả vì rất đơn giản: từ public key rất khó để suy ra private key nhưng từ private key để suy ra public key thì rất đơn giản. Nếu cho người khác private key (để họ mã hóa), những kẻ tấn công "man in the middle" sẽ dễ dàng tính toán được public key và thay đổi hoàn toàn cuộc hội thoại của chúng ta. Như thế sẽ rất nguy hiểm.

Tuy nhiên, có một số trường hợp chúng ta đã đảo vai trò của private key và public key như vậy. Đó chính là khi chúng ta sử dụng chữ ký số.

Độ an toàn của giải thuật

Độ an toàn của hệ thống RSA dựa trên 2 vấn đề của toán học: bài toán phân tích ra thừa số nguyên tố các số nguyên lớn và bài toán RSA. Nếu 2 bài toán trên là bài toán khó (không tìm được thuật toán hiệu quả để giải chúng) thì không thể thực hiện được việc phá mã toàn bộ đối với RSA. Phá mã một phần phải được ngăn chặn bằng các phương pháp chuyển đổi bản rõ an toàn.

Bài toán RSA là bài toán tính căn bậc e mô-đun n (với n là hợp số): tìm số m sao

cho $m^e = c \, mod \, n$, trong đó (e, n) chính là khóa công khai và c là bản mã. Hiện nay phương pháp triển vọng nhất giải bài toán này là phân tích n ra thừa số nguyên tố. Khi thực hiện được điều này, kẻ tấn công sẽ tìm ra số mũ bí mật d từ khóa công khai và có thể giải mã theo đúng quy trình của thuật toán. Nếu kẻ tấn công tìm được 2 số nguyên tố p và q sao cho: $n = pq$ thì có thể dễ dàng tìm được giá trị $(p-1)(q-1)$ và qua đó xác định d từ e. Chưa có một phương pháp nào được tìm ra trên máy tính để giải bài toán này trong thời gian đa thức (polynomial-time). Tuy nhiên người ta cũng chưa chứng minh được điều ngược lại (sự không tồn tại của thuật toán).

Lý do chọn giải thuật RSA: Giải thuật RSA là một giải thuật mã hóa bất đối xứng tiêu biểu. Ngoài tác dụng mã hóa văn bản, file,... giải thuật RSA còn có một chức năng khác đó là sinh ra chữ ký số. Việc ký tên và xác thực chữ ký số sử dụng hệ mã hóa RSA tương tự như quá trình mã hóa mà giải mã. Tuy nhiên vai trò của public key và private key thì có thay đổi đôi chút. Để tạo chữ ký, người gửi sẽ dùng private key và người nhận sẽ dùng public key để xác thực chữ ký đó. Vì tính thông dụng và được ứng dụng rộng rãi nên nhóm em quyết định chọn giải thuật RSA để sử dụng.

2.2.4 Các giải thuật Hash

Hashing là một phương thức mật mã nhưng nó không phải là một thuật toán mã hoá. Hàm hash (hash function) là hàm một chiều mà nếu đưa một lượng dữ liệu bất kì qua hàm này sẽ cho ra một chuỗi có độ dài cố định ở đầu ra.

Dữ liệu đầu vào của bạn có thể là một file, một ổ đĩa một quá trình truyền thông tin trên mạng, hay một bức thư điện tử. Thông số hash value được sử dụng để phát hiện khi có sự thay đổi của dữ liệu đầu vào. Nói cách khác, hashing sử dụng nó để phát hiện ra dữ liệu có toàn vẹn trong quá trình lưu trữ hay trong khi truyền hay không.

Không như các phương thức mật mã khác (chúng sẽ làm thay đổi dữ liệu thành một dạng mật mã), quá trình hashing chỉ tính toán và đưa ra thông số hash value của dữ liệu và không thay đổi dữ liệu ban đầu.

Hai tính chất quan trọng của hàm này là:

+ Tính một chiều: không thể suy ra dữ liệu ban đầu từ kết quả, điều này tương tự như việc bạn không thể chỉ dựa vào một dấu vân tay lạ mà suy ra ai là chủ của nó được.

+ Tính duy nhất: xác suất để có một vụ va chạm (hash collision), tức là hai thông điệp khác nhau có cùng một kết quả hash, là cực kì nhỏ.

Một số ứng dụng của hàm hash:

+ Chống và phát hiện xâm nhập: chương trình chống xâm nhập so sánh giá trị hash của một file với giá trị trước đó để kiểm tra xem file đó có bị ai đó thay đổi hay không.

+ Bảo vệ tính toàn vẹn của thông điệp được gửi qua mạng bằng cách kiểm tra giá trị hash của thông điệp trước và sau khi gửi nhằm phát hiện những thay đổi cho dù là nhỏ nhất.

+ Tạo chìa khóa từ mật khẩu.

+ Tạo chữ kí điện tử.

Thuật toán hashing thường được sử dụng: SHA-1 và MD5 là hai hàm hash thông dụng nhất và được sử dụng trong rất nhiều hệ thống bảo mật. Vào tháng 8 năm 2004, tại hội nghị Crypto 2004, người ta đã tìm thấy va chạm đối với MD5 và SHA-0, một phiên bản yếu hơn của hàm hash SHA-1. Không bao lâu sau đó, vào khoảng giữa tháng 2 năm 2005, một nhóm ba nhà mật mã học người Trung Quốc đã phát hiện ra một phương pháp có thể tìm thấy va chạm đối với SHA-1 chỉ trong vòng 269 bước tính toán. Do đó, các chuyên gia khuyến cáo nên bắt đầu chuyển sang các hàm hash an toàn hơn như SHA-256, SHA-384 hay SHA-512.

Ngoài ra, còn rất nhiều các hàm băm khác được thiết kế nhằm tăng cường sự an toàn như RIPEMD-160. RIPEMD-160 là một hàm băm mật mã 160 bit, được thiết kế bởi Hans Dobbertin, Antoon Bosselaers và Bart Preneel. Nó được thiết kế để sử dụng như là một thay thế an toàn cho các hàm băm 128 bit MD4 , MD5 và RIPEMD, và dự kiến sẽ là thuật toán an toàn trong vòng mười năm tới hoặc hơn.

Trong phạm vi bài tập lớn này, nhóm chúng em sẽ hiện thực 6 giải thuật hash là: MD5, SHA-1, SHA-256, SHA-384, SHA-512, và RIPEMD-160.

2.3 Tính năng cơ bản

Chương trình đáp ứng được các tính năng cơ bản hỗ trợ người dùng trong quá trình mã hóa và giải mã như:

+ Chương trình hỗ trợ ba giải thuật mã hóa, trong đó bao gồm hai giải thuật mã hóa đối xứng là DES, AES, và một giải thuật mã hóa bất đối xứng là RSA. Ngoài ra, chương trình còn cung cấp các giải thuật Hash (MD5, SHA-1, SHA-256, SHA-384, SHA-512, RIPEMD-160) giúp người dùng có thể kiểm tra tính toàn vẹn của một tập tin.

+ Chương trình có khả năng mã hóa một tập tin bất kì ví dụ như: hình ảnh, âm thanh, video, pdf, doc, docx, ppt, zip,... và với bất kỳ dung lượng nào, thành một tập tin với định dạng *".encrypted"* và trong khoảng thời gian tương đối ngắn.

+ Quá trình mã hóa: nhận input là tập tin hoặc thư mục bất kì và chìa khóa mã hóa (có thể do người dùng nhập, hoặc chương trình tự sinh khóa) và một số tùy chọn tùy theo giải thuật mã hóa. Output là tập tin hay thư mục chứa dữ liệu đã được mã hóa. Ứng dụng cung cấp chức năng truy xuất đến thư mục chứa tập tin output.

+ Quá trình giải mã: nhận input là tập tin hay thư mục chứa dữ liệu đã được mã hóa và chìa khóa mã hóa (phải nhập chính xác chìa khóa khi mã hóa) và một số tùy chọn tùy theo giải thuật mã hóa. Output là một hoặc nhiều tập tin được giải mã thành công. Ứng dụng cung cấp chức năng truy xuất đến thư mục chứa tập tin output.

2.4 Tính năng nâng cao

Chương trình có một số tính năng nâng cao, giúp hỗ trợ người dùng tốt hơn, tiện lợi hơn trong việc mã hóa và giải mã như:

+ Chương trình ngoài việc cho phép người dùng tự nhập khóa (key) thì còn có khả năng tự sinh khóa ngẫu nhiên giúp gia tăng độ bảo mật.

+ Chương trình có khả năng mã hóa và giải mã toàn bộ tập tin trong một thư mục được chọn hoặc toàn bộ tập tin trong thư mục được chọn và các tập tin trong thư mục con của nó.

+ Chương trình có hiển thị thanh trạng thái thông báo tỉ lệ phần trăm trong quá trình mã hóa và giải mã.

3 Phân tích và kết luận

Mã Hóa (cryptography) là một trong những mặt phức tạp nhất của quá trình phát triển phần mềm mà bất kỳ nhà phát triển nào cũng sẽ sử dụng.

Lý thuyết kỹ thuật mật mã hiện đại cực kỳ khó hiểu và đòi hỏi một mức kiến thức toán học mà tương đối ít người có được.

Nhờ vào các ngôn ngữ lập trình hiện đại (ví dụ như C# mà nhóm đã sử dụng trong bài tập lớn này) và các thư viện hỗ trợ mã hóa (như Cryptography trên C#) đã cung cấp các hiện thực dễ sử dụng cho hầu hết các kỹ thuật mật mã thông dụng và hỗ trợ các giải thuật phổ biến nhất, nên việc ứng dụng và lập trình một chương trình hỗ trợ mã hóa và giải mã là tương đối đơn giản và nhanh chóng.

Hạn chế của chương trình này là chưa tích hợp các giải thuật Hash ngay trong chức năng Decrypt để người dùng có thể biết được ngay tính toàn vẹn của một tập tin. Nếu muốn biết tính toàn vẹn của tập tin người dùng phải nhấp vào chức năng Hash Algorithms nằm riêng lẻ.

4 Hướng phát triển

Chương trình hỗ trợ mã hóa có nhiều hướng phát triển khác nhau tùy theo nhu cầu, và mục đích của người sử dụng để chương trình có thể đáp ứng một cách tốt nhất cho người dùng như: tích hợp lớp mã hóa vào các ứng dụng có sẵn, thông dụng nhằm nâng cao tính bảo mật của ứng dụng đó, như ứng dụng chat, nhắn tin, file sharing để bảo mật dữ liệu trong quá trình gửi và nhận.

5 Tham khảo

Tài liệu

[1] Cryptography and Network Security, Principles and Practice, William Stallings, Pearson, 7^{th} Edition, 2017

[2] Understanding Cryptography - A Textbook for Students and Practitioners, Christof Paar, Jan Pelzl - 27/11/2009

[3] The Code Book/Mat Ma: The Science Of Secrecy From Ancient Egypt To Quantum Cryptography/Tu Co Dien Den Luong Tu, Simon Singh, (Vietnamese Edition) – 1/7/2008

[4] Thuật toán mã hóa DES
(`https://viblo.asia/p/thuat-toan-ma-hoa-des-JQVGVzEwGyd`)

[5] Tìm hiểu thuật toán mã hóa khóa đối xứng AES
(`https://viblo.asia/p/tim-hieu-thuat-toan-ma-hoa-khoa-doi-xung-aes-gAm5yxOqldb`)

[6] Hệ mã hóa RSA và chữ ký số
(`https://viblo.asia/p/he-ma-hoa-rsa-va-chu-ky-so-6J3ZgkgMZmB`)

[7] Wikipedia (`https://www.wikipedia.org/` - tham khảo ngày 30/3/2018)

Phụ lục 1:
Bảng đánh giá

Mã số sinh viên	Họ và tên	Nhiệm vụ, công việc đã thực hiện	Phần trăm tham gia
1511093	Đỗ Đức Hoài	+ Tìm hiểu về cơ sở lý thuyết, độ an toàn của giải thuật DES, và các giải thuật Hash. + Ứng dụng giải thuật DES, và các giải thuật Hash để viết chương trình. + Trình bày báo cáo nhóm.	33%
1511503	Trần Công Khanh	+ Tìm hiểu về cơ sở lý thuyết, độ an toàn của giải thuật AES. + Ứng dụng giải thuật AES để viết chương trình. + Thiết kế giao diện UI cho chương trình.	33%
1511466	Hoàng Hồng Khang	+ Tìm hiểu về cơ sở lý thuyết, độ an toàn của giải thuật RSA. + Ứng dụng giải thuật RSA để viết chương trình. + Thiết kế giao diện UI cho chương trình.	33%

Phụ lục 2:
Hướng dẫn sử dụng chương trình

Quá trình mã hóa:

- Để thực hiện quá trình mã hóa một tập tin hoặc một thư mục, ta tiến hành mở chương trình lên, và chọn một trong ba giải thuật mã hóa là **DES**, **AES**, và **RSA** có trên màn hình.

- Tiếp đến sau khi nhấp vào một trong ba giải thuật mã hóa, chương trình chuyển sang một cửa sổ mới tương ứng với giải thuật mã hóa đã chọn. Tại đây, ta tiến hành điền các thông số đầu vào cho chương trình.

- **Key** là chìa khóa mã hóa tập tin, ta có thể nhập vào ô trống chìa khóa mong muốn, hoặc nhấn **Generate Key** để chương trình sinh khóa tự động.

- **Input** là tập tin hoặc thư mục cần mã hóa, ta có thể nhấn **Open file** và chỉ đường dẫn đến tập tin mong muốn mã hóa (tất cả các loại tập tin trừ tập tin có định dạng ".encrypted" - nếu sai định dạng sẽ xuất hiện thông báo). Trong trường hợp mã hóa thư mục ta chọn **Open folder** và chỉ đường dẫn đến thư mục mong muốn mã hóa, tại đây ta có hai lựa chọn hoặc chỉ mã hóa các tập tin trong thư mục này (**Only This Folder**), hoặc mã hóa các tập tin trong thư mục này và các tập tin trong thư mục con của nó (**The Folder And Its Sub-folder**).

- Đối với một số giải thuật ta còn có thêm một thông số là **Mode** thể hiện các kiểu thao tác của giải thuật đó. Ta có ba kiểu thao tác là: sách mật mã điện tử (Electronic Codebook Book - **ECB**), dây chuyền mã khối (Cipher Block Chaining - **CBC**), mã phản hồi ngược (Cipher FeedBack - **CFB**).

- Sau khi nhập đủ thông số yêu cầu, ta nhấn **Encrypt** để tiến hành mã hóa tập tin hoặc thư mục.

- Chờ thanh chỉ trạng thái phần trăm tiến đến 100% và xuất hiện ra bảng thông báo đã mã hóa thành công và hiển thị tổng thời gian thực thi tức là quá trình mã hóa đã kết thúc thành công.

- Để đi đến thư mục chứa tập tin đã được mã hóa ta nhấp vào **Open** ở mục **Output** trên màn hình giao diện của giải thuật mã hóa tương ứng.

Quá trình giải mã:

- Để tiến hành giải mã một tập tin hoặc thư mục, ta mở chương trình lên, và chọn một trong ba giải thuật để giải mã là **DES**, **AES**, và **RSA** tương ứng với giải thuật mà trước đó đã dùng để mã hóa tập tin hoặc thư mục này.

- Tiếp đến sau khi nhấp vào một trong ba giải thuật trên, chương trình chuyển sang một cửa sổ mới tương ứng với giải thuật đã chọn. Tại đây, ta tiến hành điền các thông số đầu vào cho chương trình.

- **Key** là chìa khóa giải mã tập tin, ta phải nhập chính xác chìa khóa mà trước đó ta dùng để mã hóa.
- **Input** là tập tin hoặc thư mục cần giải mã, ta có thể nhấn **Open file** và chỉ đường dẫn đến tập tin mong muốn giải mã (chỉ những tập tin có định dạng ".encrypted" - nếu sai định dạng sẽ xuất hiện thông báo). Trong trường hợp giải mã thư mục ta chọn **Open folder** và chỉ đường dẫn đến thư mục mong muốn giải mã, tại đây ta có hai lựa chọn hoặc chỉ giải mã các tập tin trong thư mục này (**Only This Folder**), hoặc giải mã các tập tin trong thư mục này và các tập tin trong thư mục con của nó (**The Folder And Its Sub-folder**).
- Đối với một số giải thuật ta còn có thêm một thông số là **Mode** thể hiện các kiểu thao tác của giải thuật đó. Ta chọn chính xác kiểu thao tác mà trước đó ta đã dùng để mã hóa.
- Sau khi nhập đủ thông số yêu cầu, ta nhấn **Decrypt** để tiến hành giải mã tập tin hoặc thư mục.
- Chờ thanh chỉ trạng thái phần trăm tiến đến 100% và xuất hiện ra bảng thông báo đã giải mã thành công và hiển thị tổng thời gian thực thi tức là quá trình giải mã đã kết thúc thành công. Trong trường hợp một hoặc nhiều các thông số trên có sự sai lệch với các thông số đã nhập lúc mã hóa, thông báo lỗi sẽ xuất hiện trên màn hình.
- Để đi đến thư mục chứa tập tin đã được giải mã ta nhấp vào **Open** ở mục **Output** trên màn hình giao diện của giải thuật giải mã tương ứng.

Quá trình sinh hoặc so sánh mã Hash:

- Để tiến hành sinh hoặc so sánh mã Hash của một tập tin hoặc văn bản, ta mở chương trình lên, và chọn **Hash Algorithms**.
- **Input** chính là tập tin hoặc văn bản ta cần sinh mã Hash hoặc so sánh mã Hash với một mã Hash khác. Ta có thể chọn **File**, nhấn **Open file** và chỉ đường dẫn đến tập tin cần sinh hoặc so sánh mã Hash, hoặc chọn **Text** và nhập văn bản vào ô trống.
- **Hash Algorithms** chỉ ra các giải thuật Hash mà chương trình cung cấp gồm 6 giải thuật là: **MD5**, **SHA-1**, **SHA-256**, **SHA-384**, **SHA-512**, và **RIPEMD-160**. Người dùng chọn một trong sáu giải thuật trên.
- Tại mục **Compare with** người dùng có thể chọn **Calculate** để chương trình sinh ra mã của tập tin hoặc văn bản đó, hoặc chọn **Compare With** để so sánh với một mã Hash khác được người dùng nhập bên dưới.
- Sau khi nhập đủ thông số yêu cầu, ta nhấn **Encrypt**, chương trình sẽ sinh ra mã Hash và hiển thị ở mục **Output**. Nếu ta chọn so sánh hai mã Hash, chương trình sẽ xuất ra thông báo hai mã Hash có khớp hay không.