ĐẠI HỌC THÁI NGUYÊN
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

ĐẶNG THÀNH CÔNG

NGHIÊN CỨU KỸ THUẬT RAINBOW- CRACK THÁM KHÓA MÃ RC4 VÀ ỨNG DỤNG

Chuyên ngành: Khoa học máy tính
Mã số: 60 48 01

LUẬN VĂN THẠC SĨ KHOA HỌC MÁY TÍNH

NGƯỜI HƯỚNG DẪN KHOA HỌC
TS. NGUYỄN NGỌC CƯƠNG

Thái Nguyên, năm 2015

LỜI CAM ĐOAN

Tôi cam đoan đây là công trình nghiên cứu của riêng tôi.

Các số liệu, kết quả nêu trong luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Qua đây em xin chân thành cảm ơn toàn thể các thầy cô trong khoa đào tạo sau đại học trường Đại học Công nghệ Thông tin và Truyền thông và đặc biệt là Thầy TS. Nguyễn Ngọc Cương, đã tạo điều kiện thuận lợi và hướng dẫn em để hoàn thành luận văn này.

MỤC LỤC

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH ẢNH

LỜI MỞ ĐẦU

1.Tính cấp thiết của đề tài

RC4 là tên của thuật toán mã hóa được sử dụng trong WEP, MS-OFFICE... Một thuật toán mã hóa là một tập hợp các hoạt động mà chúng ta sử dụng để biến đổi văn bản chưa mã hóa thành mật mã. Nó sẽ hữu ích, trừ khi có một thuật toán giải mã tương ứng. Trong trường hợp của RC4, cùng một thuật toán được sử dụng để mã hóa và giải mã. Giá trị của một thuật toán mã hóa là ở khả năng bảo mật cao và dễ dàng trong sử dụng. Sức mạnh của một thuật toán được đo bằng độ khó để crack các bản mã được mã hóa bằng thuật toán đó. Chắc chắn là có các phương pháp mạnh hơn RC4. Tuy nhiên, RC4 là khá đơn giản để thực hiện và được coi là rất mạnh, nếu được sử dụng đúng cách.

Kỹ thuật đánh đổi bộ nhớ-thời gian (Time Memory Trade –Off) còn có tên gọi khác là đánh đổi không gian-thời gian dùng để chỉ việc sử dụng bộ nhớ lưu trữ dữ liệu tính toán trước với mục đích giảm thời gian tính toán đối với một thao tác cụ thể. Đây là kỹ thuật được áp dụng trong một số bài toán có thể chia các thao tác tính toán thành hai phần: tính toán trước và tra cứu dữ liệu đã chuẩn bị trước. Nếu tính toán và lưu trữ trước được càng nhiều thì thời gian giải một bài toán cụ thể sẽ chỉ tương đương với thời gian tra cứu.

Các phương tiện lưu trữ máy tính ngày một lớn hơn làm cho khả năng ứng dụng kỹ thuật TMTO ngày càng hiện thực. Đã có nhiều ứng dụng sử dụng kỹ thuật TMTO để giải quyết các vấn đề về tốc độ và bộ nhớ lưu trữ. Chẳng hạn, các bài toán liên quan đến tra cứu bảng dữ liệu, bài toán lưu trữ dữ liệu dạng nén, bài toán lưu trữ thuật toán, lưu trữ kết quả hình ảnh trong hiển thị công thức toán học trên trang HTML,...

Kỹ thuật mật mã cần làm việc với một không gian dữ liệu lớn (không gian khóa). Tuy nhiên, ở một số chế độ làm việc, có thể tổ chức tính toán sẵn các bản mã có thể của một bản rõ để thành lập một từ điển tra cứu cho phép mã hóa và giải mã nhanh. Mã thám có thể lợi dụng tính chất này để tấn công mật mã (kiểu tấn công Brute-Force) nếu có đủ bộ nhớ.

Đề tài luận văn này lựa chọn mật mã RC4 với độ dài khoá 40 bit để nghiên cứu. Đây là dạng RC4 ứng dụng trong nhiều phần mềm. Độ dài khóa là tương

đương với một số kết quả nghiên cứu ứng dụng kỹ thuật TMTO đã công bố đối với một số thuật toán mật mã khác. Kết quả nghiên cứu của đề tài sẽ là hướng mở cho nghiên cứu ứng dụng tấn công mật khẩu bảo vệ tệp văn bản soạn trên một số phần mềm xử lý văn bản. Đồng thời là kinh nghiệm cho mã thám viên đối với kỹ thuật TMTO có thể áp dụng cho tấn công nhiều dạng mật mã ứng dụng khác.

Đề tài luận văn tìm hiểu lý thuyết cơ bản về kỹ thuật TMTO, những vấn đề cần quan tâm trong ứng dụng, những cải tiến đã công bố gần đây cho kỹ thuật TMTO. Đề tài tiến hành áp dụng kỹ thuật TMTO vào thực tế tấn công một giải thuật mật mã cụ thể có khả năng triển khai ứng dụng thực tế.

2. Mục tiêu nghiên cứu:

- Nghiên cứu kỹ thuật Time Memory Trade-Off (TMTO) đánh đổi không gian lưu trữ với thời gian tấn công mật mã. Nghiên cứu về các cải tiến "Điểm phân biệt" và "Bảng cầu vồng" đã được công bố của kỹ thuật này.

- Sử dụng kỹ thuật TMTO được Oechslin áp dụng với cải tiến "Rainbow Crack" tấn công thám khoá mã RC4 ứng dụng trong phần mềm soạn thảo văn bản MS-WORD phiên bản 2007 của MicroSoft.

3. Nội dung nghiên cứu:

Luận văn được trình bày trong 3 chương, có phần mở đầu, phần kết luận, phần mục lục, phần tài liệu tham khảo. Các nội dung cơ bản của luận văn được trình bày theo cấu trúc như sau:

Chương 1: Mật mã RC4 và kỹ thuật Time-Memory Trade-Off áp dụng trong bài toán tấn công mật

Chương 2: Kỹ thuật tấn công Rainbow đối với RC4

Chương 3: Xây dựng chương trình tính toán tham số tấn công Rainbow đối với RC4

Bằng sự cố gắng nỗ lực của bản thân và đặc biệt là sự giúp đỡ tận tình, chu đáo của thầy giáo TS. Nguyễn Ngọc Cương , em đã hoàn thành luận văn đúng thời hạn. Do thời gian làm đồ án có hạn và trình độ còn nhiều hạn chế nên không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự đóng góp ý kiến của các thầy cô cũng như là của các bạn sinh viên để bài luận văn này hoàn thiện hơn nữa.

Chương 1: MẬT MÃ RC4 VÀ KỸ THUẬT TIME-MEMORY TRADE –OFF ÁP DỤNG TRONG BÀI TOÁN TẤN CÔNG MẬT MÃ

Trong chương này là trình bày tập hợp các thông tin cơ sở về kỹ thuật TMTO; các cải tiến "điểm phân biệt" của Rivest và "bảng cầu vồng" của Oechslin. Nội dung chương làm rõ phương thức chia không gian tìm kiếm thành các bộ phận và tổ chức lưu trữ hiệu quả từng bộ phận không gian tìm kiếm. Đặc biệt là phương pháp tổ chức các "bảng cầu vồng" của Oechslin, phương pháp được ứng dụng hiệu quả trong phần mềm OPH-Crack. Thuật toán mật mã RC4 đóng vai trò trung tâm trong lược đồ xác thực mật khẩu. Bên cạnh đó là những phương pháp thám mã khác cũng đang được áp dụng nhiều trong thực tế.

1.1 Tổng quan về RC4

RC4 là tên của thuật toán mã hóa được sử dụng trong WEP, MS-OFFICE... Một thuật toán mã hóa là một tập hợp các hoạt động mà chúng ta sử dụng để biến đổi văn bản chưa mã hóa thành mật mã. Nó sẽ hữu ích, trừ khi có một thuật toán giải mã tương ứng. Trong trường hợp của RC4, cùng một thuật toán được sử dụng để mã hóa và giải mã. Giá trị của một thuật toán mã hóa là ở khả năng bảo mật cao và dễ dàng trong sử dụng. Sức mạnh của một thuật toán được đo bằng độ khó để crack các bản mã được mã hóa bằng thuật toán đó. Chắc chắn là có các phương pháp mạnh hơn RC4. Tuy nhiên, RC4 là khá đơn giản để thực hiện và được coi là rất mạnh, nếu được sử dụng đúng cách. Thật may mắn là RC4 khá đơn giản để thực hiện và mô tả. Ý tưởng cơ bản mã hóa RC4 là tạo ra một chuỗi các trình tự giả ngẫu nhiên (giả ngẫu nhiên) của các byte được gọi là khóa dòng, sau đó được kết hợp với các dữ liệu bằng cách sử dụng toán tử OR (XOR). Toán tử XOR kết hợp hai byte và tạo ra một byte duy nhất. Nó làm điều này bằng cách so sánh các bit tương ứng trong từng byte. Nếu chúng bằng nhau, kết quả là 0, nếu chúng khác nhau, kết quả là 1. Về mặt lý thuyết, RC4 không phải là một hệ thống mã hóa hoàn toàn an toàn bởi vì nó tạo ra một dòng giả ngẫu nhiên chính, không phải byte thực sự ngẫu nhiên. Nhưng nó đủ chắc chắn an toàn cho các ứng dụng, nếu được áp dụng đúng.

RC4 là mật mã có cỡ của khóa biến đổi do Ron Rivest phát triển vào những năm 1987 cho liên hợp an ninh dữ liệu RSA. Trong bảy năm nó là sở hữu độc

quyền và các chi tiết của thuật toán ta chỉ có được sau khi ký thỏa thuận không tiết lộ bí mật.

Vào tháng 9 năm 1994, một người lạc danh đã gửi mã nguồn qua bưu điện vào danh sách thư tín Cypherpunks, Nó nhanh chóng lan tỏa đến nhóm Usenet và qua Internet đến các site ftp trên thế giới. Liên hiệp an ninh dữ liệu RSA tuyên bố rằng nó vẫn còn là một bí mật thương mại mặc dù nó đã được công bố, nhưng việc này đã quá muộn. Bởi nó đã được thảo luận và phân tích kỹ trên Usenet, đọc phân phát ở các hội nghị và được đưa vào các giáo trình mật mã.

RC4 có địa vị xuất khẩu đặc biệt nếu độ dài khóa của nó là 40 bít hoặc ít hơn. Địa vị xuất khẩu đặc biệt này sẽ dẫn đến việc không có gì để làm đối với độ an toàn của thuật toán, mặc dù liên hợp an ninh dữ liệu RSA đã nói bóng gió trong nhiều năm rằng vẫn có. Tên thuật toán này ðýợc thýõng mại hóa do ðó bất kỳ ngýời nào viết mã riêng của mình ðều phải gọi nó bằng một cái tên khác. Các tài liệu bên trong khác của liên hợp an ninh dữ liệu RSA vẫn chưa được công bố.

RC4 là một phần mềm trong các sản phẩm mật mã thương mại, bao gồm Lotus Notes, Apple Computer's AOCE và ORACLE Security SQL. Nó là một bộ phận của bản chỉ dẫn kỹ thuật Cellular Digital Packed Data.

RC4 là một họ các thuật toán phụ thuộc vào các tham số nguyên dương, mà điển hình là trường hợp n= 8. Ở thời điểm t, trạng thái bên trong của RC4 gồm bảng

$S_1=(S_1(l))_{l=0}^{2^n-1}$ có từ n-bít và 2 con trỏ n-bít là it và jt. Do đó cỡ bộ nhớ trong la $M=n2^n+2n$(bít). Gọi Zt là từ ra n-bít của RC4 ở thời điểm t. Bít có nghĩa thấp nhất của một từ là bít ở bên trái nhất của nó.

1.2. Các kỹ thuật thám mã

1.2.1 WEP

WEP (Wired Equivalent Privacy) là một thuật toán nhằm bảo vệ sự trao đổi thông tin chống lại nghe trộm, chống lại những kết nối mạng không được cho phép cũng như chống lại việc thay đổi hoặc làm nhiễu thông tin truyền. WEP sử dụng stream cipher RC4 cùng với một mã 40 bit và một số ngẫu nhiên 24 bit (initialization vector - IV) để mã hóa thông tin. Thông tin mã hóa và IV sẽ được gửi đến người nhận. Người nhận sẽ giải mã thông tin dựa vào khóa WEP đã biết trước.

1.2.2 Tấn công chọn bản mã

Tấn công chọn bản mã (chosen ciphertext): người thám mã tạm thời có quyền truy xuất tới Bộ giải mã, do đó anh ta có khả năng chọn bản mã và xây dựng lại bản tin rõ tương ứng.

Trong mọi trường hợp, mục đích là tìm ra khóa mã được sử dụng. Kiểu tấn công chọn bản mã được thực hiện với hệ mật mã khóa công khai mà chúng ta sẽ xem xét trong chương kế tiếp. Trong phần này chúng ta chỉ thảo luận về kiểu tấn công được xem là "yếu nhất" - Tấn công chỉ biết bản mã.

Nhiều kỹ thuật thám mã sử dụng đặc điểm thống kê của tiếng Anh, trong đó dựa vào tần suất xuất hiện của 26 chữ cái trong văn bản thông thường để tiến hành phân tích mã. Becker và Piper đã chia 26 chữ cái thành năm nhóm và chỉ ra xác suất của mỗi nhóm như sau:

E, có xác suất khoảng 0.120

T, A, O, I, N, S, H, R, mỗi chữ cái có xác xuất nằm trong khoảng từ 0.06 đến 0.09

D, L, mỗi chữ cái có xác xuất xấp xỉ 0.04

C, U, M, W, F, G, Y, P, B, mỗi chữ cái có xác xuất nằm trong khoảng từ 0.015 đến 0.023

V, K, J, X, Q, Z, mỗi chữ cái có xác xuất nhỏ hơn 0.01

Ngoài ra, tần suất xuất hiện của dãy hai hay ba chữ cái liên tiếp được sắp theo thứ tự giảm dần như sau [11]: TH, HE, IN, ER … THE, ING, AND, HER…

1.2.3 Thám mã tích cực:

Thám mã tích cực là việc thám mã sau đó tìm cách làm sai lạc các dữ liệu truyền, nhận hoặc các dữ liệu lưu trữ phục vụ mục đích của người thám mã.

Thám mã thụ động:

Thám mã thụ động là việc thám mã để có được thông tin về bản tin rõ phục vụ mục đích của người thám mã.

1.2.4 Thám mã Affine

Giả sử Trudy đã lấy được bản mã sau đây:

FMXVEDKAPHFERBNDKRXRSREFMORUDSDKDVSHVUFEDKAPRKDLYEVLRHHRH.

Trudy thống kê tần suất xuất hiện của 26 chữ cái như trong bảng sau:

Chữ cái	Tần suất	Chữ cái	Tần suất
A	2	N	1
B	1	O	1
C	0	P	3
D	6	Q	0
E	5	R	8
F	4	S	3
G	0	T	0
H	5	U	2
I	0	V	4
J	0	W	0
K	5	X	2
L	2	Y	1
M	2	Z	0

Bảng 1.1: Bảng Thám mã Affine

Chỉ có 57 chữ cái trong bản mã nhưng phương pháp này tỏ ra hiệu quả để thám mã Affine. Ta thấy tần suất xuất hiện các chữ cái theo thứ tự là: R(8), D(6), E, H, K(5) và F, S, V(4). Vì vậy dự đoán đầu tiên của ta có thể là: R là mã của e, D là mã của t. Theo đó,eK(4)=17 và eK(19)=3. Mà eK(x)=ax+b với a, b là các biến. Để tìm K=(a, b) ta giải hệ phương trình:

$$4a+b=17$$

$$19a+b=3$$

Suy ra, a = 6, b=19. Đây không phải là khóa vì gcd(a, 26) = 2 > 1. Ta lại tiếp tục phỏng đoán: R là mã của e, E là mã của t. Ta nhận được a = 13, chưa thỏa mãn. Tiếp tục với H, ta có a=8. Cuối cùng, với K ta tìm được K= (3, 5).

Sử dụng khóa mã này ta có được bản tin rõ như sau:

Algorithmsrequiregeneraldefinitionsofarithmeticprocesses

1.2.5 Thám mã Vigenere

Để thám mã Vigenere, trước hết cần xác định độ dài từ khóa, ký hiệu làm. Sau đó mới xác định từ khóa. Có hai kỹ thuật để xác định độ dài từ khóa đó là phương pháp Kasiski và phương pháp chỉ số trùng hợp (index of coincidence).

Phương pháp Kasiski được đưa ra bởi Friedrich Kasiski năm 1863. Phương pháp này làm việc như sau:

Tìm trên bản mã các cặp xâu kí tự giống nhau có độ dài ít nhất là 3, ghi lại khoảng cách giữa vị trí chữ cái đầu tiên trong các xâu và xâu đầu tiên. Giả sử nhận được d 1 , d 2 ... Tiếp theo ta phỏng đoán m là số sao cho ước số chung lớn nhất của các d i chia hết cho m.

Ví dụ:

Plaintext: conghoa|danchun|handant|runghoa|sapsuat|hanghoa

Keyword: abcdefg

Ciphertext: CPPJLTG DBPFLZT HBPGESZ RVPJLTG SBRVYFZ HBPJLTG

Vị trí xuất hiện của dãy PJL lần lượt là: *3, 24, 38.* Do vậy, dãy *d1, d2 ... là 21, 35; gcd(d1, d2 ...) = 7*

Phương pháp chỉ số trùng hợp sẽ cho biết các bằng chứng để nhận được giá trị m. Phương pháp này được đưa ra bởi Wolfe Friedman năm 1920 như sau:

Giả sử x=x 1 x 2... x n là xâu có n ký tự. Chỉ số trùng hợp của x, ký hiệu là I c (x), được định nghĩa là xác suất mà hai phần tử ngẫu nhiên của x là giống nhau. Giả sử chúng ta ký hiệu tần suất của A, B, C, ..., Z trong x lần lượt là f 0 , f 1, ..., f 25 . Chúng ta có thể chọn hai phần tử của x theo

(2^n) = n!/(2!(n-2)!) cách. Với mỗi $0 \le i \le 25$ *, có (2^{fi}) cách chọn các phần tử là i. Vì vậy, chúng ta có công thức:*

$$I\ c\ (X) = \frac{\sum_{i=0}^{25} fi(fi-1)}{n(n-1)}$$

Bây giờ, giả sử x là xâu văn bản tiếng Anh. Ta có $I_c(x) \sum_{i=0}^{25} P_i^2 = 0.065$

Ví dụ:

Cho bản mã trong hệ mật mã Vigenere

CHREEV	**OAHMAE**	**RATBIT**	**XXWTNX**	**BEEOPH**	**BSBQMQ**	**EQERBW**
RVXUOA	XXAOSXX	…				
LXFPSK						
VRVPRT						…CHR
ZBWELE						
AMRVLO	…		…	WCHRQH	…	
PEEWEV	KAKOE	WADREM	XMTBHHC	HRTKDN	VRCHR	CLQOHP
WQAIIW	XNRMGW	OIIFKBE				

Bảng1.2: Bản mã trong hệ mật mã Vigenere

- Theo phương pháp Kasiski, đầu tiên xâu CHR xuất hiện ở 4 vị trí trong bản mã, lần lượt là: 1, 166, 236 và 286. Khoảng cách giữa các xâu là 165, 235 và 285. Ước số chung lớn nhất của các số này là 5. Vậy ta có m =5.
- Theo phương pháp chỉ số trùng hợp, với m=1 thì chỉ số trùng hợp là $I_c(x)$ = 0.045; m=2, $I_c(x)$=0.046 và 0.041; m=3, $I_c(x)$=0.043, 0.050, 0.047; m=4, $I_c(x)$=0.042, 0.039, 0.046, 0.040; m=5, $I_c(x)$=0.063, 0.068, 0.069, 0.072; Ta dừng và nhận được m = 5.

Để xác định khóa mã, ta sử dụng phương pháp thống kê sau đây:

Giả sử x=x 1 x 2… x n và y=y 1 y 2… y n' là hai xâu có n và n' ký tự . Chỉ số trùng hợp tương quan của x và y, ký hiệu là MI c (x,y), được định nghĩa là xác suất mà một phần tử ngẫu nhiên của x bằng một phần tử ngẫu nhiên của y. Nếu chúng ta ký hiệu tần suất của A, B, C, …, Z trong x và y lần lượt là f 0 , f 1 , …, f 25 . và f' 0 , f' 1 , …, f' 25 . Thì:

$$MI\ c\ (x,y) = \frac{\sum_{i=0}^{25} fif'i}{nn'}$$

Bây giờ, giả sử x,y là xâu văn bản tiếng Anh. Ta có $MI_c(x_i,y_j)$ 0.065

Ví dụ:

Giả sử m=5 như ta đã thực hiện ở trên. Theo phương pháp thống kê [11] ta tìm được khóa mã là: JANET. Vậy bản tin rõ sẽ là: *the almond tree was in ...*

1.2.6 Các tính năng trong RainbowCrack:

- Tích hợp công cụ Full time-memory tradeoff, bao gồm các xử lý trên rainbow table , sắp xếp, chuyển đổi và tra cứu
- Hỗ trợ rainbow table của bất kỳ thuật toán băm
- Hỗ trợ rainbow table của bất kỳ ký tự
- Hỗ trợ rainbow table trong định dạng tập tin nguyên (rt.) Và định dạng file nén (. Rtc)
- Tính toán về hỗ trợ xử lý đa lõi
- Hỗ trợ tính toán trên GPU (thông qua công nghệ NVIDIA CUDA)
- Hỗ trợ tính toán trên đa GPU (thông qua công nghệ NVIDIA CUDA)
- Thống nhất định dạng tập tin rainbow table trên tất cả các hệ điều hành hỗ trợ
- Giao diện người dùng dòng lệnh

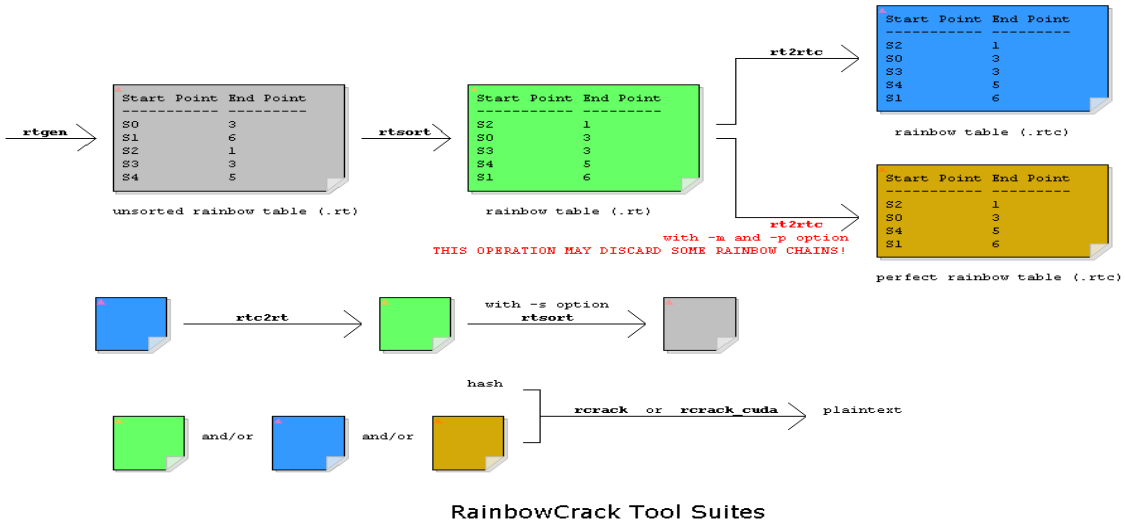
➢ Giao diện người dùng đồ họa (Windows) ,chạy trên Windows XP 32-bit, Windows Vista 32-bit và Windows 7 32-bit.

Môi trường Windows	Môi Trường Linux	Giao diện đồ họa	Giao diện Command Line	Miễn phí	Có phí	Source Code Download
✓	✓	✓	✓	✓		✓

Hình 1.1: Các tính năng trong RainbowCrack

1.2.7 Các công cụ và mối quan hệ giữa chúng trong RainbowCrack.

Phần mềm RainbowCrack bao gồm năm công cụ: *rtgen, rtsort, rt2rtc, rtc2rt và rcrack*. Hình dưới đây giải thích mối quan hệ giữa các công cụ.



Hình1.2: Các công cụ của Phần mềm RainbowCrack

1.3 Xây dựng RainbowCrack:

- ***RainbowCrack technique***is the implementation of Philippe Oechslin's faster time-memory trade-off technique.
- ***RainbowCrack*** làm việc dựa trên việc tính trước tất cả password có thể có.
- Sau khi quá trình time-consuming này hoàn thành,password và thông tin khác được mã hóa của chúng được chứa trong một file gọi là rainbow table
- Một password được mã hóa có thể được so sánh nhanh với giá trị chứa trong table và bẻ khóa trong một vài giây.
- ***Rainbow tables*** nhỏ sẽ làm giảm nhu cầu lưu trữ và tăng hiệu suất của công cụ rcrack.

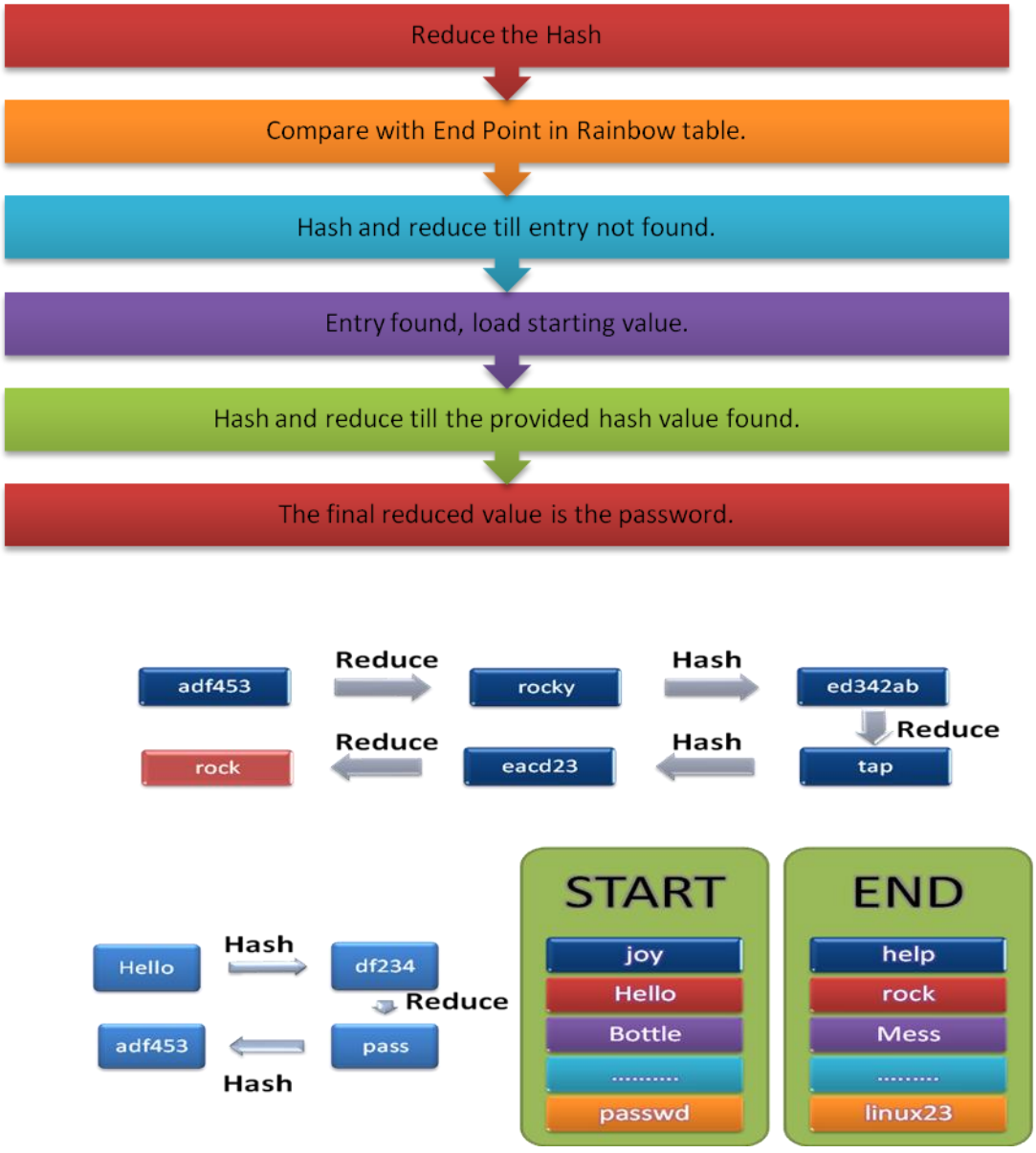
- ***Ophcrack*** là công cụ bẻ khóa password áp dụng kỹ thuật rainbow table.
- ***RainbowCrack*** bao gồm ba công cụ được sử dụng trình tự để thực hiện những điều sau.

Bước 1: Sử dụng chương trình rtgen để tạo ra các rainbow tables.

Bước 2: chương trình rtsort sử dụng để sắp xếp bảng cầu vồng được tạo ra bởi rtgen.

Bước 3: Sử dụng chương trình rcrack để tra cứu rainbow tables được sắp xếp theo rtsort.

RainbowCrack Algorithm



Hình 1.3: Giao diện của Rainbow Crack

Bước 1: Sử dụng chương trình rtgen để tạo ra các bảng cầu vồng

Cú pháp của dòng lệnh là:

_rtgen hash_algorithm charset plaintext_len_min plaintext_len_max table_index chain_len chain_num part_index.

Giải thích các thông số :

Tham số	Ýnghĩa
hash_algorithm	Các thuật toán băm (LM, NTLM, md5…) được sử dụng trong bảng cầu vồng.
Charset	Các ký tự của tất cả các plaintexts trong bảng cầu vồng. Tất cả các ký tự có thể được định nghĩa trong file charset.txt.
plaintext_len_min plaintext_len_max	Hai tham số này xác định độ dài của tất cả các plaintexts trong bảng cầu vồng. Nếu ký tự là số, plaintext_len_min là 1, và plaintext_len_max là 5. Sau đó, plaintext là "12345" có thể bao gồm trong bảng, nhưng "123456" sẽ không được tính.
table_index chain_len chain_len chain_num chain_num part_index part_index	table_index là các "chức năng làm giảm" được sử dụng trong bảng cầu vồng. chain_len là độ dài của mỗi "chuỗi cầu vồng" trong bảng cầu vồng. Một "rainbow chain" có kích thước 16 byte là đơn vị nhỏ nhất trong một rainbow table. Một rainbow table có chứa rất nhiều các chuỗi cầu vồng chain_num là số lượng các chuỗi cầu vồng trong rainbow table . part_index xác định "điểm bắt đầu" trong mỗi chuỗi cầu vồng được tạo ra như thế nào. Nó phải là một số (hoặc bắt đầu bằng số một) trong RainbowCrack .

Cấu hình được đưa ra dưới đây sẽ sẵn sàng cho công việc cần tiến hành

hash_algorithm	**LM, NTLM hoặc md5**
Charset	alpha-số = [ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789] hoặc loweralpha-số = [abcdefghijklmnopqrstuvwxyz0123456789]
plaintext_len_min	1
plaintext_len_max	7
chain_len	3800
chain_num	33554432
Key space	36 ^ 1 + 36 ^ 2 + 36 ^ 3 + 36 ^ 4 + 36 ^ 5 + 36 ^ 6 + 36 ^ 7 = 80603140212 Key space là số plaintexts có thể cho các plaintext_len_min, charset và plaintext_len_max chọn.
kích cỡ bảng	3 GB
tỷ lệ thành công	0.999 Thuật toán The time-memory tradeoff là một thuật toán xác suất. Dù các tham số được lựa chọn, luôn luôn có xác suất mà các chữ thô trong bộ ký tự lựa chọn và phạm vi chiều dài plaintext là không được. Tỷ lệ thành công là 99,9% với các tham số được sử dụng trong ví dụ này.
bảng hệ lệnh	Các lệnh rtgen được sử dụng để tạo ra các bảng cầu vồng là: rtgen md5 loweralpha-numeric 1 7 0 3800 33554432 0 rtgen md5 loweralpha-số 1 7 0 3800 33554432 0 rtgen md5 loweralpha-numeric 1 7 1 3800 33554432 0 rtgen md5 loweralpha-số 1 7 1 3800 33554432 0 rtgen md5 loweralpha-numeric 1 7 2 3800 33554432 0 rtgen md5 loweralpha-số 1 7 2 3800 33554432 0 rtgen md5 loweralpha-numeric 1 7 3 3800 33554432 0 rtgen md5 loweralpha-số 1 7 3 3800 33554432 0 rtgen md5 loweralpha-numeric 1 7 4 3800 33554432 0 rtgen md5 loweralpha-số 1 7 4 3800 33554432 0 rtgen md5 loweralpha-numeric 1 7 5 3800 33554432 0 rtgen md5

	loweralpha-số 1 7 5 3800 33554432 0 Nếu table NTLM hoặc table LM là mong muốn, thay thế "md5" trong câu lệnh trên với "NTLM" hoặc "LM". Nếu bộ ký tự chữ-số là mong muốn, thay thế "loweralpha-số" trong câu lệnh trên với "alpha-số". Nếu LM table sẽ được tạo ra, nên CONFIRM bộ ký tự là chữ-số thay vì loweralpha-số. Các thuật toán lm không bao giờ sử dụng chữ thường là plaintext .

Để tạo ra bảng rainbow. Vào thư mục RainbowCrack, và thực hiện lệnh sau:

rtgen md5 loweralpha-numeric 1 7 0 3800 33554432 0 rtgen md5 loweralpha-số 1 7 0 3800 33554432 0

Khi lệnh được hoàn tất, một file có tên "md5_loweralpha-số # 1-7_0_3800x33554432_0.rt" có kích thước 512 MB sẽ được đặt đúng chỗ. Tên file là tất cả các tham số dòng lệnh được kết nối, với phần mở rộng "rt". Chương trình rcrack cần mẩu thông tin này để biết các tham số của bảng cầu vồng.Vì vậy, đừng đổi tên file.

Các table còn lại có thể được tạo ra với các lệnh:

rtgen md5 loweralpha-numeric 1 7 1 3800 33554432 0 rtgen md5 loweralpha-số 1 7 1 3800 33554432 0

rtgen md5 loweralpha-numeric 1 7 2 3800 33554432 0 rtgen md5 loweralpha-số 1 7 2 3800 33554432 0

rtgen md5 loweralpha-numeric 1 7 3 3800 33554432 0 rtgen md5 loweralpha-số 1 7 3 3800 33554432 0

rtgen md5 loweralpha-numeric 1 7 4 3800 33554432 0 rtgen md5 loweralpha-số 1 7 4 3800 33554432 0

rtgen md5 loweralpha-numeric 1 7 5 3800 33554432 0 rtgen md5 loweralpha-số 1 7 5 3800 33554432 0

Cuối cùng, những file này được tạo ra:

md5_loweralpha-numeric#1-7_0_3800x33554432_0.rt 512MB md5_loweralpha-số # 1-7_0_3800x33554432_0.rt 512MB

md5_loweralpha-numeric#1-7_1_3800x33554432_0.rt 512MB md5_loweralpha-số # 1-7_1_3800x33554432_0.rt 512MB

md5_loweralpha-numeric#1-7_2_3800x33554432_0.rt 512MB md5_loweralpha-số #

1-7_2_3800x33554432_0.rt 512MB
md5_loweralpha-numeric#1-7_3_3800x33554432_0.rt 512MB md5_loweralpha-số # 1-7_3_3800x33554432_0.rt 512MB
md5_loweralpha-numeric#1-7_4_3800x33554432_0.rt 512MB md5_loweralpha-số # 1-7_4_3800x33554432_0.rt 512MB
md5_loweralpha-numeric#1-7_5_3800x33554432_0.rt 512MB md5_loweralpha-số # 1-7_5_3800x33554432_0.rt 512MB

Lúc này tiến trình tạo rainbow table hoàn thành.

Bước 2: Chương trình rtsort sử dụng để sắp xếp rainbow tables

Những rainbow tables được tạo ra bởi **rtgen** cần một số xử lý cụ thể để làm cho bảng tra cứu dễ dàng hơn. Chương trình rtsort được sử dụng để sắp xếp các "điểm cuối" của tất cả rainbow chains trong một rainbow table. Sử dụng các lệnh sau đây:

rtsort md5_loweralpha-numeric#1-7_0_3800x33554432_0.rt rtsort md5_loweralpha-số 1-7_0_3800x33554432_0.rt
rtsort md5_loweralpha-numeric#1-7_1_3800x33554432_0.rt rtsort md5_loweralpha-số # 1-7_1_3800x33554432_0.rt
rtsort md5_loweralpha-numeric#1-7_2_3800x33554432_0.rt rtsort md5_loweralpha-số # 1-7_2_3800x33554432_0.rt
rtsort md5_loweralpha-numeric#1-7_3_3800x33554432_0.rt rtsort md5_loweralpha-số # 1-7_3_3800x33554432_0.rt
rtsort md5_loweralpha-numeric#1-7_4_3800x33554432_0.rt rtsort md5_loweralpha-số # 1-7_4_3800x33554432_0.rt
rtsort md5_loweralpha-numeric#1-7_5_3800x33554432_0.rt rtsort md5_loweralpha-số # 1-7_5_3800x33554432_0.rt

Mỗi lệnh ở trên mất khoảng 1 đến 2 phút để hoàn thành. Chương trình rtsort sẽ ghi nhận rainbow table đã được sắp xếp các tập tin gốc.

Bây giờ tiến trình sắp xếp rainbow table hoàn thành.

Bước 3: Sử dụng chương trình rcrack để tra cứu rainbow tables

Chương trình rcrack được sử dụng để tra cứu các bảng cầu vồng. Nó chỉ chấp nhận các rainbow tables đã sắp xếp và đặt trong thư mục **c: \ rt**, để crack băm đơn dòng lệnh sẽ là:

rcrack c:\rt*.rt -h your_hash_comes_here rcrack c: \ rt \ *. rt-h your_hash_comes_here

Để crack nhiều băm, đặt tất cả băm trong một file văn bản với mỗi băm là một dòng. Và sau đó chỉ định tên file trong dòng lệnh rcrack:

rcrack c:\rt*.rt -l hash_list_file rcrack c: \ rt \ *. rt-l hash_list_file

Nếu các bảng cầu vồng tạo ra dùng thuật toán LM , chương trình hỗ trợ đặc biệt rcrack đã cho nó với "-f" chuyển lệnh. Một bãi băm tập tin ở định dạng pwdump được yêu cầu làm đầu vào cho chương trình rcrack. Tệp tin sẽ trông như thế này:

Administrator:500:1c3a2b6d939a1021aad3b435b51404ee:e24106942bf38bcf57a6a4b29016eff6:::Guest:501:a296c9e4267e9ba9aad3b435b51404ee:9d978dda95e5185bbeda9b3ae00f84b4:::

Các tập tin pwdump là đầu ra của pwdump2, pwdump3 hoặc các tiện ích khác. Nó chứa cả **LM**hash và **NTLM** hash .

Để crack hashes LM trong file pwdump, sử dụng lệnh sau đây:

rcrack c:\rt*.rt -f pwdump_file rcrack c: \ rt \ *. rt-f pwdump_file

Các thuật toán băm **LM** chuyển đổi tất cả chữ thường trong plaintext thành chữ hoa, kết quả là tất cả các plaintexts nứt qua băm **LM** không bao giờ có chữ thường, trong khi plaintext thực tế có thể chứa chữ thường. Chương trình rcrack sẽ cố gắng làm điều chỉnh trường hợp với NTLM hashes được lưu trữ trong cùng một tập tin và cho ra bản original plaintext .

1.4 Thuận toán MD5

1.4.1 Giới thiệu thuật to¸n:

MD5 ®-îc ph¸t triÓn vµ kÕ thõa tõ MD4 do vËy vÒ c¬ b¶n MD5 vÉn cã nhiÒu nÐt ®Æc tr-ng t-¬ng tù nh- MD4. Nh-ng thay v× sö dông 4 vßng trong MD4, MD5 ®· sö dông 4 vßng víi mét sè hµm phøc t¹p h¬n.

ThuËt to¸n thùc hiÖn ®Çu vµo lµ mét th«ng b¸o cã ®é dµi bÊt kú vµ x©y dùng mét ®Çu ra lµ mét th«ng b¸o 128 bit rót gän. §Çu vµo xö lý c¸c khèi bit cã ®é dµi 512 bit.

Qu¸ tr×nh x©y dùng th«ng b¸o rót gän thuËt to¸n thùc hiÖn mét sè b-íc sau:

B-íc 1: Më réng vµ g¾n thªm c¸c bit.

Cho th«ng b¸o x ®-îc biÓu diÔn b»ng mét dßng bit, |x| lµ ®é dµi cña x. Ta s¶n sinh M = M[0]M[1].....M[N-1], trong ®ã M[i] lµ dßng bit cã ®é dµi 32 bit, gäi mçi M[i] lµ mét tõ, nh- sau:

Bæ sung 1 vµo x, rải thªm d sè 0 nèi vµo kÕt qu¶ 64 bit, nã biÓu diÔn nhÞ ph©n cña |x| (nÕu cÇn thiÕt cã thÓ rót gän modul 2^{64}) soa cho ®é dµi cña th«ng b¸o míi chia hÕt cho 512. Nh- vËy, |M| chia hÕt cho 32 (v× 512 chia hÕt 32). H¬n n÷a N lµ mét sè chia hÕt cho 16 (v× 512 = 32 x 16).

C«ng thøc chung ®Ó tÝnh d nh- sau:

M = x || 1 || 0^d || 1 , trong ®ã |1| = 64

|M|= |x|+ 1 + d + 64 ≡ 0 mod 512

d ≡ -65 - (|x|mod 512) ≡ 447 - (|x|mod 512). Nh-ng d≥ 0:

§Æt α = (|x|mod 512).

NÕu α> 447 th× d = 447 + 512 - α.

NÕu $\alpha\leq$ 447 th× d = 447-α.

B-íc 2: Më réng ®é dµi

§é dµi th«ng b¸o nguyªn thuû ®-îc biÓu diÔn b»ng mét chuçi 64 bit (tr-íc khi thùc hiÖn viÖc më réng). Th«ng b¸o sÏ ®-îc më réng sao cho cã kÕt qu¶ nh- b-íc mét.

KÕt qu¶ ®Çu tiªn cña hai b-íc ®Çu s¶n sinh mét th«ng b¸o. Th«ng b¸o nµy cã ®é dµi lµ béi cña mét sè nguyªn víi 512 bit ®é dµi. Th«ng b¸o më réng ®-îc biÓu diÔn nh- mét chuçi c¸c khèi 512 bit nh- Y_0 , Y_1 ,....., Y_L , v× vËy tæng ®é dµi cña th«ng b¸o sÏ lµ L x 512 bit. T-¬ng tù, kÕt qu¶ lµ béi mét sè nguyªn cña 16 tõ (mçi tõ cã ®é dµi 32 bit). Trong ®ã M[0 ... N-1] biÓu diÔn c¸c

tõ cña kÕt qu¶ th«ng b¸o, víi N lµ mét sè nguyªn vµ N = L * 16.

B-íc 3: Gi¸ trÞ ban ®Çu (khëi nguån) cña bé ®Öm MD

Bé ®Öm 128 bit ®-îc sö dông ®Ó l-u tr÷ trùc tiÕp vµ kÕt qu¶ cuèi cïng cña hµm hash. Bé ®Öm ®-îc thÓ hiÖn lµ 4 thanh ghi 32 bit (A, B, C, D). C¸c thanh ghi nµy ®-îc nhËn gi¸ trÞ ban ®Çu lµ c¸c gi¸ trÞ thËp lôc ph©n d-íi ®©y:

A = 0x01234567

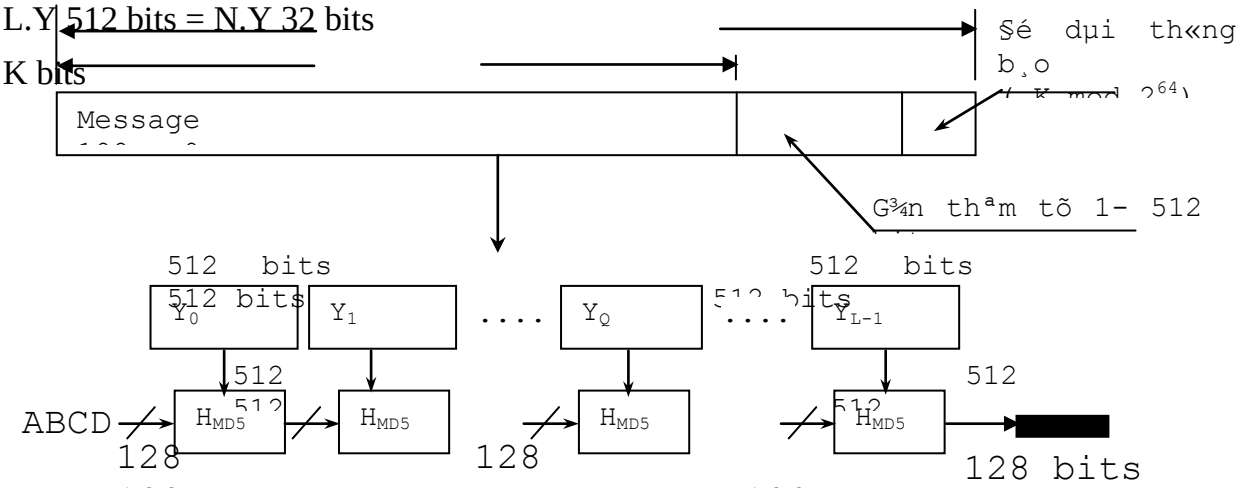
B = 0x89abcdef

C = 0xfedcba98

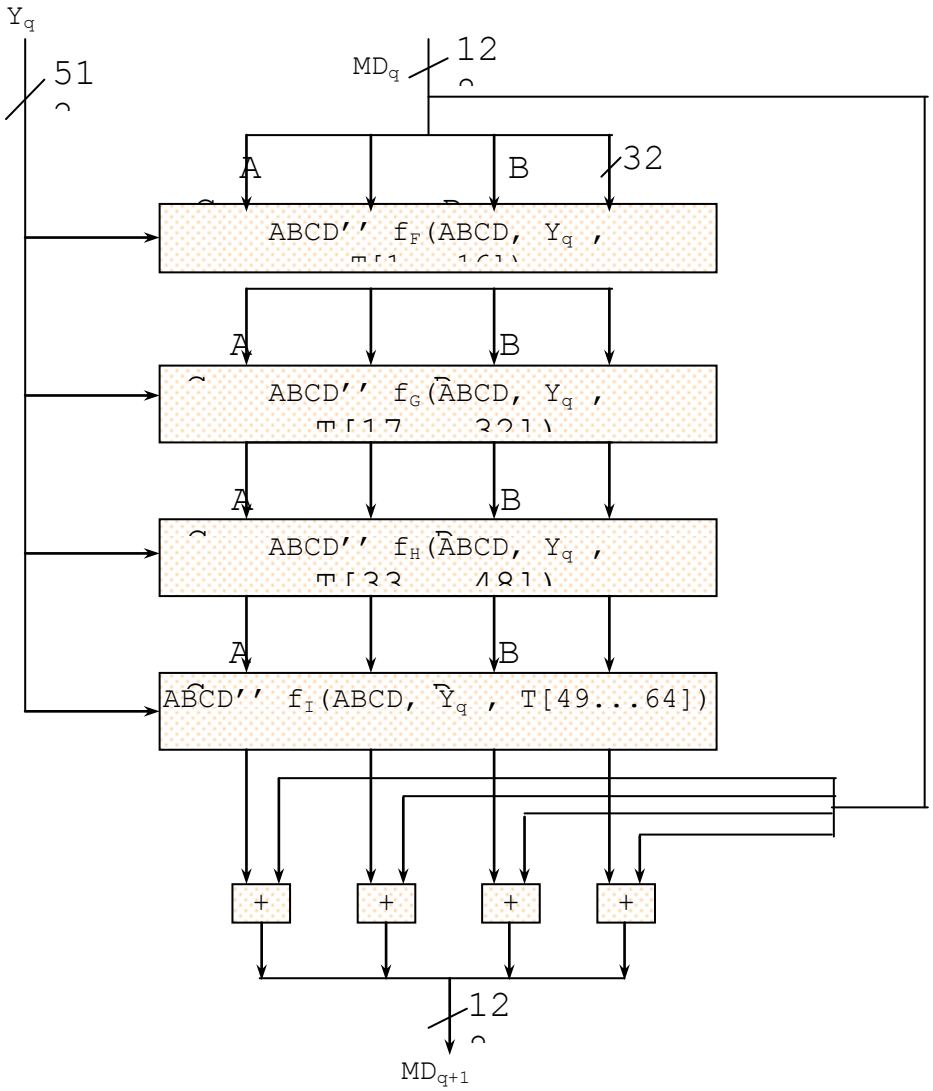
D = 0x76543210

B-íc 4: Xö lý th«ng b¸o trong c¸c khèi 512 bit (16 tõ)

Néi dung chÝnh cña thuËt to¸n lµ phÐp thÝnh module trªn 4 vßng cña qu¸ tr×nh xö lý. Mçi module ®-îc ®Æt mét nh·n H_{MD5} , chóng ®-îc thÓ hiÖn theo h×nh 1. Bèn vßng cã cÊu tróc nh- nhau, nh-ng mçi hµm sö dông c¸c hµm logic nguyªn thñy kh¸c nhau, nh- c¸c hµm F, G, H vµ I ®-îc m« t¶ chi tiÕt trong h×nh 2, bèn vßng lµ c¸c nh·n f_F , f_G , f_H , f_I , c¸c nh·n nµy chØ ra r»ng mçi vßng cã chøc n¨ng cÊu tróc tæng qu¸t lµ nh- nhau, nh-ng f phô thuéc vµo c¸c hµm nguyªn thñy (F, G, H, I).



Hình 1.4: Mô hình tổng quát sản sinh thông báo rút gọn sử dụng MD5



Hình 1.5: Mô hình biểu diễn công việc xử lý các khối đơn 512 bit (H_{MD5})

Chó ý mçi vßng thùc hiÖn nh- ®Çu vµo mét khèi hiÖn t¹i 512 bit ®-îc xö lý (Y_q) vµ bé ®Öm 128 bit cã gi¸ trÞ ABCD vµ cËp nhËt néi dung cña bé ®Öm. Mçi vßng t¹o ra sö dông 1/4 cña mét table 64 phÇn tö T[1...64] ®-îc x©y dùng tõ hµm sine. PhÇn tö thø i cña T lµ T[i] cã gi¸ trÞ b»ng mét phÇn cña sè nguyªn (2^{32} x abs(sin(i))), trong ®ã i tÝnh b»ng radians. Khi abs(sin(i)) lµ mét sè n»m gi÷a 0 vµ 1, th× mçi phÇn tö cña T lµ mét sè nguyªn cã thÓ

®-îc biÓu diÔn b»ng 32 bit nhÞ ph©n. Trong b¶ng x©y dùng mét c¸ch ngÉu nhi¬n mét tËp c¸c mÉu 32 bit, c¸c mÉu bit nµy ®-îc khö ®Òu ®Æn tr¬n mét d÷ liÖu vµo. B¶ng 1 biÓu diÔn c¸c gi¸ trÞ cña T ®-îc x©y dùng tõ hµm sine.

B¶ng 1 d-íi ®©y lµ b¶ng gi¸ trÞ cña T.

T[1] = D76AA478	T[17] = F61E2562	T[33] = FFFA3942	T[49] = F4292244
T[2] = E8C7B756	T[18] = C040B340	T[34] = 8771F681	T[50] = 432AFF97
T[3] = 242070DB	T[19] = 265E5A51	T[35] = 69D96122	T[51] = AB9423A7
T[4] = C1BDCEEE	T[20] = E9B6C7AA	T[36] = FDE5380C	T[52] = FC93A039
T[5] = F57C0FAF	T[21] = D62F105D	T[37] = A4BEEA44	T[53] = 655B59C3
T[6] = 4787C62A	T[22] = 02441453	T[38] = 4ADFCFA9	T[54] = 8F0CCC92
T[7] = A8304613	T[23] = D8A1E681	T[39] = F6BB4B60	T[55] = FFEFF47D
T[8] = FD469501	T[24] = E7D3FBC8	T[40] = BEBFBC70	T[56] = 85845DD1
T[9] = 698098D8	T[25] = 21E1CDE6	T[41] = 28987EC6	T[57] = 6FA87E4F
T[10] = 8B44F7AF	T[26] = C33707D6	T[42] = EAA127FA	T[58] = FE2CE6E0
T[11] = FFFF5BB1	T[27] = F4D50D87	T[43] = D4EF3085	T[59] = A3014314
T[12] = 895CD7EB	T[28] = 455A14ED	T[44] = 04881D05	T[60] = AE0811A1
T[13] = 6B901122	T[29] = A9E3E905	T[45] = D4D9D039	T[61] = F7537E82
T[14] = FD987193	T[30] = FCEFA3F8	T[46] = 6EDB99E5	T[62] = BD3AF235
T[15] = A679438E	T[31] = 676F02D9	T[47] = 1FA27CF8	T[63] = 2AD7D2BB
T[16] = 49B40821	T[32] = 8D2A4C8A	T[48] = C4AC5665	T[64] = EB68D391

B-íc 5: §Çu ra:

Sau khi tÊt c¶ L khèi 512 bit ®· ®ược xö lý, th× ®Çu ra tõ tÇng thø L lµ th«ng b¸o 128 bit rót gän.

Chóng ta nghi¬n cøu chi tiÕt h¬n viÖc xö lý møc logic trong 4 vßng cña mçi khèi 512 bit. Mçi vßng bao gåm mét chuçi 16 b-íc xö lý tr¬n bé ®Öm ABCD. Mçi b-íc biÓu diÔn theo c«ng thøc d-íi ®©y:

$$a \leftarrow b + CLS_S\ (a + g(b,c,d) + X[k] + T[i])$$

Trong ®ã:

a, b, c, d = 4 tõ cña bé ®Öm, ®-îc chØ râ trËt tù qua mçi b-íc nhÈy.

g = lµ mét trong 4 hµm nguyªn thuû F, G, H, I.

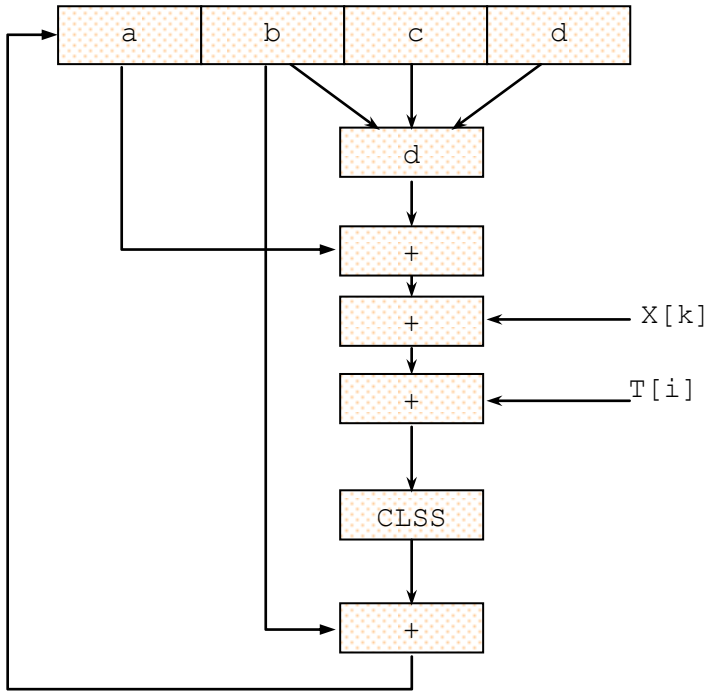
CLS_S = quay vßng dÞch tr¸i cña 32 bit argument bëi s bit.

X[k] = M[q*16 + k] = bit thø k trong 32 bit cña mét tõ trong bit thø q
cña mét khèi 512 bit cña th«ng b¸o.

T[i] = bit thø i cña tõ 32 bit trong ma trËn T.

+ = céng modulo 2^{32}.

M« h×nh ®-îc thÓ hiÖn nh- sau:



Hình 1.6: Các yếu tố của MD5

Mét trong 4 hµm logic nguyªn thuû ®-îc sö dông cho mçi vßng cña thuËt to¸n. Mçi hµm nguyªn thuû thùc hiÖn 3 tõ 32 bit ®Çu vµo vµ ®-a ra mét tõ 32 bit ®Çu ra. Mçi hµm thùc hiÖn mét tËp c¸c phÐp ®iÒu khiÓn logic c¸c bit,

v× vËy bÝt thø n cña ®Çu ra lµ mét hµm cña bit thø n cña 3 ®Çu vµo.

Tãm t¾t c¸c hµm nh- sau:

Round	Primitive function g	G(b,c,d)
f_F	F(b,c,d)	(b . c) ∨ ((~b) . d)
f_G	G(b,c,d)	(b . d) ∨ (c . (~d))
f_H	H(b,c,d)	b ⊕ c ⊕ d
f_I	I(b,c,d)	c ⊕ (b . (~d))

C¸c phÐp to¸n logic (AND, OR, NOT, XOR) ®-îc biÓu diÔn b»ng c¸c biÓu t-îng

(. , ∨ , ~ , ⊕). Hµm F lµ hµm ®iÒu kiÖn: if b then c else d. T-¬ng tù G cã thÓ ®-îc x¸c ®Þnh if d then b else c. Hµm H x©y dùng bit kiÓm tra ch½n lÎ.

B¶ng d-íi ®©y lµ b¶ng ch©n lý cña 4 hµm kÓ trªn.

b	C	d	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

1.4.2 Thuật toán MD5

```
/* Qu¸ tr×nh xö lý thùc hiÖn tõng khèi 512 bit (16 tõ)
*/
For q=0 to (N/16)-1 do
/* Copy khèi i vµo X */
For j=0 to 15 do
          Set X[j] to M[q*16 + j]
end /* KÕt thóc vßng lÆp trong j */

/* L-u gi¸ trÞ cña A vµo AA, B vµo BB, C vµo CC, vµ D
vµo DD */
    AA = A
    BB  = B
    CC  = C
    DD = D
/* Round 1 */
/* Cho [abcd  k  s  i] ®-îc thÓ hiÖn bëi c«ng thøc tÝnh
nh- sau
    a = b + ((a + F(b,c,d) + X[k] + T[i]) << s)
Thùc hiÖn 16 phÐp tÝnh d-íi ®©y: */
```

[ABCD 0 7 1]
[DABC 1 12 2]
[CDAB 2 17 3]
[BCDA 3 22 4]
[ABCD 4 7 5]
[DABC 5 12 6]
[CDAB 6 17 7]
[BCDA 7 22 8]
[ABCD 8 7 9]
[DABC 9 12 10]
[CDAB 10 17 11]
[BCDA 11 22 12]
[ABCD 12 7 13]
[DABC 13 12 14]
[CDAB 14 17 15]
[BCDA 15 22 16]

/* Round 2 */

/* BiÓu thøc [abcd k s i] ®-îc thÓ hiÖn bëi c«ng thøc tÝnh nh- sau

a = b + ((a + F(b,c,d) + X[k] + T[i]) << s)

Thùc hiÖn 16 phÐp tÝnh d-íi ®©y: */

[ABCD 1 5 17]
[DABC 6 9 18]
[CDAB 11 14 19]
[BCDA 0 20 20]
[ABCD 5 5 21]
[DABC 10 9 22]
[CDAB 15 14 23]
[BCDA 4 20 24]
[ABCD 9 5 25]
[DABC 14 9 26]
[CDAB 3 14 27]
[BCDA 8 20 28]
[ABCD 13 5 29]
[DABC 2 9 30]
[CDAB 7 14 31]
[BCDA 12 20 32]

```
/* Round 3 */
/* Cho [abcd   k    s    i] ®-îc thÓ hiÖn bëi c«ng thøc
tÝnh nh- sau
    a = b + ((a + F(b,c,d) + X[k] + T[i]) << s)
Thùc hiÖn 16 phÐp tÝnh d-íi ®©y: */
                  [ABCD    5    4   33]
                  [DABC    8   11   34]
                  [CDAB   11   16   35]
                  [BCDA   14   23   36]
                  [ABCD    1    4   37]
                  [DABC    4   11   38]
                  [CDAB    7   16   39]
                  [BCDA   10   23   40]
                  [ABCD   13    4   41]
                  [DABC    0   11   42]
                  [CDAB    3   16   43]
                  [BCDA    6   23   44]
                  [ABCD    9    4   45]
                  [DABC   12   11   46]
                  [CDAB   15   16   47]
                  [BCDA    2   23   48]
```

```
/* Round 4 */
/* Cho [abcd   k    s    i] ®-îc thÓ hiÖn bëi c«ng thøc
tÝnh nh- sau
    a = b + ((a + F(b,c,d) + X[k] + T[i]) << s)
Thùc hiÖn 16 phÐp tÝnh d-íi ®©y: */
                  [ABCD    0    6   49]
                  [DABC    7   10   50]
                  [CDAB   14   15   51]
                  [BCDA    5   21   52]
                  [ABCD   12    6   53]
                  [DABC    3   10   54]
                  [CDAB   10   15   55]
                  [BCDA    1   21   56]
                  [ABCD    8    6   57]
                  [DABC   15   10   58]
                  [CDAB    6   15   59]
                  [BCDA   13   21   60]
                  [ABCD    4    6   61]
                  [DABC   11   10   62]
                  [CDAB   14   15   63]
                  [BCDA    5   21   64]
/* T¨ng gi¸ trÞ cña A, B, C, D b»ng c¸ch céng l¹i víi
c¸c gi¸ trÞ l-u kÕt qu¶ trong trong b-íc trªn. C¸c gi¸
trÞ l-u lµ AA, BB, CC, DD */
A = A + AA
B = B + BB
C = C + CC
D = D + DD
end.  /* KÕt thóc vßng lÆp trong q */
```

Chương 2: KỸ THUẬT TẤN CÔNG RAINBOW ĐỐI VỚI RC4

Nội dung của chương này là chúng ta chỉ ra được các kỹ thuật tấn công mật khẩu, các phương pháp phân tích mã nguồn của các nghiên cứu có liên quan, bao gồm dự án Rainbow Project phiên bản 1.2; hệ phần mềm Abi-Word phiên bản 2.4.6 và phần mềm Wcracker. Trên cơ sở những phân tích kể trên để tiến hành xây dựng các kỹ thuật mà đề tài đã đăng ký thực hiện.

2.1. Các kỹ thuật tấn công mật khẩu

2.1.1 Kỹ thuật tấn công Bruteforce.

Bruteforce là cách thức thử tất cả các khả năng có thể có để đoán các thông tin cá nhân đăng nhập: tài khoản, mật khẩu, số thẻ tín dụng…Nhiều hệ thống cho phép sử dụng mật khẩu hoặc thuật toán mã hóa yếu sẽ tạo điều kiện cho tin tặc sử dụng phương pháp tấn công này để đoán tài khoản và mật khẩu đăng nhập. Sau đó sử dụng các thông tin này để đăng nhập truy cập vào tài nguyên hệ thống.

Biện pháp phòng tránh: Tăng cường độ mạnh cho mật khẩu (Độ dài ít nhất 6 ký tự, không chứa chuỗi username, chứa ít nhất 1 ký tự số, chứa ít nhất 1 ký tự đặc biệt, không cho phép thay đổi mật khẩu trùng lặp đã sử dụng, quản lý, điều khiển thông báo lỗi)

Sử dụng cơ chế chứng thực (Basic hoặc Digest Authentication).Hạn chế số lần đăng nhập hoặc khóa tài khoản đăng nhập sai Sử dụng module Mod_Dosevasive để xác định dấu hiệu của kiểu tấn công này

2.2.2. Kỹ thuật tấn công vào hệ thống có cấu hình không an toàn.

Cấu hình không an toàn cũng là một lỗ hổng bảo mật của hệ thống. Các lỗ hổng này được tạo ra do các ứng dụng có các thiết lập không an toàn hoặc người quản trị hệ thống định cấu hình không an toàn. Chẳng hạn như cấu hình máy chủ web cho phép ai cũng có quyền duyệt qua hệ thống thư mục. Việc thiết lập như trên có thể làm lộ các thông tin nhạy cảm như mã nguồn, mật khẩu hay các thông tin của khách hàng.

Nếu quản trị hệ thống cấu hình hệ thống không an toàn sẽ rất nguy hiểm vì nếu người tấn công duyệt qua được các file pass thì họ có thể download và giải mã ra, khi đó họ có thể làm được nhiều thứ trên hệ thống.

2.2.3. Kỹ thuật tấn công dùng Cookies.

Cookie là những phần tử dữ liệu nhỏ có cấu trúc được chia sẻ giữa website và trình duyệt của người dùng.

Cookies được lưu trữ dưới những file dữ liệu nhỏ dạng text (size dưới 4KB). Chúng được các site tạo ra để lưu trữ, truy tìm, nhận biết các thông tin về người dùng đã ghé thăm site và những vùng mà họ đi qua trong site. Những thông tin này có thể bao gồm tên, định danh người dùng, mật khẩu, sở thích, thói quen,

Cookies được Browser của người dùng chấp nhận lưu trên đĩa cứng của máy tính, không phải Browser nào cũng hỗ trợ cookies.

2.2.4. Kỹ thuật time – memory trade –off (TMTO) áp dụng trong bài toán tấn công mật mã.

Kỹ thuật đánh đổi bộ nhớ - thời gian còn có tên gọi khác là đánh đổi không gian- thời gian (Time Memory Trade-Off (TMTO) dùng để chỉ việc sử dụng bộ nhớ lưu trữ dữ liệu tính toán trước với mục đích giảm thời gian tính toán đối với một thao tác cụ thể. Đây là kỹ thuật được áp dụng trong một số bài toán có thể chia thao tác tính toán thành hai phần: tính toán trước và tra cứu dữ liệu đã chuẩn bị trước. Nếu tính toán và lưu trữ trước được càng nhiều thì thời gian giải một bài toán cụ thể sẽ chỉ tương đương với thời gian tra cứu.

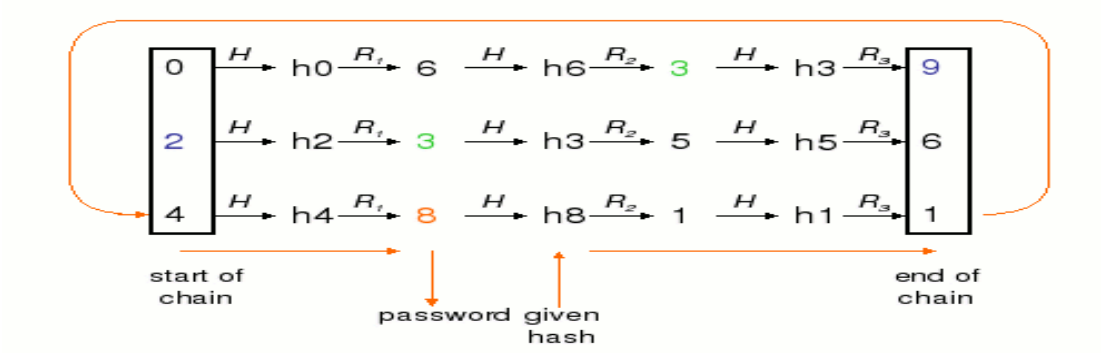
Các phương tiện lưu trữ máy tính ngày một lớn hơn làm cho khả năng ứng dụng kỹ thuật TMTO ngày càng hiện thực và hiện nay có nhiều ứng dụng sử dụng kỹ thuật này để giải quyết các vấn đề về tốc độ và bộ nhớ lưu trữ như: các bài toán liên quan đến tra cứu bảng dữ liệu, bài toán lưu trữ dữ liệu dạng nén, bài toán lưu trữ thuật toán, lưu trữ kết quả hình ảnh trong hiển thị công thức toán học trên trang HTML,...

Kỹ thuật mật mã cần làm việc với một không gian dữ liệu lớn (không gian khóa). Tuy nhiên, ở một số chế độ làm việc, có thể tổ chức tính toán sẵn các bản mã có thể của một bản rõ để thành lập một từ điển tra cứu cho phép mã hóa và giải mã nhanh. Mã thám có thể lợi dụng tính chất này để tấn công mật mã (kiểu tấn công Brute- Force) nếu có đủ bộ nhớ

2.2.5. Kỹ thuật RainbowCrack:

RainbowCrack là chương trình tạo ra các bảng cầu vồng được dùng để crack password đã được mã hóa. Là công cụ bẻ khóa mật khẩu "mạnh mẽ" rất hiệu quả và giảm thiểu thời gian crack Password.

Rainbow Table là một bảng lookup đưa ra một Time-Memory Trade-Off được sử dụng để khôi phục mật khẩu dạng text từ một password hash (băm). Các chương trình thường sử dụng giải thuật băm (hash) để lưu trữ mật khẩu, Rainbow Table được sử dụng để làm ngược lại quá trình hash.



Hình 2.1: Bảng cầu vồng-Rainbow Table

Thông thường các chương trình dò tìm mật khẩu thường dùng Brute Force Attach để thử với số lượng rất lớn các ký tự. Với máy tính hiện nay, việc thử này chỉ có hiệu quả khi chiều dài của mật khẩu ít hơn 8 ký tự. Với mật khẩu dài 7 ký tự và bao gồm tất cả các ký tự thì thời gian dò tìm theo kiểu Brute Force Attach mất khoảng 30 ngày. Tất nhiên với một số tính năng gắn kèm như từ điển, thuật toán thử,… thì khoảng thời gian này sẽ được rút ngắn đi, nhưng không đáng kể. Nếu sử dụng Rainbow Table, thời gian này khoảng 40 phút.

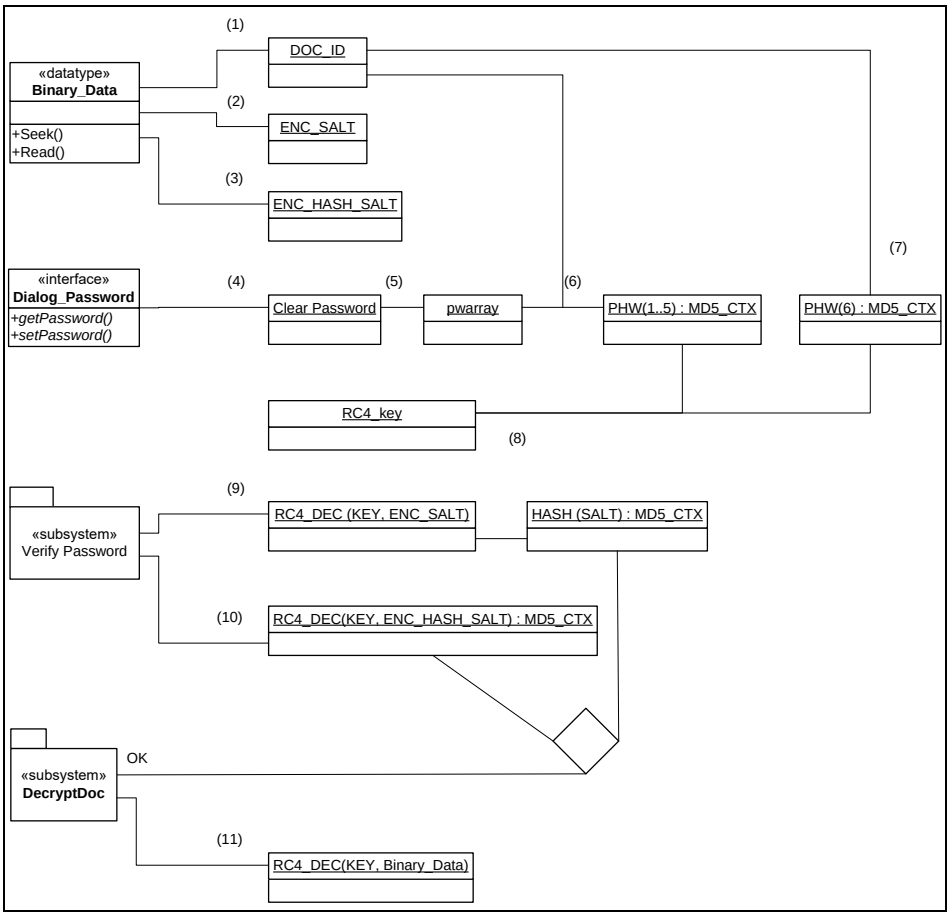
2.2.6 Xác thực mật khẩu bảo mật văn bản bằng MS- Word 2007

MS-Word 2007 sử dụng thuật toán mật mã RC4 với cơ chế xác thực mật khẩu khá phức tạp, có sử dụng muối mật khẩu (giá trị được sinh ngẫu nhiên cho mỗi tệp văn bản) và số nhận dạng mỗi văn bản (DOCID). Điều này gây cản trở cho việc ứng dụng trực tiếp kỹ thuật TMTO tấn công khóa mã RC4 với các bản mã tính toán trước. Sau khi xác định được cơ chế mật khẩu, Đề tài phải chuyển hướng sang tấn công tìm khóa mã RC4 tại thời điểm sử dụng giá trị băm MD5 để khởi tạo khóa mã trong sơ đồ xác thực mật khẩu mà MS-Word ứng dụng.

2.2.6.1 Lược đồ xác thực mật khẩu bảo mật văn bản

Để xây dựng được lược đồ xác thực mật khẩu trong MS-Word, Đề tài phải tiến hành việc khảo sát toàn bộ các thư viện mã nguồn của Abi-Word với những mức độ phân tích khác nhau. Qua đó, xác định được thư viện cốt lõi thực hiện xác thực mật khẩu của văn bản. Từ thư viện này, Đề tài tiếp tục xây dựng hướng kết hợp giữa phương pháp TMTO và cơ chế xác thực mật khẩu MS-Word để ứng dụng tấn công vào khâu tạo lập khóa cho RC4.

Phân tích mã của các hàm đã nêu ở trên, lược đồ 11 bước sau thực hiện việc nhận mật khẩu do người dùng đưa ra trên giao diện chương trình, xác nhận mật khẩu đúng và cuối cùng là giải mã văn bản để hiển thị nội dung trên giao diện chương trình Abi-Word. Đây cũng là kết quả chính của việc nghiên cứu bộ phần mềm AbiWord 2.4.6 phục vụ mục tiêu nghiên cứu của Đề tài.



Hình 2.2: Lược đồ xác thực mật khẩu bảo mật văn bản

Phương pháp TMTO chỉ có thể tham gia vào lược đồ xác thực khóa thay thế cho bước (8): tạo khóa RC 4 cho quá trnh xác thực bằng so sánh các giá trị băm

MD5 ở bước (9) và bước (10). Lược đồ này không tạo thuận lợi cho phương pháp TMTO do có tới hai giá trị ngẫu nhiên đối với mỗi văn bản, đó là DOCID và SALT. Đề tài thực sự không gặp thuận lợi như mong muốn ban đầu. Tuy nhiên, nhóm nghiên cứu tiếp tục tìm hiểu bộ phần mềm RainbowCrack theo hướng tạo các khóa quan hệ dùng thay thế cho bước (8) của lược đồ xác thực khóa của MS-Word.

2.2.6.2 Cấu trúc bộ phần mềm RainbowCrack

Bộ phần mềm RainbowCrack gồm hai thành phần: 1) Tạo các bảng Rainbow bằng các thuật toán tạo lập khóa như SHA1, MD5, LMHash và tập ký tự đầu vào mà các mật khẩu có thể sử dụng; và 2) Tấn công mật khẩu sử dụng các hàm, bảng Rainbow phù hợp.

Dựa vào kết quả nghiên cứu ta thấy rằng, các khóa RC4 được khởi tạo bằng MD5, chỉ sử dụng 5 byte đầu tiên của mỗi giá trị băm MD5, nên Đề tài không đi vào nghiên cứu theo hướng tạo các bảng Rainbow với RC4 khóa cố định hoặc bản rõ cố định có thể sử dụng để tìm khóa đúng cho lược đồ xác thực khóa MS-Word. Hướng nghiên cứu bộ phần mềm RainbowCrack được tập chung vào phần Tấn công mật khẩu sử dụng hàm băm MD5. Một số bảng MD5 đã được xây dựng sẵn sẽ tạo thuận lợi cho Đề tài.

Đề tài đã nghiên cứu kỹ thuật TMTO nhanh theo hướng lập bảng Rainbow của Philippe Oechslin. Đó là cơ sở để tìm ra và đi sâu nghiên cứu bộ phần mềm RainbowCrack do Zhu Shuanglei shuanglei@hotmail.com nghiên cứu cài đặt. RainbowCrack 1.1 được công bố mã nguồn trên Internet. Nhưng việc nghiên cứu mã nguồn gặp khó khăn do tài liệu về thiết kế cài đặt cũng như những mô tả ở ngay trong mã nguồn hầu như không có.

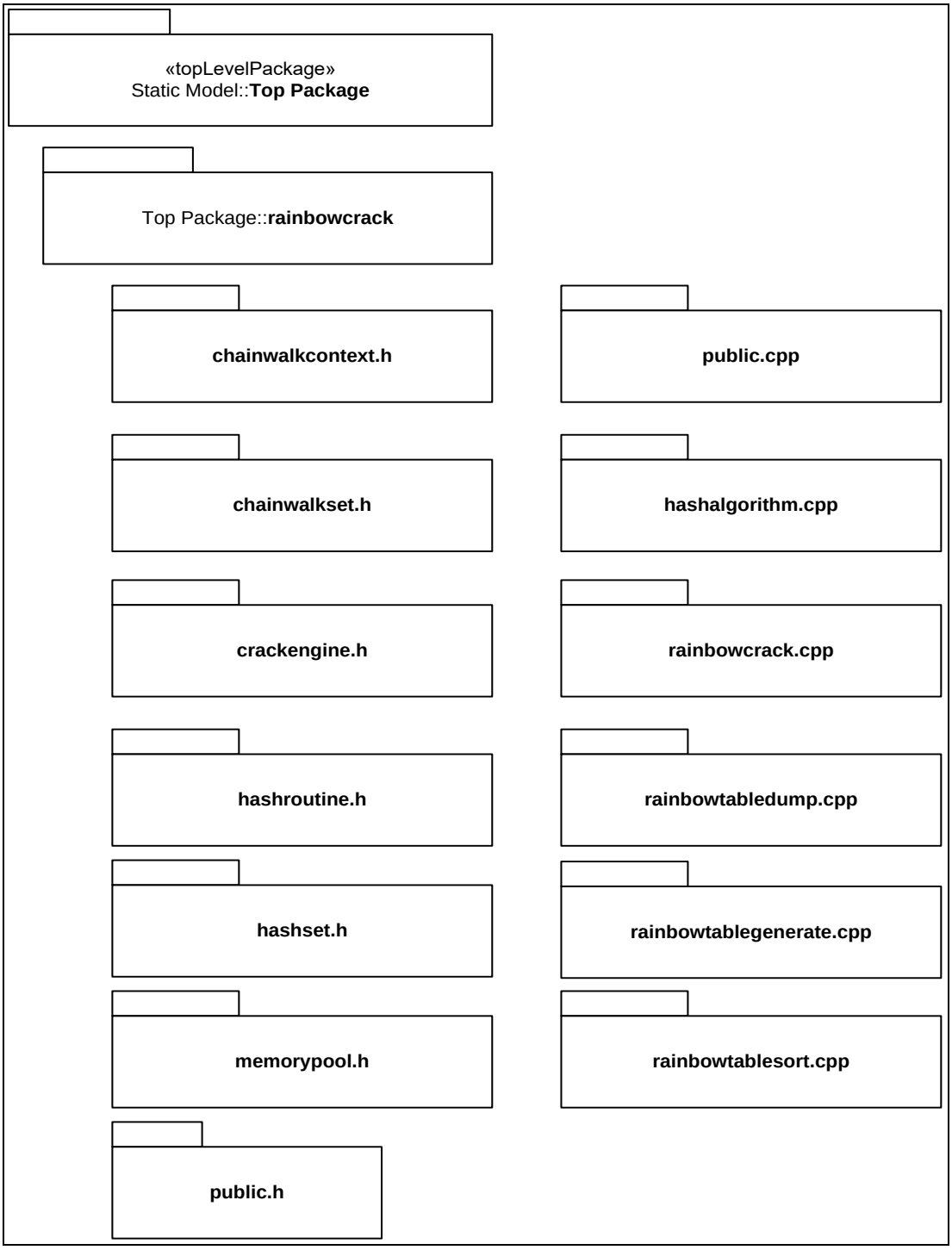
RainbowCrack gồm 9 class:

- ChainWalkContext,
- ChainWalkSet,
- CrackEngine,
- ChainWalk,
- HashRoutine,
- HashSet,
- MemoryPool,
- SortedSegment và

- RainbowChain.

Việc nghiên cứu mã nguồn Abi-Word giúp định hướng sát thực hơn trong nghiên cứu mã nguồn của RainbowCrack. Trên cơ sở lựa chọn hàm hash MD5, tấn công TMTO đối với mã hash MD5 được tiến hành từng bước, từ đó định ra các hàm chức năng cụ thể

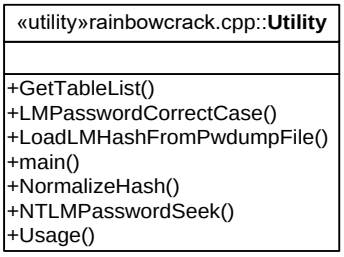
2.2.6.3Cấu trúc tổng thể bộ phần mềm RainbowCrack



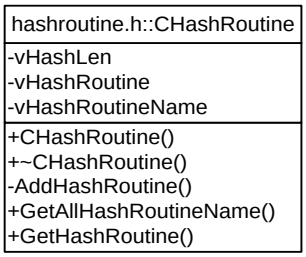
Hình 2.3: Cấu trúc tổng thể bộ phần mềm RainbowCrack

2.2.6.4 Một số hàm chính của RainbowRack

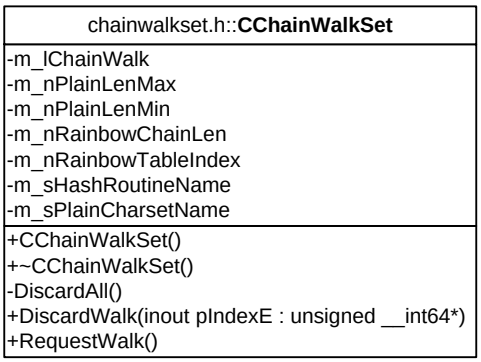
a. RainbowCrack



b. HashRoutine



c. ChainWalkSet



d. ChainWalkContext

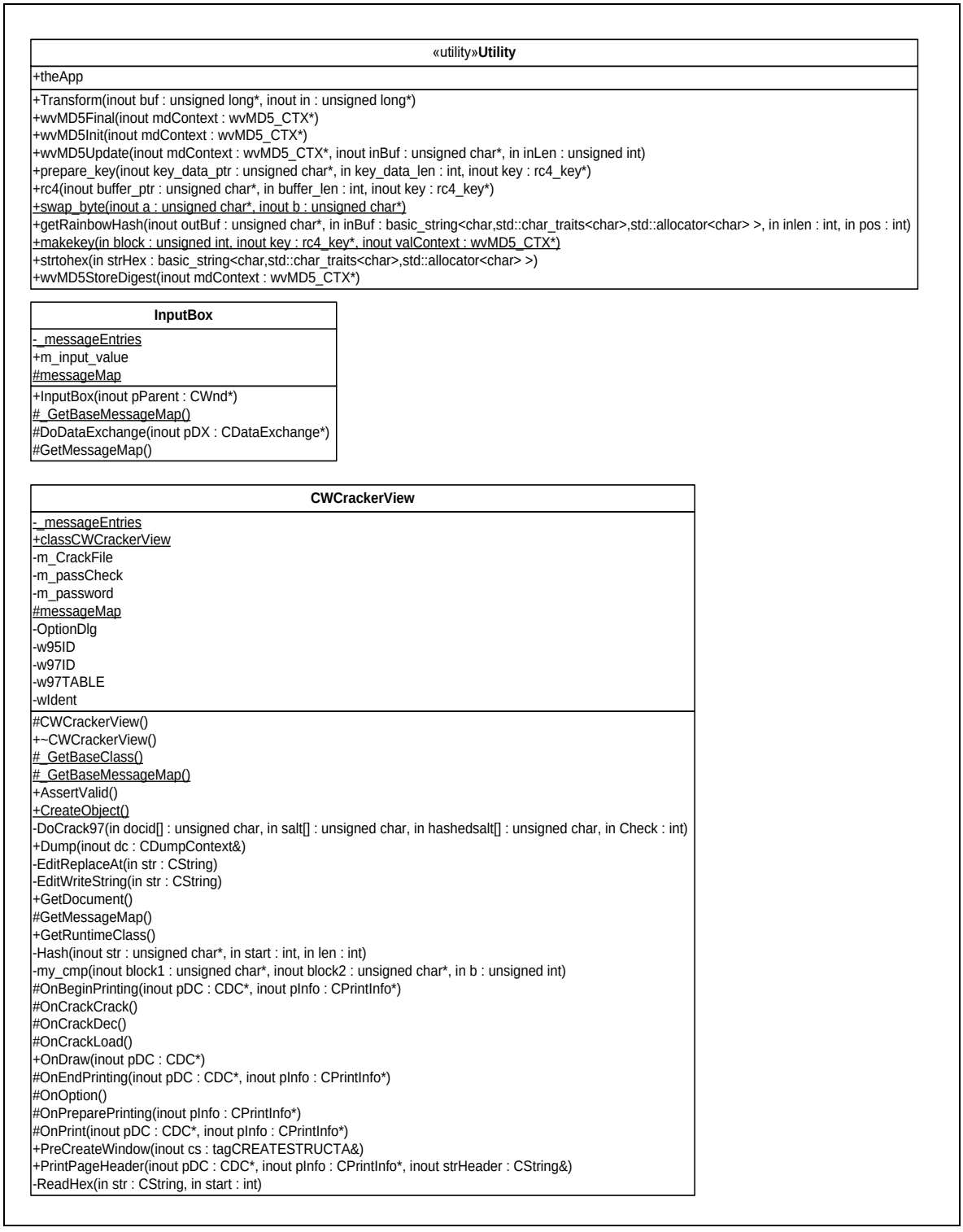
chainwalkcontext.h::**CChainWalkContext**
-m_Hash -m_nHashLen -m_nIndex -m_nPlainCharsetLen -m_nPlainLen -m_nPlainLenMax -m_nPlainLenMin -m_nPlainSpaceTotal -m_nPlainSpaceUpToX -m_nRainbowTableIndex -m_nReduceOffset -m_pHashRoutine -m_Plain -m_PlainCharset -m_sHashRoutineName -m_sPlainCharsetContent -m_sPlainCharsetName
+CChainWalkContext() +~CChainWalkContext() +CheckHash(inout pHash : unsigned char*) +Dump() +GenerateRandomIndex() +GetBinary() +GetHash() +GetHashLen() +GetHashRoutineName() +GetIndex() +GetPlain() +GetPlainBinary() +GetPlainCharsetContent() +GetPlainCharsetName() +GetPlainLenMax() +GetPlainLenMin() +GetPlainSpaceTotal() +GetRainbowTableIndex() +HashToIndex(in nPos : int) +IndexToPlain() -LoadCharset() +PlainToHash() +SetHash(inout pHash : unsigned char*) +SetHashRoutine() +SetIndex(in nIndex : unsigned __int64) +SetPlainCharset() +SetRainbowTableIndex() +SetupWithPathName()

e. CrackEngine

CCrackEngine
-m_cws -m_fTotalCryptanalysisTime -m_fTotalDiskAccessTime -m_nTotalChainWalkStep -m_nTotalChainWalkStepDueToFalseAlarm -m_nTotalFalseAlarm
+CCrackEngine() +~CCrackEngine() -BinarySearch(RainbowChain*) -CheckAlarm() -GetChainIndexRangeWithSameEndpoint() +GetStatTotalChainWalkStep() +GetStatTotalChainWalkStepDueToFalseAlarm() +GetStatTotalCryptanalysisTime() +GetStatTotalDiskAccessTime() +GetStatTotalFalseAlarm() -ResetStatistics() +Run() -SearchRainbowTable() -SearchTableChunk()

2.2.7 Cấu trúc phần mềm Wcracker

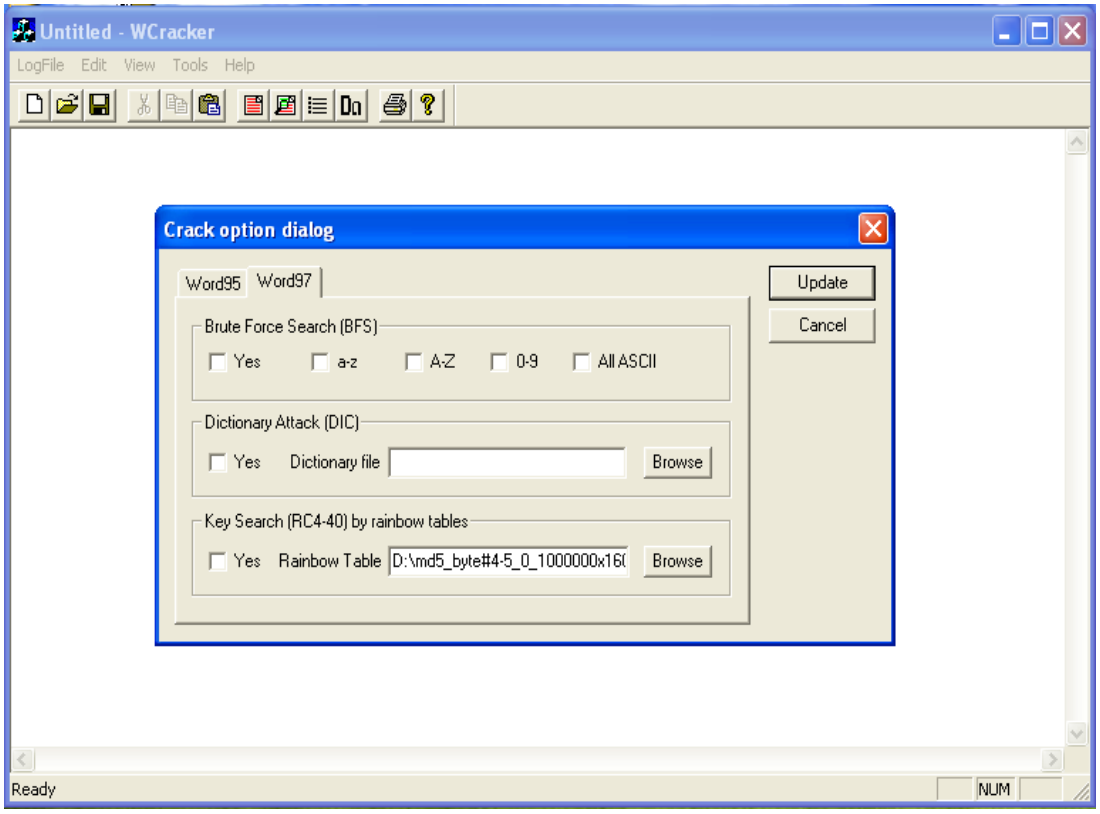
2.2.7.1 Phần mềm Wcracker



2.2.7.2 Nâng cấp đối với Wcracker

a. Tính năng nạp bảng Rainbow

Để sử dụng phương pháp tấn công mật khẩu bằng TMTO, Wcracker phải được nạp bảng Rainbow. Bảng này được lưu dưới dạng tệp nhị phân gồm các chỉ số của các Chain. Tên bảng được lưu trong Registry hệ thống thông qua giao diện hộp thoại option của Wcracker.



Hình 2.4: Hộp thoại option của Wcracker.

b. Kiểm tra mật khẩu

Tính năng này nhằm mục tiêu khẳng định mã nguồn xác thực mật khẩu hoạt động chính xác theo lược đồ 11 bước xác thực mật khẩu mà Đề tài đã xác định. Mã nguồn là kết quả nghiên cứu bộ phần mềm Abi-Word gồm các hàm chức năng như mở rộng mảng mật khẩu (expandpw), chuẩn bị khóa (prepare_key), khởi tạo khóa (makekey) và kiểm tra mật khẩu (verifypwd) được tích hợp trong một hàm chung.

Khi biến toàn cục m_passCheck == TRUE, lời gọi hàm DoCrack97 sẽ rẽ nhánh thực hiện đoạn mã xác thực mật khẩu mà Đề tài đã xây dựng.

Toàn bộ thử nghiệm mà Đề tài thực hiện đều cho kết quả xác thực được mật khẩu mà người sử dụng cung cấp đối với các tệp MS-Word. Dưới đây xin dẫn một

kết quả thử nghiệm để làm ví dụ. Lần thứ nhất, người dùng cung cấp mật khẩu sai, kết quả Wcracker thông báo "INVALID PASSWORD". Đối với trường hợp thứ hai, người dùng cung cấp mật khẩu đúng, chương trình Wcracker xác thực được mật khẩu đã cho và thông báo "VALID PASSWORD" đồng thời hiển thị mật khẩu mà người dùng đã cung cấp. Mật khẩu này được sử dụng để mở văn bản trong MS-Word hoặc Abi-Word.

```
--- Tuesday, October 05, 2010---
Test1.doc is Word97 or Word2K
Protected with Password to open
Document Identification Number:
882D2C65111D29B702A76D804C23AB7A
Encrypted Salt: E55902BD43184C970D8386B53607AB2D
Encrypted Hash of Salt: 2531586144ECF44BCCD2D44F73342D95
HashedPass:84C42A8455EB261FD5138C49CACAD144
INVALID PASSWORD or RC4-KEY NOT FOUND

--- Tuesday, October 05, 2010---
Test1.doc is Word97 or Word2K
Protected with Password to open
Document Identification Number:
882D2C65111D29B702A76D804C23AB7A
Encrypted Salt: E55902BD43184C970D8386B53607AB2D
Encrypted Hash of Salt: 2531586144ECF44BCCD2D44F73342D95
HashedPass:58EB96266CC29407EC3685F543F54167
VALID PASSWORD = test1
```

c. Tấn công mật khẩu

Mã nguồn cập nhật cho các tính năng kiểm tra mật khẩu, tấn công mật khẩu bằng phương án sử dụng bảng Rainbow MD5 xây dựng trước được cập nhật trong hàm chức năng DoCrack97 của lớp CWCrackerView.

+) Chức năng kiểm tra mật khẩu

```
if (Check) {
    //Kiem tra mat khau do nguoi dung cung cap
    InputBox m_InputBox;
    if (m_InputBox.DoModal() != IDOK) { return (0); }
    int passLen = m_InputBox.m_input_value.GetLength();
    if (passLen > 16) passLen = 16;
    m_password.Empty();
    for (i =0; i < passLen; i++)
        m_password += m_InputBox.m_input_value.GetAt(i);
    /* expandpw expects null terminated 16bit unicode input */
    /* Kiểm ta mật khẩu theo qui trình kiểm tra*/
    /* Tính toán giá trị HASH của mật khẩu và in kết quả */
    temp = "HashedPass:";
    for (i = 0; i < 16; i++) {
        stemp.Format("%02X", valContext.digest[i]);
        temp += stemp;
    }
    EditWriteString(temp);
    /* Giải mã HASH mật khẩu lưu trên tệp
    makekey (0, &key, &valContext);
    memcpy(tmp_salt, salt, 64);
    memcpy(tmp_hashedsalt, hashedsalt, 16);
    rc4 (tmp_salt, 16, &key);
    rc4 (tmp_hashedsalt, 16, &key);
    /* TínhHASHPASS vào mdContext2 */
    /* So sánh và ra quyết định */
    ret = !memcmp (mdContext2.digest, tmp_hashedsalt, 16);
    return (ret);
    }
```

+) Chức năng tấn công tìm khóa RC4

```
/* Chức năng tấn công tìm khóa RC4 sử dụng bảng Rainbow */
   RainbowChain *pChain;
   CChainWalkContext cwc;
   int nRainbowChainIndex = 0;
   int nRainbowChainLen, nRainbowChainCount;
   if (!CChainWalkContext::SetupWithPathName(
      sPathName, nRainbowChainLen, nRainbowChainCount))
      return 0;
   if (nRainbowChainIndex < 0
         || nRainbowChainIndex > nRainbowChainCount - 1) {
            printf("valid rainbow chain index range: 0 - %d\n",
                        nRainbowChainCount - 1);
            return 0;
   }
   // Open file
   FILE* file = fopen(sPathName.c_str(), "rb");
   if (file == NULL) {
      printf("failed to open %s\n", sPathName.c_str());
      return 0;
   }
   // Dump
        unsigned int nFileLen = GetFileLen(file);
        if (nFileLen != nRainbowChainCount * 16)
                printf("rainbow table size check fail\n");
        else {
                // Read all chains from table
      temp.Format("Free memory size: %u bytes",
GetAvailPhysMemorySize());
      EditWriteString(temp);
      static CMemoryPool mp;
      unsigned int nAllocatedSize;
```

```
        pChain = (RainbowChain*)mp.Allocate(nFileLen, nAllocatedSize);
        nAllocatedSize = nAllocatedSize / 16 * 16;
        while (true) {
            fseek(file, nRainbowChainIndex * 16, SEEK_SET);
            if (ftell(file) == nFileLen)
                break;
            unsigned int nDataRead = fread(pChain, 1, nAllocatedSize, file);
            //fread(&chain, 1, 16, file);
            int nRainbowChainCountRead = nDataRead / 16;
            temp.Format("Read chunk of %d chains: ",
nRainbowChainCountRead);
            EditWriteString(temp);
            for (int nChainIndex = 0;
                    nChainIndex < nRainbowChainCountRead;
                    nChainIndex++) {
                    // Trying for each chain
                    temp.Format("ChainIndex %d -- Free memory size: %u bytes",
                                    nChainIndex, GetAvailPhysMemorySize());
                    // EditWriteString(temp);
                AfxGetApp()->WriteProfileString("option", "runx",temp);
            cwc.SetIndex(pChain[nChainIndex].nIndexS);
            int nPos;
            for (nPos = 0; nPos < nRainbowChainLen - 1; nPos++) {
                cwc.IndexToPlain();
                cwc.PlainToHash();
                cwc.HashToIndex(nPos);
                //1. Trying HashIndex
                wvMD5Init (&valContext);
                getRainbowHash(valContext.digest, cwc.GetHash().c_str(), 16, 0);
                //Prepare RC4 key
                //Giải mã SALT
                //Giải mã HashedSalt
```

```
            //Tính HASH của SALT đã giải mã
            // So sánh 2 giá trị và ra quyết định
            ret = my_cmp (mdContext2.digest, tmp_hashedsalt, 16);
            if (ret == 0) {
               temp = "FOUND RC4 KEY: ";
               AfxGetApp()->WriteProfileString("option", "found",temp);
               nPos = nRainbowChainLen;
               nChainIndex = nRainbowChainCountRead;
               ret = 1;
            } else { ret = 0; }
            wvMD5Final(&valContext);
         }
         //Next chain
         temp.Format("Finished Chain %d at %d", nChainIndex, nPos);
         AfxGetApp()->WriteProfileString("option", "run",temp);
      } //for nChainIndex
   } //while next reading from file
   return (ret);
}
```

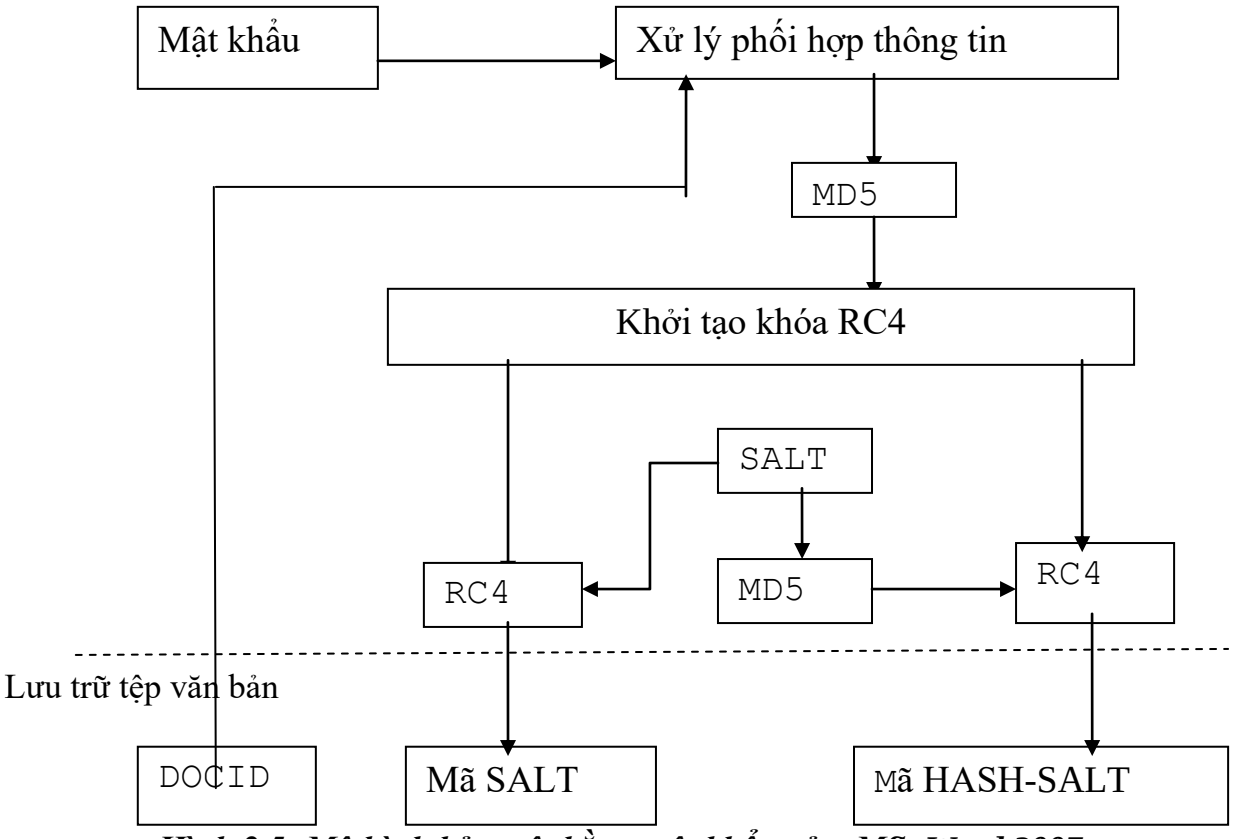
2.2.8.Mô hình bảo mật tệp văn bản MS- Word 2007

Mô hình bảo mật bằng mật khẩu của MS- Word 2007 dựa trên ứng dụng các thuật toán mật mã RC4 với độ dài khoảng 40 bít và thuật toán băm MD5. Mô hình này có sử dụng muối mật khẩu(SALT). Muối mật khẩu là những chuỗi ngẫu nhiên gồm 16 byte được lựa chọn cho từng văn bản. Mỗi văn bản lại có chuỗi nhận dạng riêng cũng gồm 16 byte được đặt tên là DOCID trong cấu trúc OLE2.

Mật khẩu được sử dụng như một giá trị khởi tạo để tạo lên giá trị băm MD5 chuẩn bị cho khóa mã RC4 (chỉ ứng dụng 5 byte đầu trong số 16 byte giá trị MD5). Muối mật khẩu (SALT) đóng vai trò bản rõ trong mã khóa RC4. Kết quả mã hóa SALT được lưu trên tệp MS- Word 2007. Đồng thời, SALT được băm một lần nữa với MD5 và tiếp tục được mã hóa RC4 với cùng khóa. Giá trị mã hóa này được gọi là HASH-SALT cũng được lưu trữ trên tệp MS- Word 2007.

Khi người dùng cung cấp mật khẩu qua giao diện của trình soạn thảo văn bản MS- Word 2007 trong quá trình mở tệp. Mật khẩu này sẽ được băm và tạo ra khóa mã RC4 như mô tả ở đoạn trên. Khóa này sẽ được sử dụng để giải mã SALT và HASH-SALT đọc từ văn bản. Giá tri SALT được băm bằng MD5, kết quả băm sẽ được so sánh với HASH-SALT đã giải mã, nếu thống nhất thì mật khẩu được xác thực.

Sơ đồ sau đây mô tả quá trình trên:



Hình 2.5: Mô hình bảo mật bằng mật khẩu của MS- Word 2007

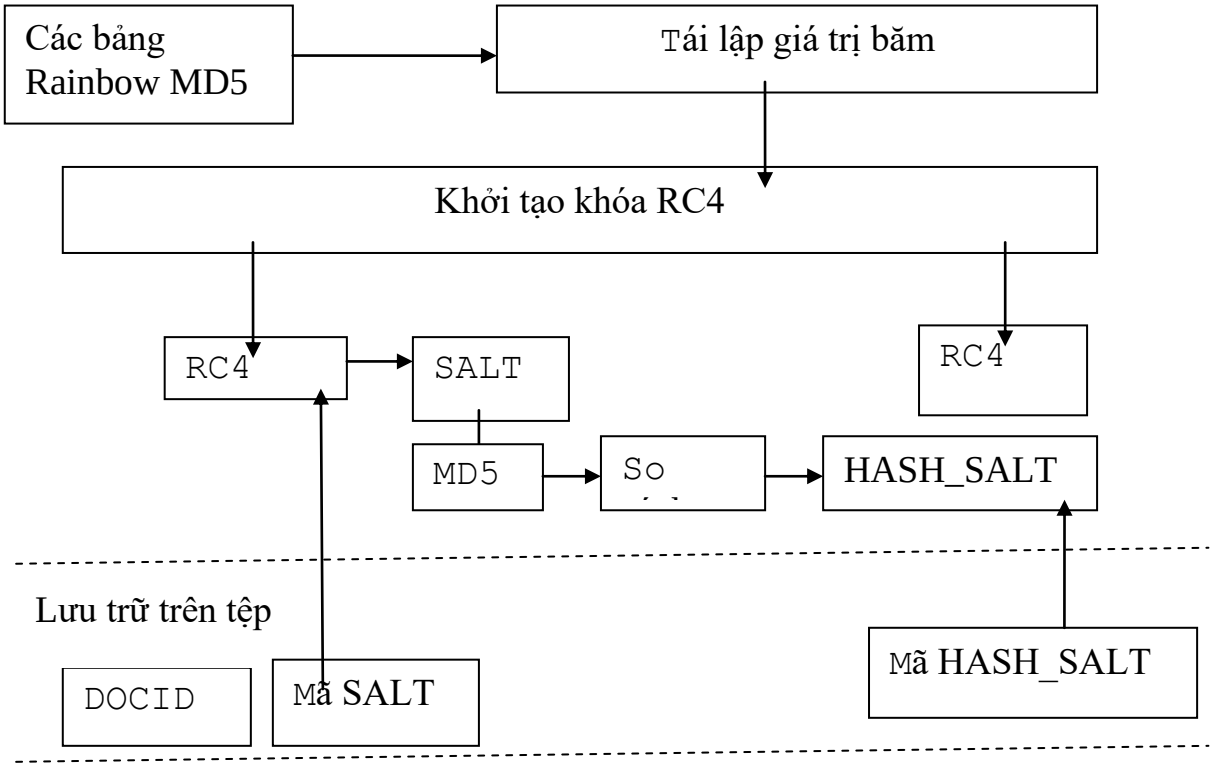
2.2.9. Lựa chọn điểm tấn công

Mô hình bảo mật bằng mật khẩu có sử dụng muối của MS- Word 2007 không cho phép tính toán trước các giá trị mã hóa SALT với toàn bộ các khóa có thể của RC4 do SALT là giá trị ngẫu nhiên. Nếu tính toán trước sẽ mở rộng thêm 2^{128}. Cũng không thể áp dụng tấn công Rainbow với MD5 do giá trị băm để tạo khóa RC4 cũng được lưu trữ trên tệp MS- Word 2007.

Kỹ thuật Rainbow chỉ có thể được lợi dụng để tìm khóa đúng của RC4. Trên cơ sở các bảng Rainbow MD5 ta có thể chiết xuất các khóa 5 byte cho RC4 và xác thực khóa đúng thông qua so sánh SALT và HASH_SALT sau giải mã như chu trình xác thực mật khẩu của MS- Word 2007 mô tả trên.

Kết quả mô hình tấn công là tìm được khóa đúng RC4. Khóa này được sử dụng để giải mã tệp văn bản MS- Word 2007 bằng phần mềm Gua Word là kết quả nghiên cứu của đề tài" Thám mã mật khẩu văn bản MS-Word" năm 2007 của phòng 5 A22. Gua Word được ứng dụng ở chế độ giải mã khi có khóa nên sẽ cho kết quả giải mã tệp tức thì.

Mô hình tấn công như sau:

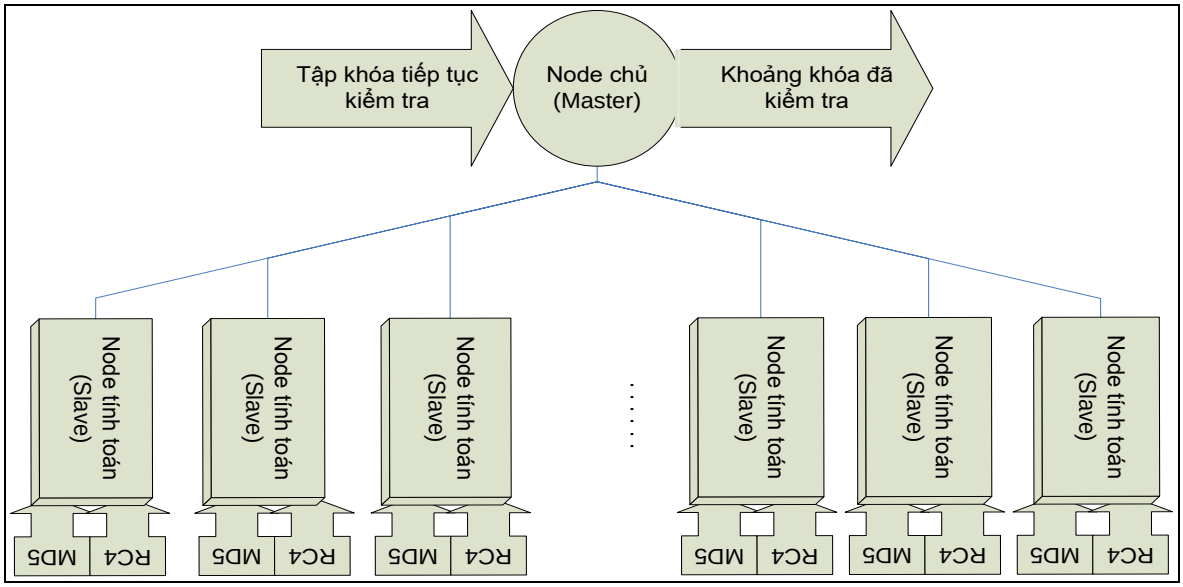


Hình 2.6: Mô hình bảo mật bằng mật khẩu

2.2.10. Phần mềm song song tìm khóa RC4 trong Word 2007

2.2.10.1. Mô hình tính toán song song

Phần mềm tính toán song song tìm khóa RC4 ứng dụng trong MS-Word 2007 được xây dựng trên môi trường hệ điều hành Linux, cài đặt môi trường truyền thông điệp LAM/MPI. Mô hình xử lý song song được lựa chọn như sau:



Hình 2.7: Mô hình tính toán song song

Ở mô hình lựa chọn, Master làm chủ toàn bộ cây tính toán song song. Các node tính toán và node Master gửi nhận thông điệp điều khiển qua kênh LAM/MPI. Ngoài kiểm soát hoạt động của các node tính toán, Master sẽ cung cấp khoảng khóa cần truy vấn cho từng node tính toán và nhận kết quả về từ node đó.

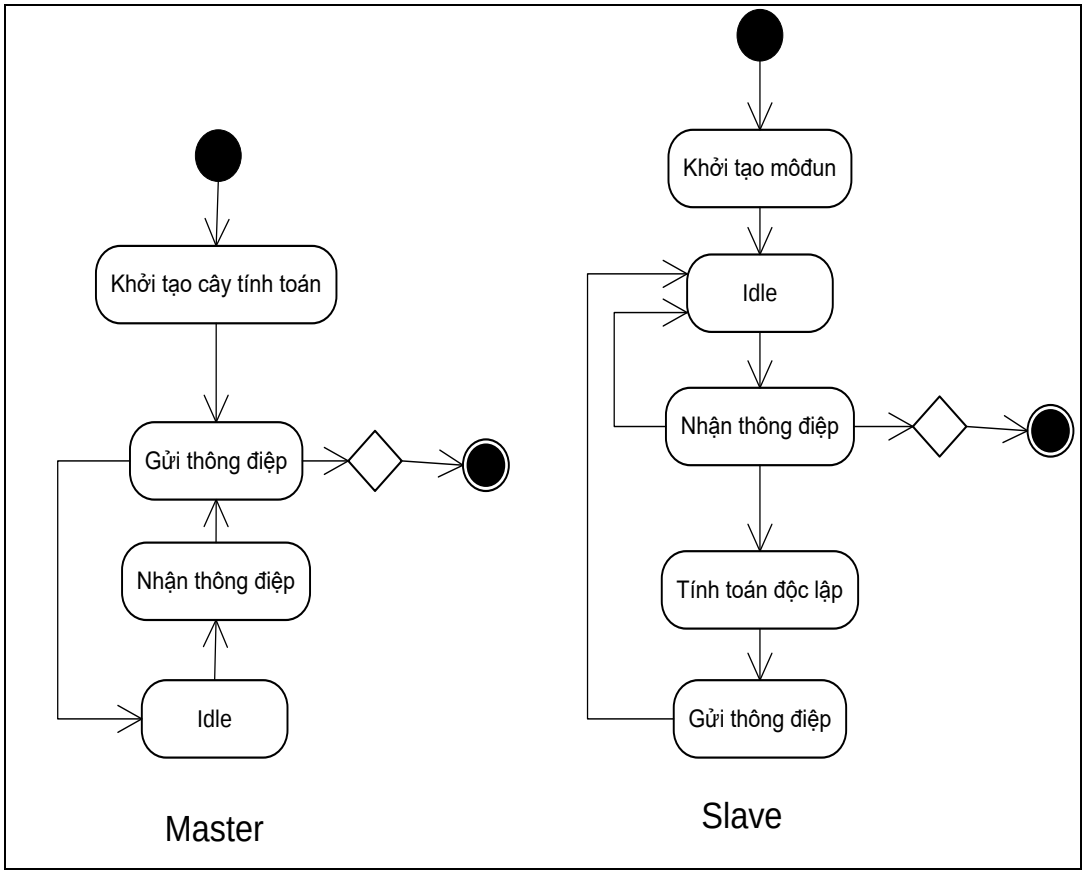
Các node tính toán (Slave) được đăng ký với Master, nhận khoảng khóa cần tìm kiếm và thực hiện tìm kiếm độc lập với Master. Các node tính toán làm việc song song. Kết quả trả về Master sẽ là thông điệp thông báo dã tìm kiếm hết khoảng khóa được cấp (chưa thấy khóa đúng) hoặc thông điệp đã tìm thấy khóa đúng, xin ngắt toàn bộ các node khác.

2.2.10.2 Lưu đồ trạng thái của Master và Slave

Master sau khi khởi tạo chính mình, tiến hành khởi tạo các node tính toán theo như đã đăng ký. Khi các node khởi tạo xong, Master bắt đầu gửi các thông điệp yêu cầu tính toán đến các node thành viên. Khi gửi hết các thông điệp, Master vào trạng thái chờ kết quả. Khi có thông điệp đến, Master chuyển sang trạng thái nhận thông điệp, xử lý thông điệp nhận được. Thông thường sau khi xử lý thông điệp nhận

được, Master chuyển sang trạng thái gửi thông điệp, để gửi thông tin tính toán tiếp theo cho node tính toán hoặc gửi thông điệp báo kết thúc cho node đó.

Khi đã gửi hết thông điệp kết thúc, Master chuyển đến trạng thái cuối cùng và tự hủy.



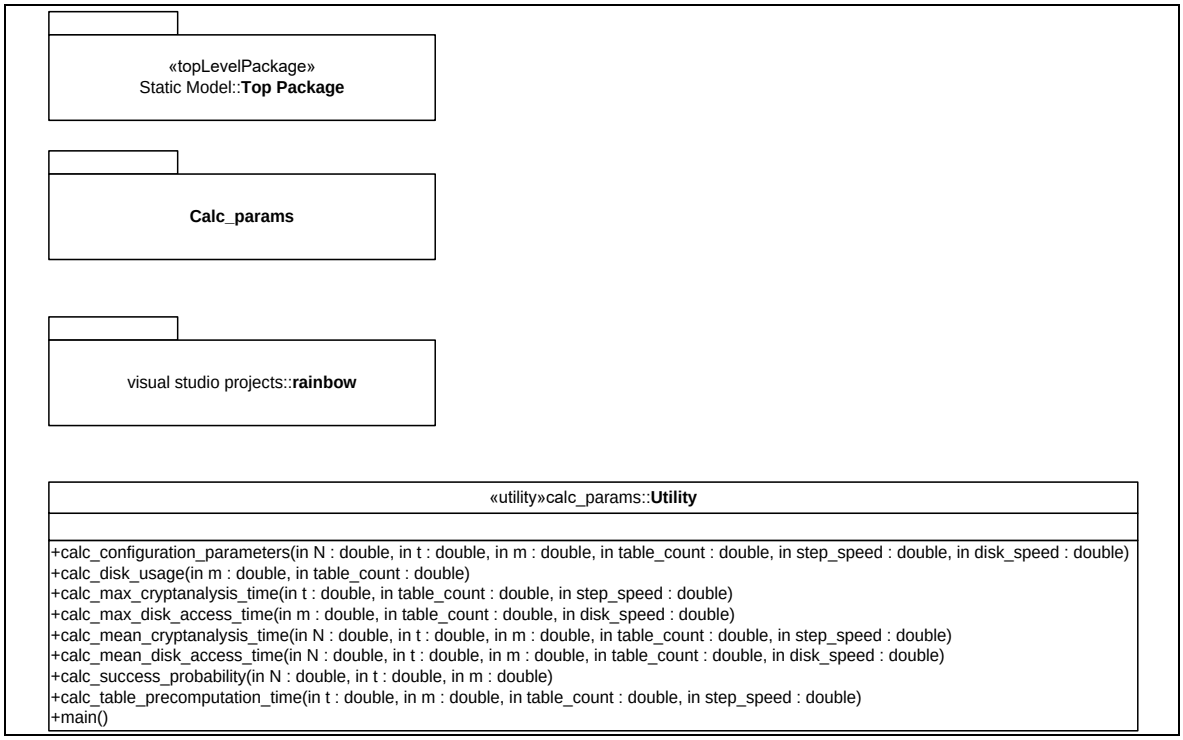
Hình 2.8: Lưu đồ trạng thái của Master và Slave

Node tính toán khởi tạo các môđun tính toán và chuyển sang trạng thái chờ. Khi thông điệp LAM/MPI từ Master đến, node tính toán sẽ nhận thông điệp, nếu là thông điệp hủy, node tính toán sẽ chuyển đến trạng thái cuối cùng để tự hủy, nếu là thông số tính toán, node này sẽ bóc tách các thông tin cần thiết và chuyển vào trạng thái tính toán độc lập. Kết quả tính toán sẽ được node tính toán gửi về Master trong trạng thái gửi thông điệp ở lưu đồ trên.

2.2.11. Phần mềm tính toán tham số tấn công Rainbow đối với RC4

2.2.11.1 Cấu trúc tĩnh của chương trình

Phần mềm tính toán tham số xây dựng các bảng Rainbow tấn công RC4 được xây dựng kiểu Console Application gồm 9 tiện ích bao gồm cả hàm chương trình chính main().



Các hàm tiện ích thực hiện các chức năng sau:

+) calc_configuration_parameters(): Hiển thị các tham số đã tính toán với khuôn dạng phù hợp với màn hình Console.

+) calc_disk_usage(): Tính dung lượng đĩa cứng cần thiết để lưu trữ các bảng Rainbow được xây dựng trên cơ sở các tham số đã lựa chọn.

+) calc_max_disk_access_time(): Tính số lượng truy vấn đĩa cứng để đọc các chuỗi từ các bảng Rainbow với tham số đã lựa chọn. Đây là toàn bộ số lần đọc dữ liệu từ môi trường lưu trữ các bảng Rainbow khi tấn công.

+) calc_mean_disk_access_time(): Tính số lượng truy vấn đĩa cứng trung bình cho mỗi tấn công, khi sử dụng các bảng Rainbow với tham số đã chọn.

+) calc_max_cryptanalysis_time(): Tính toán lượng thời gian tối đa cho một tấn công, khi sử dụng các bảng Rainbow với tham số đã lựa chọn.

+) calc_mean_cryptanalysis_time(): Tính toán lượng thời gian trung bình cho một tấn công, khi sử dụng các bảng Rainbow với tham số đã chọn.

+) calc_success_probability(): Tính xác suất thành công của mỗi tấn công, khi sử dụng 1 trong số các bảng Rainbow với tham số đã chọn.

Xác suất thành công của 1 bảng Rainbow được tính toán bằng công thức:

$p = 1-(1-1/N)^{sum(m_i)}$ với $m_i = N(1-(1-1/N)^{m_{i-1}})$; $m_0 = m$.

Xác suất thành công của toàn bộ các bảng Rainbow sẽ là xác suất kết hợp và được tính bằng công thức:

$P = 1 – (1 – p)^l$

với P là xác suất thành công khi sử dụng toàn bộ các bảng Rainbow đã xây dựng, p là xác suất thành công của 1 bảng, l là số bảng Rainbow sử dụng trong tấn công.

+) calc_table_precomputation_time(): Tính thời gian cần thiết để xây dựng các bảng Rainbow với tham số đã chọn. Đây là tính toán trước trong phương pháp tấn công TMTO. Thời gian tính toán trước đảm bảo cho thời gian tấn công giảm xuống.

2.2.11.2 Giải thuật của các hàm chức năng

Sau đây trình bày giải thuật các hàm chức năng đã mô tả ở trên. Hình thức mô tả giải thuật bằng ngôn ngữ mô tả dựa trên các hàm chuẩn của ngôn ngữ C.

+) calc_configuration_parameters(): Hiển thị các tham số đã tính toán với khuôn dạng phù hợp với màn hình Console.

```
void calc_configuration_parameters
      (in out double N, t, m, table_count, step_speed, disk_speed)
      // In ket qua tinh toan tham so cho Rainbow
      // Nguoi su dung lua chon cac tham so: N (phu thuoc do dai khoa ma);
      // t, table_count
      {
        inketqua (Manhinh, N);      //Khong gian khoa cua RC4
        inketqua (Manhinh, t);      //Do dai chuoi Rainbow
        inketqua (Manhinh, m);      //So chuoi Rainbow trong moi bang
        inketqua (Manhinh, table_count);  //So bang Rainbow
        inketqua (Manhinh, calc_disk_usage());
        inketqua (Manhinh, 1 - pow((1 - p), table_count));
        inketqua (Manhinh, calc_mean_cryptanalysis_time());
        inketqua (Manhinh, calc_max_cryptanalysis_time());
        inketqua (Manhinh, calc_mean_disk_access_time());
        inketqua (Manhinh, calc_max_disk_access_time());
        inketqua (Manhinh, calc_table_precomputation_time());
      }
```

+) calc_disk_usage(): Tính dung lượng đĩa cứng cần thiết để lưu trữ các bảng Rainbow được xây dựng trên cơ sở các tham số đã lựa chọn.

```
double calc_disk_usage(in out double m, table_count) {
    //Dung luong luu tru tinh bang GB
    //TABLE_ENTRY_SIZE = 10 (bytes) luu tru cap(Begin_point, End_point)
      return ceil(m*TABLE_ENTRY_SIZE*table_count / 1024 / 1024 / 1024);
    }
```

+) calc_max_disk_access_time(): Tính số lượng truy vấn đĩa cứng để đọc các chuỗi từ các bảng Rainbow với tham số đã lựa chọn. Đây là toàn bộ số lần đọc dữ liệu từ môi trường lưu trữ các bảng Rainbow khi tấn công.

```
double calc_max_disk_access_time
    (in out double m, table_count, disk_speed) {
      return (m*TABLE_ENTRY_SIZE/1024/1024 * disk_speed * table_count);
    }
```

+) calc_mean_disk_access_time(): Tính số lượng truy vấn đĩa cứng trung bình cho mỗi tấn công, khi sử dụng các bảng Rainbow với tham số đã chọn.

```
double calc_mean_disk_access_time
    (in out double N, t, m, table_count, disk_speed) {
      double rate_of_all = calc_success_probability(N, t, m);
      double temp = rate_of_all;
      double all  = 0;
      for (double i = 1; i < table_count; i++) {
            all = all + temp * i;
            temp = temp * (1 - rate_of_all);
      }
      all = all + pow((1 - rate_of_all), table_count) * table_count;

      return (m*TABLE_ENTRY_SIZE/1024/1024 * disk_speed * all);
    }
```

+) calc_max_cryptanalysis_time(): Tính toán lượng thời gian tối đa cho một tấn công, khi sử dụng các bảng Rainbow với tham số đã lựa chọn.

```
double calc_max_cryptanalysis_time
    // Tham so step_speed phu thuoc vao thuat toan mat ma va toc
    // do xu ly cua may tinh
    (in out double t, table_count, step_speed) {
    return (t*t/2 / step_speed * table_count);
    }
```

+) calc_mean_cryptanalysis_time(): Tính toán lượng thời gian trung bình cho một tấn công, khi sử dụng các bảng Rainbow với tham số đã chọn.

```
double calc_mean_cryptanalysis_time
    (in out double N, t, m, table_count, step_speed) {
    double rate_of_all = calc_success_probability(N, t, m);
    double temp = rate_of_all;
    double all  = 0;
    for (double i = 1; i < table_count; i++) {
            all = all + temp * i;
            temp = temp * (1 - rate_of_all);
    }
    all = all + pow((1 - rate_of_all), table_count) * table_count;

    return (all * t*t/2 / step_speed);
    }
```

+) calc_success_probability(): Tính xác suất thành công của mỗi tấn công, khi sử dụng các bảng Rainbow với tham số đã chọn.

```
double calc_success_probability
    (in out double N, t, m);
    // Tinh toan tham so ty le thanh cong cua moi bang trong
    // Rainbow series. Cong thuc tinh nhu da mo ta
    // Abstracted
```

+) calc_table_precomputation_time(): Tính thời gian cần thiết để xây dựng các bảng Rainbow với tham số đã chọn. Đây là tính toán trước trong phương pháp tấn công TMTO. Thời gian tính toán trước đảm bảo cho thời gian tấn công giảm xuống.

```
double calc_table_precomputation_time
      (in out double t, m, table_count, step_speed) {
        double time_in_second = t * m * table_count / step_speed;
        double time_in_day = time_in_second / 3600 / 24;
        return (time_in_day);
      }
```

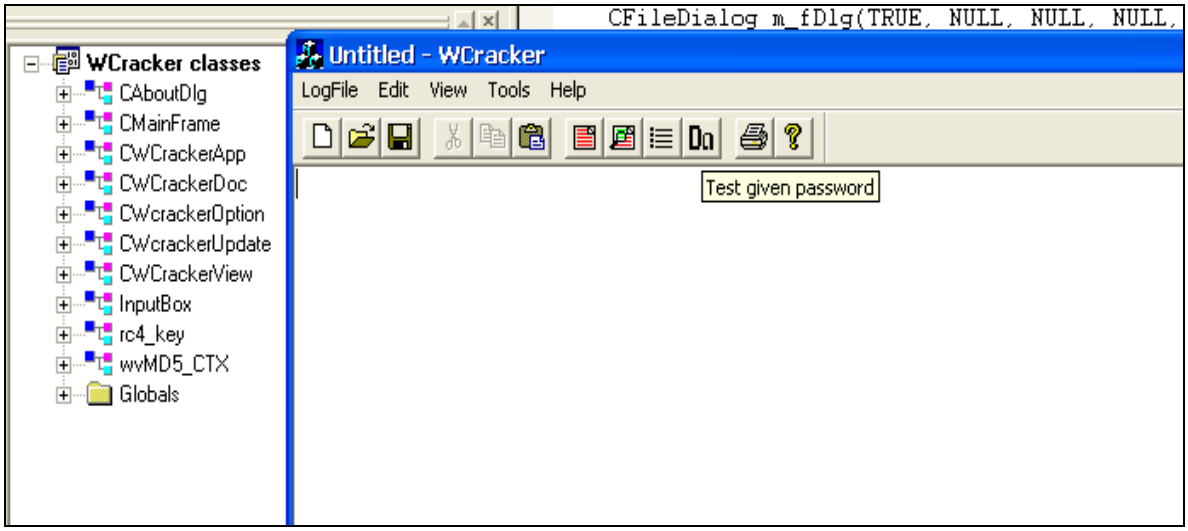
Chương 3: XÂY DỰNG CHƯƠNG TRÌNH TÍNH TOÁN THAM SỐ TẤN CÔNG RAINBOW ĐỐI VỚI RC4

Cho tệp văn bản được soạn thảo bằng MS-Word 2007 có cài đặt mật khẩu. Bài toán đặt ra là chúng ta phải tìm được cơ chế để giải mã được các tệp văn bản MS-Word 2007 đó. Để giải quyết bài toán này ta sẽ sử dụng trực tiếp kỹ thuật TMTO tấn công khóa mã RC4 với các bản mã tính toán trước. Sau khi xác định được cơ chế mật khẩu Ta phải chuyển hướng sang tấn công tìm khóa mã RC4 tại thời điểm sử dụng giá trị băm MD5 để khởi tạo khóa mã trong sơ đồ xác thực mật khẩu mà MS-Word ứng dụng như đã nêu ở hai chương trước.

3.1 Các tính năng tấn công RC4 trong Wcracker

3.1.1 Chức năng kiểm tra mật khẩu

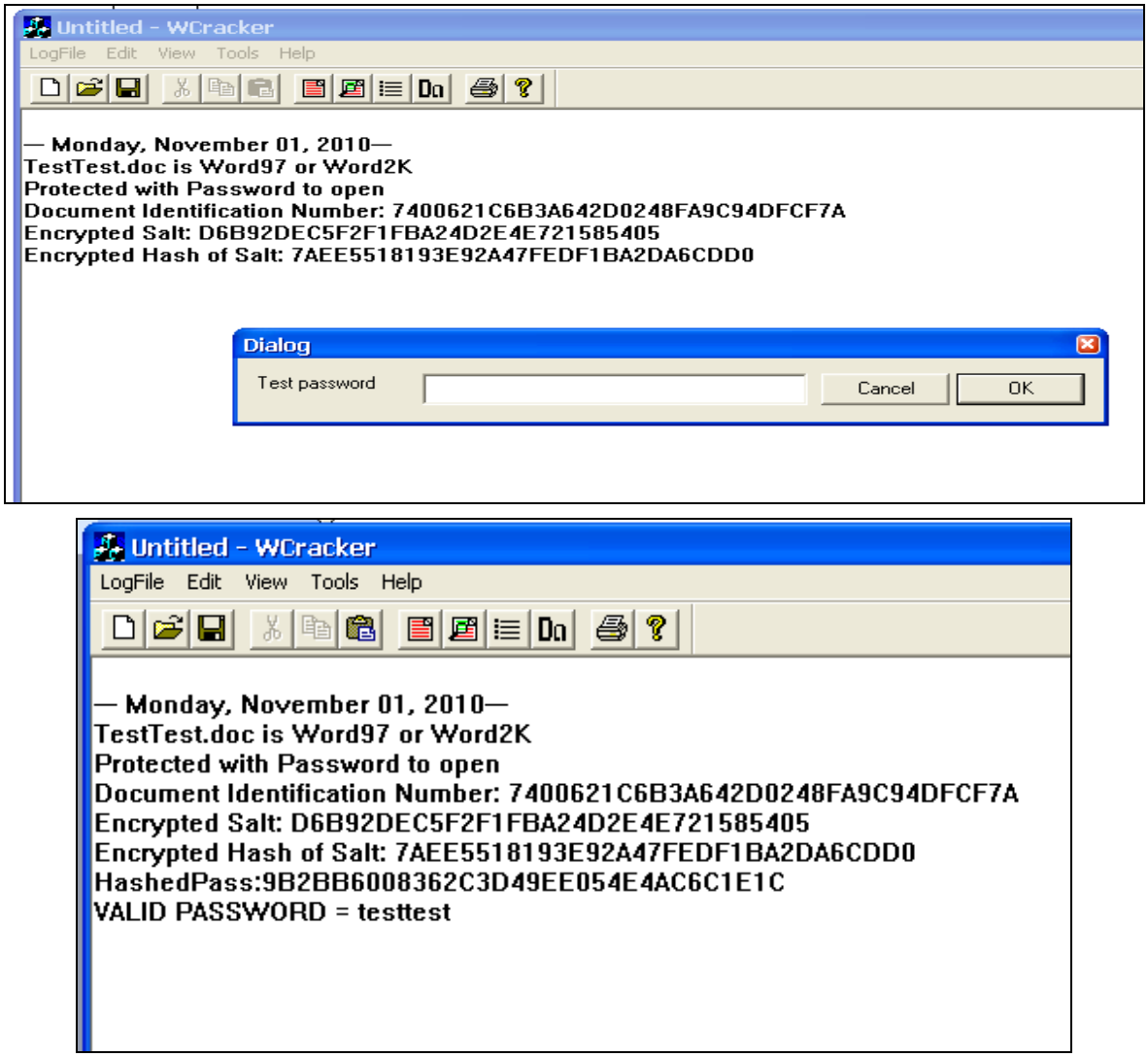
Để thử nghiệm phương thức lấy thông tin mã hóa để xác thực mật khẩu, xác định các hàm thư viện mật mã RC4 và lưu đồ xác thực mật khẩu MS-Word 2007 đã nghiên cứu, phần mềm Wcracker được mở rộng chức năng kiểm tra mật khẩu. Chức năng được đưa lên ToolBar của Wcracker như hình dưới đây.



Hình 3.1: Thử nghiệm 1 với văn bản MS-Word 2007

Người sử dụng được mời nhập tên tệp văn bản cần xác nhận mật khẩu. Khi Wcracker xác nhận đây là tệp văn bản MS-Word 2007 có cài đặt mật khẩu khi mở văn bản (Password to Open), hộp thoại yêu cầu nhập mật khẩu được hiện và nhận mật khẩu mà người sử dụng đưa ra.

Với mật khẩu đúng, Wcracker sẽ xác nhận mật khẩu hợp lệ (VALID PASSWORD), thông báo mật khẩu dạng rõ và giá trị HASH của mật khẩu. Khóa RC4 là 5 byte đầu của giá trị HASHPASS.



Hình 3.2: Thử nghiệm 2 với văn bản MS-Word 2007

Các hình trên là thử nghiệm với văn bản MS-Word 2007 có tên TestTest.doc. Đây là tệp văn bản có cài đặt mật khẩu. Các giá trị tham gia quá trình xác thực mật khẩu gồm:

DOCID = 7400621C6B3A642D0248FA9C94DFCF7A (Hexa)
SALT = D6B92DEC5F2F1FBA24D2E4E721585405 (Hexa, mã hóa)
HASHSALT = 7AEE5518193E92A47FEDF1BA2DA6CDD0 (Hexa, mã hóa)

Với mật khẩu đúng "testtest" mà người sử dụng đưa ra, Wcracker xác nhận mật khẩu đúng:

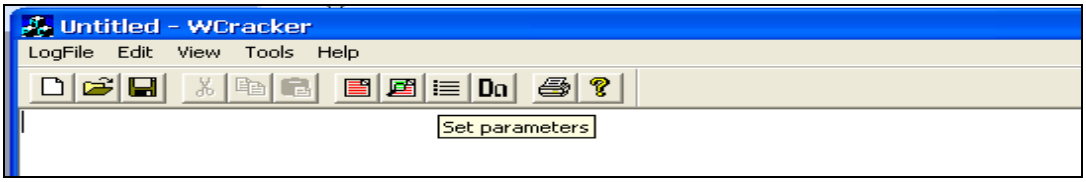
HashedPass:9B2BB6008362C3D49EE054E4AC6C1E1C (Hexa)
VALID PASSWORD = testtest

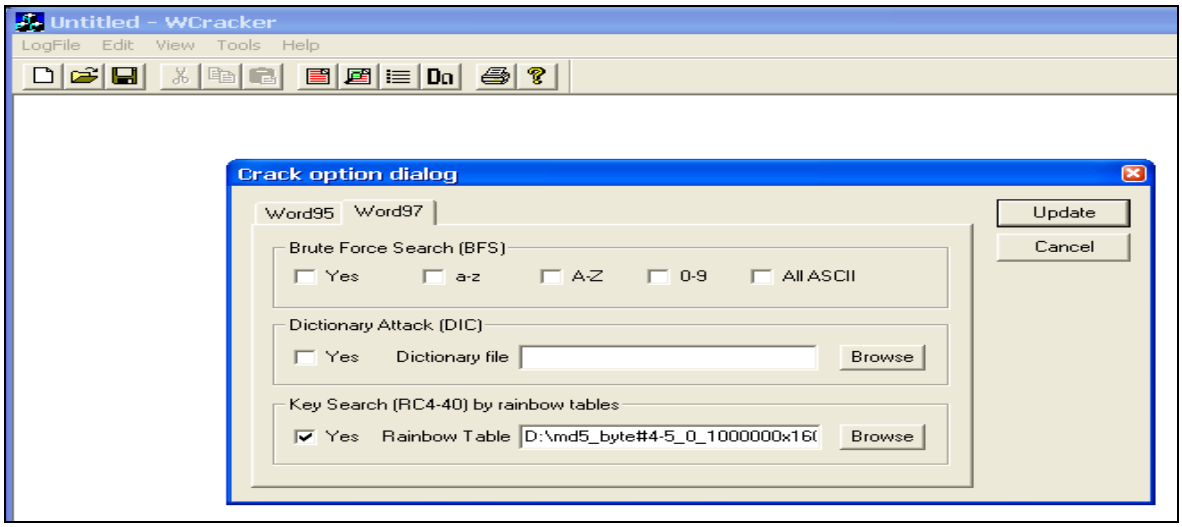
Như vậy các hàm thư viện RC4, lưu đồ thuật toán xác thực mật khẩu MS-Word 2007 mà Đề tài đã nghiên cứu cài đặt chính xác. Giá trị HASHPASS còn được sử dụng để so sánh với kết quả tấn công tìm khóa đúng RC4 sẽ mô tả dưới đây.

Kiểm tra mật khẩu là một bước nghiên cứu quan trọng nhằm đảm bảo lưu đồ xác thực mật khẩu và các hàm thư viện mật mã, hàm băm hoạt động đúng.

3.1.2 Chức năng thiết lập tham số tấn công

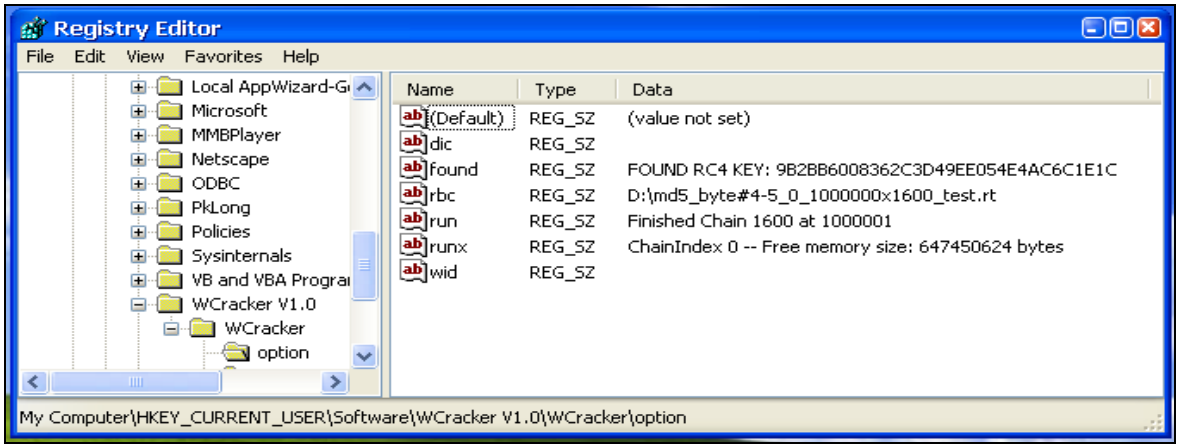
Tấn công mật khẩu MS-Word 2007 sử dụng các bảng Rainbow yêu cầu cài đặt các tham số mới cho Wcracker. Đó là tên bảng Rainbow mà giải thuật tấn công cần sử dụng. Tính năng cài đặt tham số của Wcracker được mở rộng như ở các hình dưới đây.





Hình 3.3: Tính năng cài đặt tham số của Wcracker

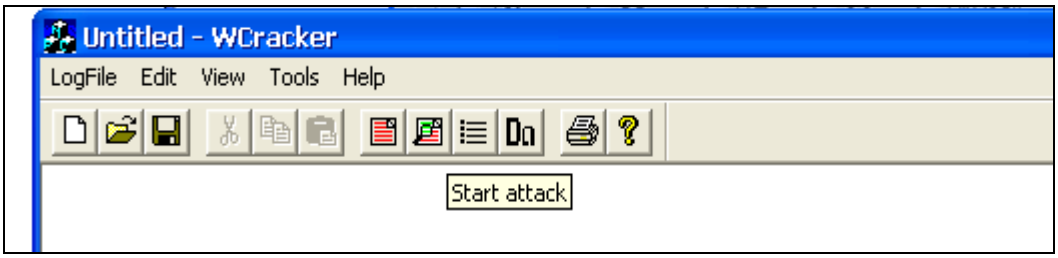
Các tham số sẽ được Wcracker lưu trữ trong registry của hệ thống để chỉ cần thiết lập 1 lần cho các lần tấn công với cùng một bảng Rainbow.



Hình 3.4: Các tham số được Wcracker lưu trữ trong registry

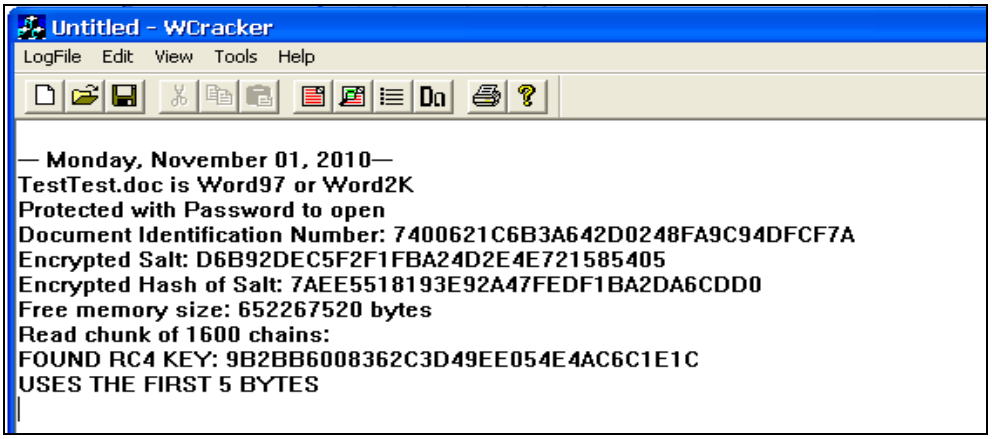
3.1.3 Chức năng tấn công tìm khóa RC4

Tấn công tìm khóa đúng của RC4 được xây dựng theo phương thức lợi dụng tính có "sắp đặt" của các chuỗi Rainbow MD5. Các giá trị MD5 trong mỗi chuỗi được truy vấn và được sử dụng làm khóa của RC4 trong lưu đồ xác thực mật khẩu. Với khóa đúng, lưu đồ sẽ xác nhận giá trị băm của mật khẩu lưu trữ và giá trị băm tính toán được. Đến khi đó, việc truy vấn các chuỗi Rainbow dừng lại. Khóa đúng được thông báo cho người sử dụng.



Hình 3.5: Tấn công tìm khóa đúng của RC4

Cần lưu ý, Đề tài không đặt mục tiêu giải mã tệp văn bản với khóa RC4 đúng. Chúng ta đã có phần mềm thực hiện thao tác này trong thực tế. Việc xây dựng thêm chức năng giải mã cũng không phải là khó khăn với thư viện WV, và gói mã nguồn của Abi-Word.



Hình 3.6: Kết quả thử nghiệm tấn công với tệp TestTest.doc

Hình trên là kết quả thử nghiệm tấn công với tệp TestTest.doc (có mật khẩu là testtest). Thời gian thực hiện tấn công còn lớn, chưa đáp ứng được yêu cầu hiệu quả mà Đề tài đặt ra.

3.1.4 Cài đặt chương trình

Sau đây là các cài đặt đối với Master, Slave và chương trình chính TMTO. Môi trường lập trình trên Linux, sử dụng truyền thông điệp LAM/MPI.

+) Master

//

```
// Master.h       -- Header file 2 of TMTO Project
// Created: 16-10-2014          -- @dang thanh cong
// Please put all data types, prototypes ref in Master.c here
////////////////////////////////////////////////////////////////
#ifndef TDH_TMTO_MASTER_H
#define TDH_TMTO_MASTER_H
  #include <sys/time.h>
  #include <sys/timeb.h>

  #include "tmto.h"

  void master_init (void);
  void master_finalize (void);

  void getCmdInputs (void);
```

```
    void getCmdFile (FILE *p);

    void getTimeDiff (char *s, struct timeb *sb, struct timeb *st);

    void get_input (void);
    void do_master (void);
    void work_in_parallel (void);

    unit_of_work_t *get_next_work_item (size_t *count);

    BYTE *put_mem (size_t *count, unit_of_work_t *gwork);
    void process_results (size_t, BYTE *mem, int id);

#endif //TDH_TMTO_MASTER_H
```

+) Slave

```
////////////////////////////////////////////////////////////////
// Slave.h         -- Header file 3 of TMTO Project
// Created: 16-10-2014          -- @ dang thanh cong
// Please put all data types, prototypes ref in Master.c here
////////////////////////////////////////////////////////////////
#ifndef TDH_TMTO_SLAVE_H
#define TDH_TMTO_SLAVE_H
#include "tmto.h"
size_t get_mem (BYTE *mem, unit_of_work_t *work);
void slave_init (void);
void slave_finalize (void);
BYTE do_compare (BYTE *block1, BYTE *block2, size_t);
void do_slave (void);
int key_searchEX (unit_of_work_t *work);

#endif//TDH_TMTO_SLAVE_H
```

+) TMTO

```
//////////////////////////////////////////////////////////////
// TMTO.h       -- Header file 1 of TMTO Project
// Created: 16-10-2014          -- @ dang thanh cong
// Please put all data types, prototypes ref in TMTO Proj here
//////////////////////////////////////////////////////////////
#ifndef TDH_TMTO_TMTO_H
#define TDH_TMTO_TMTO_H

#include <stdio.h>
#include "mpi.h"

#define WORKTAG 1
#define DIEDTAG 2
#define DONETAG 3
#define IDLETAG 4
//#define KEY_BLOCK 1048576    // = 2^20
//#define KEY_BLOCK 4194304    // = 2^22
 #define KEY_BLOCK 67108864  // = 2^26
#define MAX_KEY 1099511627776         // = 2^40
typedef unsigned char U8;
typedef unsigned long int  U32;
typedef unsigned long long int U64;
typedef unsigned char BYTE;

typedef struct {
  U32 crank;                    //node ID
  U64 start_key;                //begin point of RC4 key range
  U64 end_key;                  //end point of RC4 key range
  U8 tmp_salt[64];     //Salt for comparison
  U8 tmp_hsalt[16];     //Encrypted hash salt for comparison
} unit_of_work_t;
#endif //TDH_TMTO_TMTO_H
```

+) Mã nguồn TMTO

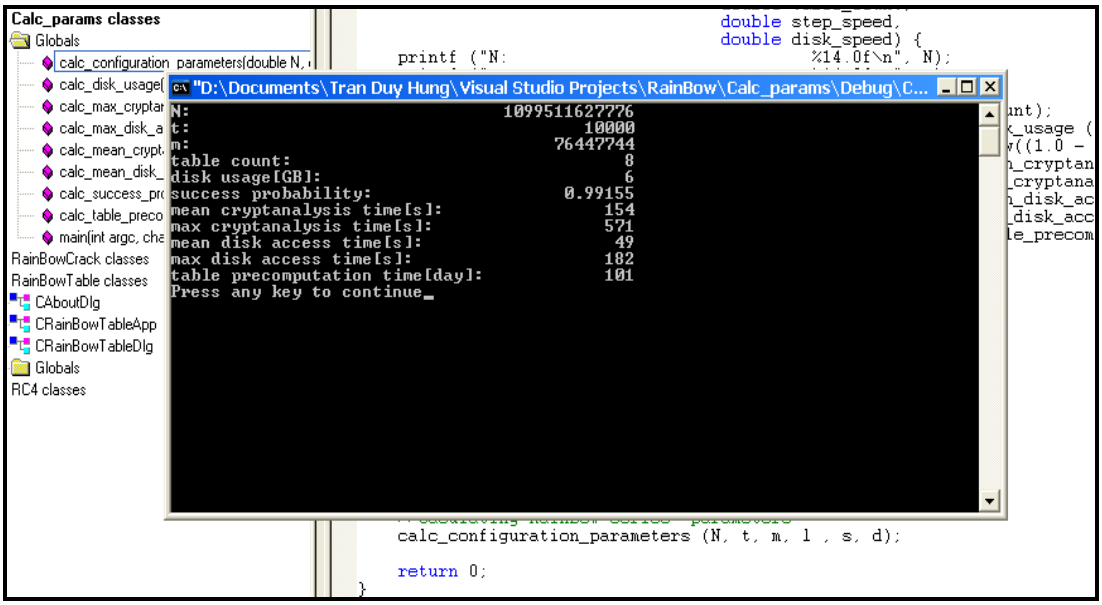
```
#include <stdio.h>
#include "mpi.h"        //Refs to LAM/MPI communicator
#include "tmto.h"       //Refs to TMTO header file
#include "master.h"     //Refs to TMTO master functions
#include "slave.h"      //Refs to TMTO slave functions
int myrank;
int main (int argc, char ** argv) {
  /* Initialize MPI */
  MPI_Init (&argc, &argv);
  /* Find out my identification in the communicator */
  MPI_Comm_rank (MPI_COMM_WORLD, &myrank);
  if (myrank == 0) {
        do_master ();
  } else {
        do_slave ();
  }
  /* Shutdown MPI communicator */
  MPI_Finalize ();
  return (0);
}
```

3.2 Lựa chọn tham số Rainbow để tấn công RC4

Theo một kết quả nghiên cứu đã công bố, với máy tính có bộ vi xử lý Pentium 4, tần số xung nhịp 1.5GHz, bộ nhớ trong 500MB SDRAM (tương đương với máy tính xách tay loại trung bình), tốc độ tính toán cho một mắt xích của chuỗi Rainbow khoảng 700.000/s; ổ địa cứng hiện tại truy vấn dữ liệu với tốc độ 610 truy vấn trong 19 giây; không gian khóa của RC4 là 256^5.

Lựa chọn các tham số: số bảng Rainbow (*l*) là 8; mỗi bảng gồm các chuỗi có độ dài (*t*) là 10.000; số chuỗi trong mỗi bảng Rainbow (*m*) là: 76.447.744.

Kết quả tính toán dựa trên các tham số trên như sau:



Hình 3.7: Lựa chọn tham số Rainbow để tấn công RC4

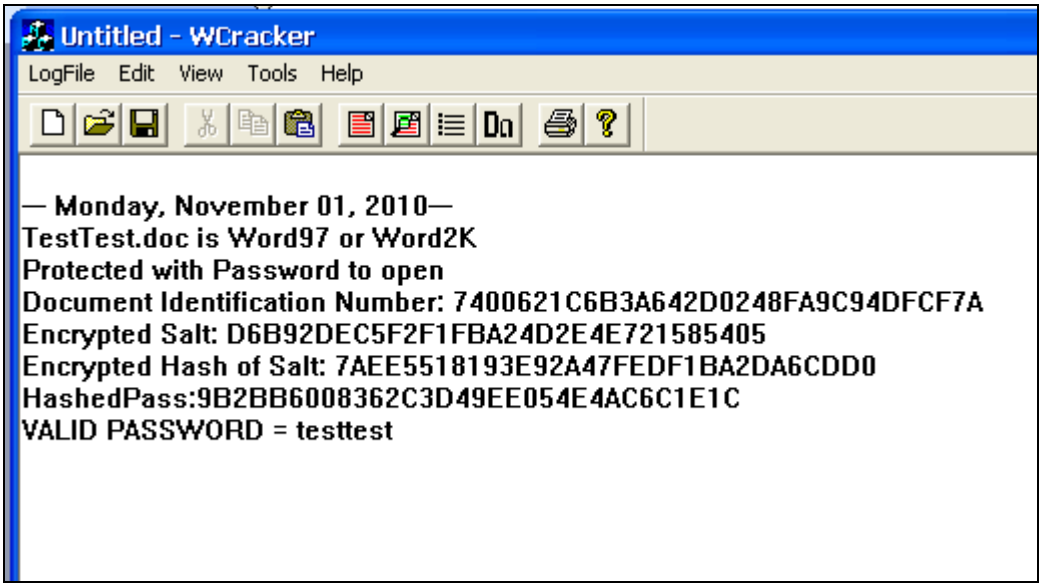
Với các tham số đã lựa chọn, tấn công RC4-40bit khi biết 2 trong 3 tham số thuật toán đạt tỷ lệ thành công là 99% với thời gian trung bình là 154 giây. Thời gian tính toán trước là trên 100 ngày.

3.3 Xây dựng bảng Rainbow

Sử dụng các chương trình phần mềm trong gói phần mềm RainbowCrack, Đề tài đã xây dựng bảng Rainbow cho MD5 với các tham số m = 1.000.000, t=1.600. Bảng Rainbow đã sắp xếp được lưu ở tệp "md5_byte#4-5_0_1000000x1600_test.rt". Tệp này được sử dụng trong Wcracker cho chức năng tấn công tìm khóa RC4.

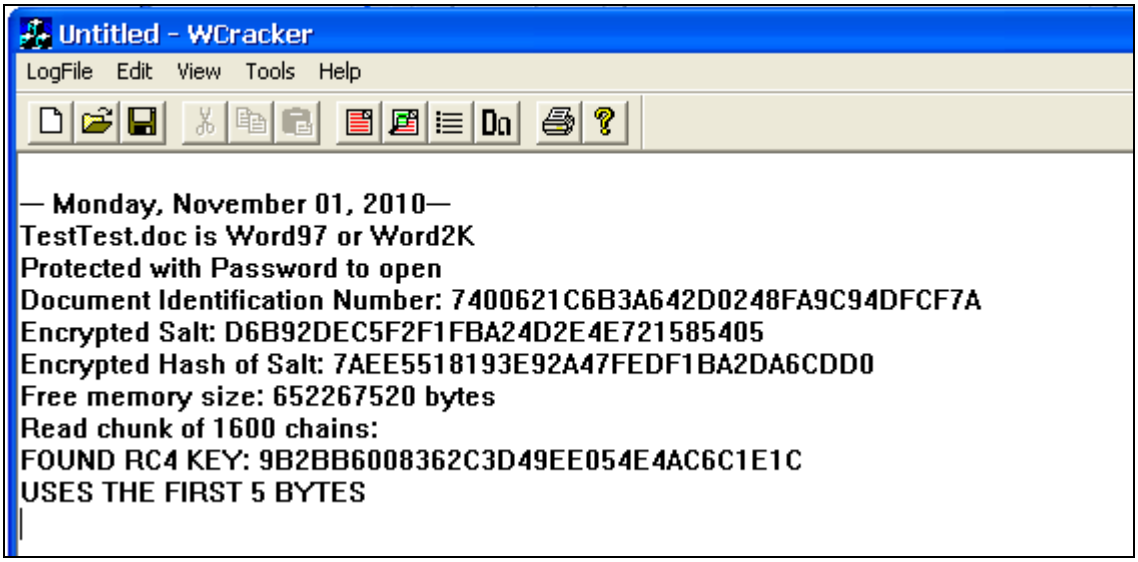
3.4 Thử nghiệm các tính năng mở rộng của Wcracker

Hình dưới đây là kết quả thử nghiệm chức năng kiểm tra mật khẩu của Wcracker khi người sử dụng cho biết mật khẩu đúng.



Hình 3.8: Kết quả thử nghiệm chức năng kiểm tra mật khẩu của Wcracke.

Hình 3.9 dưới đây cho biết kết quả của chức năng tấn công tìm khóa RC4 khi tìm được khóa đúng từ các chuỗi của bảng Rainbow.



Hình 3.9: Kết quả của chức năng tấn công tìm khóa RC4

3.5 Kết quả phân tích khóa bằng phần mềm xử lý song song

Thử nghiệm với tệp Test.doc, không biết mật khẩu để mở tệp. Chương trình tính toán song song chạy trên bộ máy tính 20Gflop với cấu hình 64 node tính toán. Kết quả đã tìm thấy khóa đúng của RC4 sau hơn 2 ngày thực hiện.

```
WORK:10268:689006706688:689073815552
WORK:10269:689073815552:689140924416
WORK:10270:689140924416:689208033280
WORK:10271:689208033280:689275142144
WORK:10272:689275142144:689342251008
WORK:10273:689342251008:689409359872
WORK:10274:689409359872:689476468736
WORK:10275:689476468736:689543577600
WORK:10276:689543577600:689610686464
WORK:10277:689610686464:689677795328
WORK:10278:689677795328:689744904192
WORK:10279:689744904192:689812013056
WORK:10280:689812013056:689879121920
WORK:10281:689879121920:689946230784
WORK:10282:689946230784:690013339648
WORK:10283:690013339648:690080448512
WORK:10284:690080448512:690147557376
WORK:10285:690147557376:690214666240
WORK:10286:690214666240:690281775104
FOUND RC4 encryption key:DAEECACB9F  node: 49
0001d:37h:30m:39.015s
```

KẾT LUẬN

Đề tài đã thực hiện các nghiên cứu về:

1.Giải thuật xác thực mật khẩu bảo mật văn bản trong MS- Word sử dụng thuật toán mã hóa RC4. Giải thuật xác thực mật ứng dụng trong MS-Word là phức tạp có kết hợp các yếu tố ngẫu nhiên (bao gồm DOCID và SALT), tăng cường khả năng chống lại các tấn công dựa trên từ điển mã hóa (ECB-Electronic Code Book). Mỗi văn bản soạn thảo trên MS-Word được gắn với 2 giá trị ngẫu nhiên xác định văn bản (Document Identification Number- DOCID) và muối mật khẩu (SALT), vì vậy các văn bản có cùng mật khẩu bảo mật nhưng sẽ có các giá trị mã hóa lưu trữ trên tệp văn bản khác nhau. Tấn công TMTO dựa trên tính toán trước để giảm thời gian tấn công là một dạng từ điển mã nên gặp khó khăn.

2. Gói phần mềm Rainbow-crack của Rainbow-crack Project. Đây là gói phần mềm tính toán sẵn, sau đó sắp xếp các bảng lưu trữ các giá trị tính toán sẵn thành các chuỗi phân lập, được đặt tên là chuỗi cầu vồng (Rainbow) gần với tán sắc ánh sáng của hiện tượng tự nhiên này. Không gian mã của thuật toán mật mã lựa chọn được phân lập thành các chuỗi liên kết lại với nhau, tương tự như các thành phần ánh sáng cùng màu tạo thành các dải màu của cầu vồng. Với kết cấu như vậy, chỉ cần lưu trữ điểm đầu và điểm cuối của mỗi chuỗi cầu vồng và xác định thuật toán xây dựng lại các giá trị trung gian là đủ để lưu trữ từ điển mã của thuật toán mật mã. Tuy nhiên nếu muốn lưu trữ toàn bộ không gian mật mã (xác suất tấn công đạt 100%) thì số số lượng chuỗi cầu vồng cũng tăng. Philippe Oechslin để tính toán các tham số trong xây dựng các bảng rainbow cho RC4 với mã khóa 40 bít. Phần mềm tính toán tham số đã được xây dựng , các tham số cũng đã được xác định : t=10.000; m= 76.447.744; l= 8; thời gian tính toán trước là 101ngày, xác suất đạt 99%. Nghiên cứu Rainbow-crack đã xác định chương trình tấn công mật khẩu vào hệ thống Windows (Lmpass, Nthansh, MD5hash..) ứng dụng hiệu quả trong xác định mật khẩu truy nhập hệ thống Windows khi có giá trị mã mật khẩu.

Tuy nhiên, kết quả nghiên cứu về giải thuật sử dụng RC4 ứng dụng cụ thể trong MS-Word để xác thực mật khẩu bảo mật không thuận lợi cho tính toán áp dụng TMTO, nên đề tài không tiến hành xây dựng các bảng Rainbow cho RC4.

Thay vào đó, Đề tài lựa chọn hướng triển khai áp dụng TMTO thông qua các bảng MD5 trực tiếp xây dựng khóa RC4 để tấn công tìm khóa mã đúng cho mỗi văn bản được bảo mật bằng mật khẩu. Chương trình phần mềm chạy trên hệ điều hành Windows và chương trình tính toán song song trên hệ điều hành Linux LAM/MPI đã được xây dựng và thử nghiệm tấn công. Chương trình phần mềm tính toán song song đã cho kết quả tìm được khóa mã RC4 đúng để giải mã văn bản.

Để thực hiện tấn công thử nghiệm, Đề tài đã sử dụng RainbowCrack để xây dựng bảng rainbow MD5 cho mật khẩu có độ dài 4 đến 5 byte trên tập 356 giá trị của byte. Bảng Rainbow được lựa chọn kích thước gồm 1.000.000 chuỗi với độ dài mỗi chuỗi 16.000 là phần tử.

Chương trình phần mềm chạy trên hệ điều hành Windows và các chương trình tính toán song song trên hệ điều hành Linux LAM/MPI đã được xây dựng và thử nghiệm tấn công. Chương trình phần mềm tính toán song song đã cho kết quả tìm được khóa mã RC4 đúng để giải mã văn bản.

TÀI LIỆU THAM KHẢO

Phần này liệt kê một số phần mềm chính mà Đề tài đã sử dụng để cài đặt các chương trình phần mềm trong nội dung nghiên cứu đã đề ra.

[1]McKenzie, Patrick (9 tháng 4 năm 2014). "What Heartbleed Can Teach The OSS Community About Marketing" (bằng tiếng Anh). Truy cập ngày 10 tháng 4 năm 2014.

[2] Biggs, John (9 tháng 4 năm 2014). "Heartbleed, The First Security Bug With A Cool Logo". *TechCrunch* (bằng tiếng Anh). Truy cập ngày 10 tháng 4 năm 2014.

[3]Goodin, Dan (8 tháng 4 năm 2014). "Critical crypto bug in OpenSSL opens two-thirds of the Web to eavesdropping". *Ars Technica* (bằng tiếng Anh). Truy cập ngày 8 tháng 4 năm 2014.

[4] Seggelmann, R.; Tuexen, M.; Williams, M. (tháng 2 năm 2012). "RFC 6520: Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension" (bằng tiếng Anh). Internet Engineering Task Force. Truy cập ngày 8 tháng 4 năm 2014.

[5] Mutton, Paul (8 tháng 4 năm 2014). "Half a million widely trusted websites vulnerable to Heartbleed bug" (bằng tiếng Anh). Netcraft. Truy cập ngày 8 tháng 4 năm 2014.

[6] Thư viện RainbowCrack của Zhu Shuanglei@hotmail.comcài đặt phương pháp tấn công TMTO do Philippe Oechslin đề xuất năm 1994.

[7] Martin Hellman, IEEE Transaction on information theory vol IT -26, No 4, July 7/1980, *A cryptanalytic time –memory trade-off*

[8] D.R. Stinson, Crytography: Theory and Practice, 1995 by CRC Press. Inc

[9] Philippe Oechslin, Lecture notes in computer science volume 2729, 2003, phương pháp617-630, *Making a faster A cryptanalytic time –memory trade-off*

[10] Man Young Rhee, Wilay, *Internet Security - Cryptographic Principles, Algorithms and Protocols,* 2003.

[11] William Stallings, *Network Security Essentials: Applications and Standards*, Prentice Hall, New Jersey, 1999.

[12] William Stallings, *Network Security Essentials*, 2009.

[13] Wasim.E. Rajput. *Commerce Systems-Architecture & Application*, 2013.

[14] Douglas R. Stinson, *Cryptography Theory and Practice*, University of Nebraska-Lincoln

[15] Leong Yu Kiang. *Living with Mathematics*. 3rd Edition. 2011. McGraw-Hill Education (Asia), Singapore. ISBN 978-007-132677-3

[16] G. Paul, S. Rathi and S.Maitra, "Non-negligible Bias of the First Output Byte of RC4 towards the First Three Bytes of the Secret Key", Proceedings of the International Workshop on Coding and Cryptography (WCC) 2007, pp. 285-294 and Designs, Codes and Cryptography Journal, pp. 123-134, vol. 49, no. 1-3, December 2008 .