Chương trình KC-01:
Nghiên cứu khoa học
phát triển công nghệ thông tin
và truyền thông

Đề tài KC-01-01:
Nghiên cứu một số vấn đề bảo mật và an toàn thông tin cho các mạng dùng giao thức liên mạng máy tính IP

PHỤ LỤC: MỘT SỐ NGHIÊN CỨU VỀ HÀM BĂM VÀ GIAO THỨC MẬT MÃ

HÀ NỘI-2004

MỤC LỤC

NGHIÊN CỨU VỀ THÁM MÃ MD4

Trần Hồng Thái

I. MÔ TẢ THUẬT TOÁN MD4

Thuật toán MD4 lấy một đầu vào là một message có độ dài bất kỳ và đầu ra là một "tóm lược thông báo" (message digest), còn được gọi là "fingerprint". Nó được thiết kế khá gọn và nhanh trên máy 32-bit.

1. Mô tả thuật toán

Giả sử chúng ta có một message với độ dài là b-bit là đầu vào và chúng ta muốn tìm tóm lược thông báo (message digest) của nó. Ở đây b là một số nguyên dương bất kỳ, có thể bằng 0, hoặc lớn bất kỳ.

Ở đây chúng ta sử dụng các thuật ngữ sau: 'word' là biến 32 bit, byte là 8 bit. Quá trình tính tóm lược thông báo này được thực hiện qua 5 bước sau:

Bước 1: Thêm vào các bit đệm (Padding bits)
Thông điệp được mở rộng bằng việc thêm các "bit đệm" sao cho độ dài (theo bit) của nó đồng dư với 448 (modulo 512). Việc này sẽ đảm bảo khi thêm 64 bit (độ dài - được trình bày sau) nữa, thì độ dài thông điệp là bội của 512.

Việc đệm được thực hiện như sau: một bit '1' được nối với thông điệp và sau đó là các bit '0' cho đến khi độ dài thông điệp đồng dư với 448.

Bước 2: Thêm độ dài thông điệp (Length)
Độ dài thông điệp trước khi mở rộng b được biểu diễn bởi một số 64 bit (gồm 2 word) sẽ được thêm vào thông điệp sau bước 1 (theo thứ tự word thấp trước). Lúc này độ dài thông điệp là bội của 512 bit (16 word 32 bit). Ký hiệu message kết quả là M[0 … N-1], N là bội của 512.

Bước 3: Khởi tạo bộ đệm MD
Bốn biến A, B, C, D (là các thanh ghi 32 bit) được dùng để tính "message digest". Chúng được khởi tạo bằng các giá trị sau ở dạng Hexa, theo thứ tự các byte thấp trước:

word A: 01 23 45 67
word B: 89 ab cd ef
word C: fe dc ba 98
word D: 76 54 32 10

Bước 4: Xử lý thông điệp (message) theo từng khối 16-words

Trước tiên ta định nghĩa 3 hàm, mỗi hàm lấy 3 word (32 bit) làm đầu vào và đưa ra một từ 32 bit:

F(X,Y,Z) = XY v not(X)Z
G(X,Y,Z) = XY v XZ v YZ
H(X,Y,Z) = X xor Y xor Z

Trong đó: XY là phép 'and' bit của X và Y, not(X) là phép lấy bù bit của X, X xor Y là phép cộng modulo 2 theo bit của X và Y, X v Y là phép toán OR (hoặc) bit của X và Y.

Thông điệp được xử lý theo từng khối 16 word như sau:

```
for i = 0 to N/16-1 do

 /* Copy block i into X. */
 for j = 0 to 15 do
      set X[j] to M[i*16 +j]
 end

 /* Save A as AA, B as BB, C as CC, D as DD */
 AA = A;
 BB = B;
 CC = C;
 DD = D;

 /* Round 1 */
 /* Ký hiệu [abcd k s] là phép toán sau:
       a = (a + F(b,c,d) + X[k]) <<< s
 Thực hiện 16 lần như sau. */

 [ABCD  0  3]; [DABC  1  7]; [CDAB  2  11]; [BCDA  3  19];
 [ABCD  4  3]; [DABC  5  7]; [CDAB  6  11]; [BCDA  7  19];
 [ABCD  8  3]; [DABC  9  7]; [CDAB 10  11]; [BCDA 11  19];
 [ABCD 12  3]; [DABC 13  7]; [CDAB 14  11]; [BCDA 15  19];

 /* Round 2 */
 /* Ký hiệu [abcd k s] là phép toán sau:
       a = (a + G(b,c,d) + X[k] + 5A82799) <<< s */

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

 /* Round 3 */
 /* Ký hiệu [abcd k s] là phép toán sau:
       a = (a + H(b,c,d) + X[k] + 6ed9eba1) <<< s */

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

```
 A = A + AA;
 B = B + BB;
 C = C + CC;
 D = D + DD;
```
end; */* of loop on i */*

Bước 5: Kết quả đầu ra (output)
Bản tóm lược thông điệp (message digest) là nội dung các thanh ghi A, B, C, D theo thứ tự từ byte thấp của A đến byte cao của D.

2. Mã nguồn của MD4

Chúng tôi đã download mã nguồn của thuật toán mã hoá MD4 trên Internet. Chương trình khá nhỏ gọn, gồm 3 file là global.h, md4.h (2 file header khai báo các PROTOTYPE, hằng) và md4c.c - mã nguồn của MD4. Chúng tôi đã đọc hiểu mã nguồn của MD4 và thử nghiệm nhỏ như sau: thực hiện băm một xâu có độ dài nhỏ hơn 448 bit 1000000 lần trên máy Dell 350MHz. Thời gian tính toán này là

II. THUẬT TOÁN THÁM MD4

Tác giả của thuật toán thám MD4 là Hans Dobbertin với bài "Cryptanalysis of MD4" - năm 1997. Thuật toán MD4 được Rivest đề nghị năm 1990 và 2 năm sau là RIPEMD được thiết kế mạnh hơn MD4. Năm 1995, tác giả đã tìm ra một tấn công chống lại 2 vòng của RIPEMD. Phương pháp này được bổ sung và có thể sử dụng cho tấn công đủ 3 vòng của MD4.

Theo tác giả thì thuật toán này có thể tìm ra "va chạm" (collision) cho MD4 chỉ trong một vài giây trên một máy PC. Đặc biệt tác giả đã đưa ra một ví dụ rất cụ thể có tính thực hành và thuyết phục cao bằng việc tìm ra va chạm của thông điệp có ý nghĩa. Kết quả chính của bài báo này là khẳng định "MD4 là không phải hàm hash không va chạm".

Thuật toán này là kiểu tấn công tìm collision: tức là biết giá trị khởi đầu IV_o, tìm các thông điệp X và X' sao cho hash(IV_o, X) = hash(IV_o, X').

1. Tóm tắt thuật toán do Dobbertin trình bày.

1.1 Ký hiệu và qui ước sử dụng cho thuật toán

Tất cả các biến và hằng số được sử dụng đều là các số 32 bit. Các số hạng được biểu diễn theo modulo 2^{32}. Các ký hiệu ^, v, $\oplus$ và $\neg$ lần lượt là các phép toán AND, OR, XOR và lấy phần bù theo bit. Với một từ W - 32 bit, $W^{<<32}$ ký hiệu của phép dịch vòng trái W đi s vị trí (với $0 \le s \le 32$). Và $-W^{<<s}$ viết tắt cho $-(W^{<<s})$.

Ký hiệu $X=(X_i)$, i<16 là toàn bộ 16 words (512 bits) và $\tilde{X} = (\tilde{X}_i)_{i<16}$ được thiết lập như sau:

$\tilde{X}_i = X_i$, với i ≠ 12.

$\tilde{X}_{12} = X_{12} + 1$

Việc chọn $\tilde{X}_{12} = X_{12} + 1$ là vì X_{12} xuất hiện ở vòng 1 và vòng 2 với khoảng cách ngắn nhất so với các X_i khác.

Bài toán đặt ra là: làm thế nào để tìm X sao cho giá trị băm MD4 của X, $\tilde{X}$ trùng nhau, nghĩa là compress(IV_o; X) = compress(IV_o; $\tilde{X}$).

Tấn công được chia thành 3 phần, mỗi phần ta chỉ xét một đoạn của hàm nén. Với n < m < 48, có công thức sau:

$$\text{compress}_m^n((A,B,C,D);X_{\Psi(n)}...X_{\Psi(m)}) = (A',B',C',D')$$

là đoạn của hàm nén từ bước thứ n tới bước m, mà ψ(i) là ánh xạ sao cho giá trị của $X_{\psi(i)}$ được sử dụng ở bước thứ i của hàm nén. Nghĩa là tính compress_m^n với giá trị ban đầu (A, B, C, D) và từ bước thứ n đến m, tương ứng sử dụng các từ đầu vào $X_{\psi(n)}$ … $X_{\psi(m)}$, và kết quả ra là (A', B', C', D') là nội dung của 4 thanh ghi sau bước m. Đôi khi để đơn giản chúng ta viết X thay cho $X_{\psi(n)}$ … $X_{\psi(m)}$. Một dạng công thức khác cũng được sử dụng:

(A_i, B_i, C_i, D_i) là nội dung của các thanh ghi sau bước thứ i.

Thiết lập:

$$\Delta_i = (A_i - \tilde{A}_i, B_i - \tilde{B}_i, C_i - \tilde{C}_i, D_i - \tilde{D}_i)$$

Chú ý rằng mỗi bước chỉ có nội dung một thanh ghi bị thay đổi.

1.2 Tìm Inner Almost-Collision (các bước từ 12 - 19 của MD4) (tạm dịch là "hầu va chạm bên trong")

Chú ý rằng X_{12} xuất hiện 1 lần trong mỗi vòng là trong các bước 12, 19, 35. X và $\tilde{X}$ có va chạm nếu và chỉ nếu $\Delta_{35} = 0$, bởi X_{12} xuất hiện trong bước 35 lần cuối cùng. Để điều này xảy ra chúng ta cần chọn các giá trị sao cho

$$\Delta_{19} = (0, 1^{<<25}, -1^{<<5}, 0) \qquad (*)$$

Điều này có nghĩa là đầu ra của $\text{compress}_{19}^{12}$ cho X và $\tilde{X}$ là gần bằng nhau. Lý do để chọn giá trị này sẽ được trình bày rõ ràng hơn trong phần tiếp theo. Gọi (A, B, C, D) là giá trị khởi đầu của $\text{compress}_{19}^{12}$.

Để đơn giản, ta sử dụng các ký hiệu sau: $A_* = A_{19}$, $B_* = B_{19}$, …, $U = A_{12}$, $V=D_{13}$, $W=C_{14}$, $Z = B_{15}$, $\tilde{U} = \tilde{A}_{12}$, $\tilde{V} = \tilde{D}_{13}$, $\tilde{W} = \tilde{C}_{14}$, $\tilde{Z} = \tilde{B}_{15}$. K1=0x5a82799 là hằng số sử dụng trong vòng 2 của MD4.

Ở đây chúng ta cần $\tilde{B}_* + 1^{<<25} = B_*$, $C_* + 1^{<<5} = \tilde{C}_*$. Lúc này việc tìm inner almost collision tương đương với việc giải hệ các phương trình sau:

$$1 = \tilde{U}^{<<29} - U^{<<29} \quad (1)$$

$$F(\tilde{U}, B, C) - F(U, B, C) = \tilde{V}^{<<25} - V^{<<25} \quad (2)$$

$$F(\tilde{V},\tilde{U},B) - F(V,U,B) = \tilde{W}^{<<21} - W^{<<21} \quad (3)$$

$$F(\tilde{W},\tilde{V},\tilde{U}) - F(W,V,U) = \tilde{Z}^{<<13} - Z^{<<13} \quad (4)$$

$$G(\tilde{Z},\tilde{W},\tilde{V}) - G(Z,W,V) = U - \tilde{U} \quad (5)$$

$$G(A_*,\tilde{Z},\tilde{W}) - G(A_*,Z,W) = V - \tilde{V} \quad (6)$$

$$G(D_*,A_*,\tilde{Z}) - G(D_*,A_*,Z) = W - \tilde{W} + \tilde{C}_*^{<<23} - C_*^{<<23} \quad (7)$$

$$G(\tilde{C}_*,D_*,A_*) - G(C_*,D_*,A_*) = Z - \tilde{Z} + \tilde{B}_*^{<<19} - B_*^{<<19} - 1 \quad (8)$$

Các biểu thức trên thu được từ việc khử các $X_{\psi(i)}$, với i=12, 13, … 19 của hàm nén $\text{compress}_{19}^{12}$. Ví dụ: từ định nghĩa trong bước 15, có

$$Z=(B + F(W,V,U) + X_{15})^{<<19}$$

$$\tilde{Z} = (B + F(\tilde{W},\tilde{V},\tilde{U}) + X_{15})^{<<19}$$

ta thu được biểu thức (4). Và cũng từ mỗi bước trong hàm nén này, ta thu được các biểu thức sau:

X_{13} = tùy ý (9)

$X_{14} = W^{<<21} - C - F(V,U,B)$ (10)

$X_{15} = Z^{<<13} - B - F(W,V,U)$ (11)

$X_0 = A_*^{<<29} - U - G(Z,W,V) - K1$ (12)

$X_4 = D_*^{<<27} - V - G(A_*, Z, W) - K1$ (13)

$X_8 = C_*^{<<23} - W - G(D_*, A_*, Z) - K1$ (14)

$X_{12} = B_*^{<<19} - Z - G(C_*, D_*, A_*) - X_{13}$ (15)

$D = V^{<<25} - F(U,B,C) - X_{13}$ (16)

$A = U^{<<29} - F(B,C,D) - X_{12}$ (17)

Thấy rằng hệ (1) đến (8) gồm có 14 biến, do đó ý tưởng là thiết lập một số biến sao cho có thể tìm được các biến còn lại. Do vậy, chọn:

$U = -1 = 0xffffffff$, $\tilde{U} = 0$, $B = 0$

Khi đó (1) đã thoả mãn. Các biểu thức (2), (3), (6), (7) và (8) được chuyển đổi và sắp xếp lại như sau:

$$\tilde{Z} = Z - G(\tilde{C}_*, D_*, A_*) + G(C_*, D_*, A_*) + \tilde{B}_*^{<<19} - B_*^{<<19} - 1 \quad (18)$$

$$\tilde{\mathrm{W}} = \mathrm{W} - G(D_*, A_*, \tilde{Z}) + G(D_*, A_*, Z) + \tilde{C}_*^{<<23} - C_*^{<<23} \quad (19)$$

$$V = \tilde{\mathrm{W}}^{<<21} - \mathrm{W}^{<<21} \quad (20)$$

$$\tilde{V} = V - G(A_*, \tilde{Z}, \tilde{W}) + G(A_*, Z, W) \quad (21)$$

$$\mathrm{C} = V^{<<25} - \tilde{V}^{<<25} \quad (22)$$

Trong hệ phương trình này A_*, B_*, C_*, D_*, Z và W là các tham số tự do để tìm các biến khác. Hai biểu thức (4) và (5) còn lại sẽ là:

$$G(Z, W, V) - G(\tilde{Z}, \tilde{W}, \tilde{V}) = 1 \quad (23)$$

$$F(\tilde{W}, \tilde{V}, 0) - F(W, V, -1) - \tilde{Z}^{<<13} + Z^{<<13} = 0 \quad (24)$$

Việc tìm hầu va chạm bên trong là tìm các giá trị A, B, C, D và X_{12}, X_{13}, X_{14}, X_{15}, X_0, X_4, X_8.

Bây giờ ta thu được 16 phương trình (*từ (9) đến (24)*) với 20 biến. Cụ thể là X_{12}, X_{13}, X_{14}, X_{15}, X_0, X_4, X_8 và A, C, D, W, Z, V, $\tilde{Z}, \tilde{W}, \tilde{V}$, A_*, B_*, C_*, D_*. Nhận thấy rằng trong các biểu thức từ (18) đến (21) nếu ta coi A_*, B_*, C_*, D_*, Z và W là các tham số tự do thì ta có thể tìm các biến khác thông qua chúng.

<u>Thuật toán tìm Inner Almost Collision thực hiện như sau:</u>

Bước 1:Chọn A_*, B_*, C_*, D_*, Z, W một cách ngẫu nhiên, tính $\tilde{Z}, \tilde{\mathrm{W}}, V, \tilde{V}$ theo các biểu thức (18) đến (21) và kiểm tra (test) biểu thức (23). Nếu test qua thì nhảy sang bước 2.

Bước 2: Lấy A_*, B_*, C_*, D_*, Z, W tìm được từ bước 1 làm các "giá trị cơ sở" (basic value). Thay đổi 1 bit ngẫu nhiên nào đó trong các biến trên, tính $\tilde{Z}, \tilde{\mathrm{W}}, V, \tilde{V}$ và test biểu thức (23). Nếu nó vẫn thoả mãn và 4 bit phải của biểu thức:

$$F(\tilde{W}, \tilde{V}, 0) - F(W, V, -1) - \tilde{Z}^{<<13} + Z^{<<13} \quad (25)$$

bằng 0 thì lấy các giá trị A_*, B_*, C_*, D_*, Z, W tương ứng làm "giá trị cơ sở" mới. Thực hiện giống như trên và test 8 bit phải của (25). "Giá trị cơ sở" mới được lấy chỉ khi 8 bit phải của (25) bằng 0. Thực hiện tiếp như vậy với 12, 16, … 32 bit phải của (25) bằng 0 (tức biểu thức (24) thoả mãn).

Bước 3: Bây giờ (23), (24) đã thoả mãn, ta đã thu được một "inner almost-collision" bằng việc chọn B=0 và tính các giá trị A, C, D và X_i (i=0, 4, 8, 12, 13, 14, 15) theo các biểu thức (9)-(17) và (22).

Để sử dụng "inner almost-collision" vào tấn công vi sai trong phần tới, thì cần phải thoả mãn thêm điều kiện nữa là:

$$G(B_*, C_*, D_*) = G(\tilde{B}_*, \tilde{C}_*, D_*) \qquad (26)$$

Khi B_* và $\tilde{B}_*$ (C_* và $\tilde{C}_*$ tương ứng) gần nhau, có một xác suất cao để va chạm này là đúng. Chúng ta gọi một inner almost-collision là chấp nhận được nếu (26) thoả mãn. Tác giả đã khẳng định rằng thuật toán này tìm ra một inner almost-collision dưới 1 giây trên máy tính PC.

<u>Bổ đề 1:</u> *Có một thuật toán thực hành cho phép ta tính một "inner almost-collision" chấp nhận được, ví dụ: giá trị ban đầu A, B, C, D và các đầu vào X_{12}, X_{13}, X_{14}, X_{15}, X_0, X_4, X_8 cho compress$_{19}^{12}$ thoả mãn:*

$\Delta_{19} = (0, 1^{<<25}, -1^{<<5}, 0)$

$G(B_*, C_*, D_*) = G(\tilde{B}_*, \tilde{C}_*, D_*)$

Ta có thể tóm tắt ý tưởng của bước này theo quan điểm giải một hệ phương trình như sau:
Hệ phương trình ban đầu gồm 17 phương trình: (1)-> (17);
Số biến: 23, cụ thể là X_{12}, X_{13}, X_{14}, X_{15}, X_0, X_4, X_8 và A, B, C, D, W, U, Z, V, $\tilde{U}, \tilde{Z}, \tilde{W}, \tilde{V}$, A_*, B_*, C_*, D_*.

Sau đó hệ phương trình này được biến đổi (một cách tương đương) về hệ phương trình gồm có 16 phương trình: (9)-> (17); (18)->(22) ; (23) ; (25)

Số biến: 20, cụ thể là X_{12}, X_{13}, X_{14}, X_{15}, X_0, X_4, X_8 và A, C, D, W, Z, V, $\tilde{Z}, \tilde{W}, \tilde{V}$, A_*, B_*, C_*, D_*.

Đây là một hệ phương trình không tuyến tính, nên mặc dù số ẩn nhiều hơn số phương trình, nhưng việc giải không dễ. Ở đây có lẽ cần những kiến thức và sự nhạy cảm của một người làm điều khiển, tính xấp xỉ,….Chắc là có nhiều thuật toán để giải hệ phương trình này, chúng tôi đã phải sử dụng đúng thuật toán do Dobbertin nêu ra (thuật toán tìm Inner Almost Collision); đây là thuật toán thực hành được nhắc tới trong bổ đề 1. Chiến lược giải của Dobbertin được chia làm 3 giai đoạn :

Bước 1: Xét các phương trình (18)-> (21) và (23)
Bước 2: Xét đến các phương trình (18)-> (21); (23) và (24) (ở dạng (25)).
Bước 3: Xét đến các phương trình còn lại (9)-> (17) và (22)

1.3 Differential Attack Modulo 2^{32} (các bước 20-35 của MD4)

Bổ đề 2: Giả sử rằng một 'inner almost collision', cụ thể: giá trị khởi đầu (A,B,C,D) cho bước 12 và các biến X_{12}, X_{13}, X_{14}, X_{15}, X_0, X_4, X_8 theo bổ đề 1. Chọn các X_i còn lại một cách ngẫu nhiên tương ứng với giá trị khởi đầu bằng việc tính $compress_{11}^{0}$ sao cho:

$$(A_{11}, B_{11}, C_{11}, D_{11}) = (A, B, C, D)$$

Thì xác suất để X và $\tilde{X}$ xảy ra va chạm ($\Delta_{35} = 0$) trong hàm nén của MD4 là khoảng 2^{-22}.

Bảng 3

Bước	Δ^*_i				Hàm	Dịch	p_i^{i-1}	Vào	constant
19	0	$1^{<<25}$	$-1^{<<5}$	0	*	*	*	*	*
20	0	$1^{<<25}$	$-1^{<<5}$	0	G	3	1	X_1	K_1
21	0	$1^{<<25}$	$-1^{<<5}$	0	G	5	1/9	X_5	K_1
22	0	$1^{<<25}$	$-1^{<<14}$	0	G	9	1/3	X_9	K_1
23	0	$1^{<<6}$	$-1^{<<14}$	0	G	13	1/3	X_{13}	K_1
24	0	$1^{<<6}$	$-1^{<<14}$	0	G	3	1/9	X_2	K_1
25	0	$1^{<<6}$	$-1^{<<14}$	0	G	5	1/9	X_6	K_1
26	0	$1^{<<6}$	$-1^{<<23}$	0	G	9	1/3	X_{10}	K_1
27	0	$1^{<<19}$	$-1^{<<23}$	0	G	13	1/3	X_{14}	K_1
28	0	$1^{<<19}$	$-1^{<<23}$	0	G	3	1/9	X_3	K_1
29	0	$1^{<<19}$	$-1^{<<23}$	0	G	5	1/9	X_7	K_1
30	0	$1^{<<19}$	-1	0	G	9	1/3	X_{11}	K_1
31	0	1	-1	0	G	13	1/3	X_{15}	K_1
32	0	1	-1	0	H	3	1/3	X_0	K_2
33	0	1	-1	0	H	9	1/3	X_8	K_2
34	0	1	0	0	H	11	1/3	X_4	K_2
35	0	0	0	0	H	15	1	$X_{12}(+1)$	K_2

Giả sử p là xác suất để $\Delta_{35} = 0$ với điều kiện cho trước. Chúng ta phải chứng minh rằng $p \approx 2^{-22}$. Phần chứng minh cụ thể của phần này được chúng tôi thực hiện lại và trình bày tường minh ở phần III.

1.4 Right Initial Value (các bước 0 - 11 của MD4)

Vấn đề còn lại là tính va chạm với giá trị ban đầu IV_o của MD4. Giả sử có một 'inner almost-collision' với giá trị khởi đầu (A, B, C, D). Lấy ngẫu nhiên các giá trị X_1, X_2, X_3, X_5 (X_0, X_4 và X_8 đã được cố định). Tính $\text{compress}_5^0(IV_o; X_0, \ldots, X_5)$. Lúc này $A_5 = A_4$, $B_5 = B_4 = B_3$, $C_5 = C_4 = C_3 = C_2$, D_5 là cố định. Tiếp theo ta tìm X_6, X_7, X_9, X_{10} và X_{11} sao cho đầu ra của compress_{11}^0 trùng với (A, B, C, D). Nói cách khác:

$$\text{compress}_{11}^6((A_4, B_3, C_2, D_5); X_6, \ldots, X_{11}) = (A, B, C, D)$$

Từ bước 8 của MD4, ta có:

$$A_8 = (A_4 + F(B_7, C_6, D_5) + X_8)^{<<3}$$

Từ công thức này, ta thấy rằng $A_8 = A$ nếu $B_7 = -1 =$ 0xffffffff và $C_6 = A^{<<29} - A_4 - X_8$. Ý tưởng này dẫn tới các biểu thức sau:

$$X_6 = -C_2 - F(D_5, A_4, B_3) + (A^{<<29} - A_4 - X_8)^{<<21},$$
$$C_6 = (C_2 + F(D_5, A_4, B_3) + X_6)^{<<11} = A^{<<29} - A_4 - X_8$$
$$X_7 = -B_3 - F(C_6, D_5, A_4) - 1$$
$$B_7 = (B_3 + F(C_6, D_5, A_4) + X_7)^{<<19} = -1$$
$$A_8 = (A_4 + F(-1, C_6, D_5) + X_8)^{<<3} = A$$
$$X_9 = D^{<<25} - D_5 - F(A, -1, C_6)$$
$$D_9 = (D_5 + F(A, -1, C_6) + X_9)^{<<7} = D$$
$$X_{10} = C^{<<21} - C_6 - F(D, A, -1)$$
$$C_{10} = (C_6 + F(D, A, -1) + X_{10})^{<<11} = C$$
$$X_{11} = B^{<<13} + 1 - F(C,D,A)$$
$$B_{11} = (-1 + F(C,D,A) + X_{11})^{<<19} = B$$

Có nghĩa là chúng ta có:

$$\text{compress}_{11}^0((A_4, B_3, C_2, D_5); X_0, \ldots, X_{11}) = (A_{11}, B_{11}, C_{11}, D_{11})$$
$$= (A_8, B_{11}, C_{10}, D_9) = (A, B, C, D)$$

1.5 Thuật toán tìm kiếm va chạm

Chúng ta đã mô tả cả 3 phần tấn công của MD4, đến đây ta có một thuật toán tổng quan cho tìm kiếm va chạm đối với MD4:

1. Tính A, B, C, D và X_{12}, X_{13}, X_{14}, X_{15}, X_0, X_4, X_8 để tìm 'inner almost-collision' (từ bước 12 đến 19). Thuật toán chi tiết được mô tả trong phần 1.2.
2. Theo phần 1.3 và 1.4 ở trên, chọn X_1, X_2, X_3, X_4 ngẫu nhiên và tính:

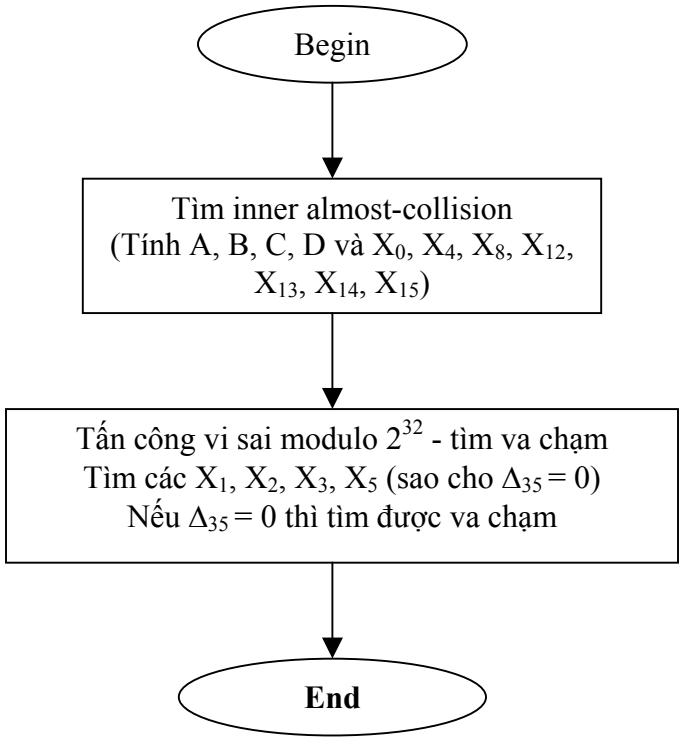
$$(A_5, B_5, C_5, D_5) = \text{compress}_5^0(IV_o; X_0, \ldots, X_5), \quad (27)$$
$$t = A^{<<29} - A_5 - X_8, \quad (28)$$
$$X_6 = t^{<<21} - C_5 - F(D_5, A_5, B_5), \quad (29)$$
$$X_7 = -B_5 - F(t, D_5, A_5) - 1 \quad (30)$$

$$X_9 = D^{<<25} - D_5 - F(A, -1, t), \quad (31)$$
$$X_{10} = C^{<<21} - t - F(D, A, -1) \quad (32)$$
$$X_{11} = B^{<<13} + 1 - F(C,D,A) \quad (33)$$
$$(A_{35}, B_{35}, C_{35}, D_{35}) = \text{compress}_{35}^{20}(A_{19}, B_{19}, C_{19}, D_{19});X), \quad (34)$$
$$(\tilde{A}_{35}, \tilde{B}_{35}, \tilde{C}_{35}, \tilde{D}_{35}) = \text{compress}_{35}^{20}((\tilde{A}_{19}, \tilde{B}_{19}, \tilde{C}_{19}, \tilde{D}_{19});X), \quad (35)$$

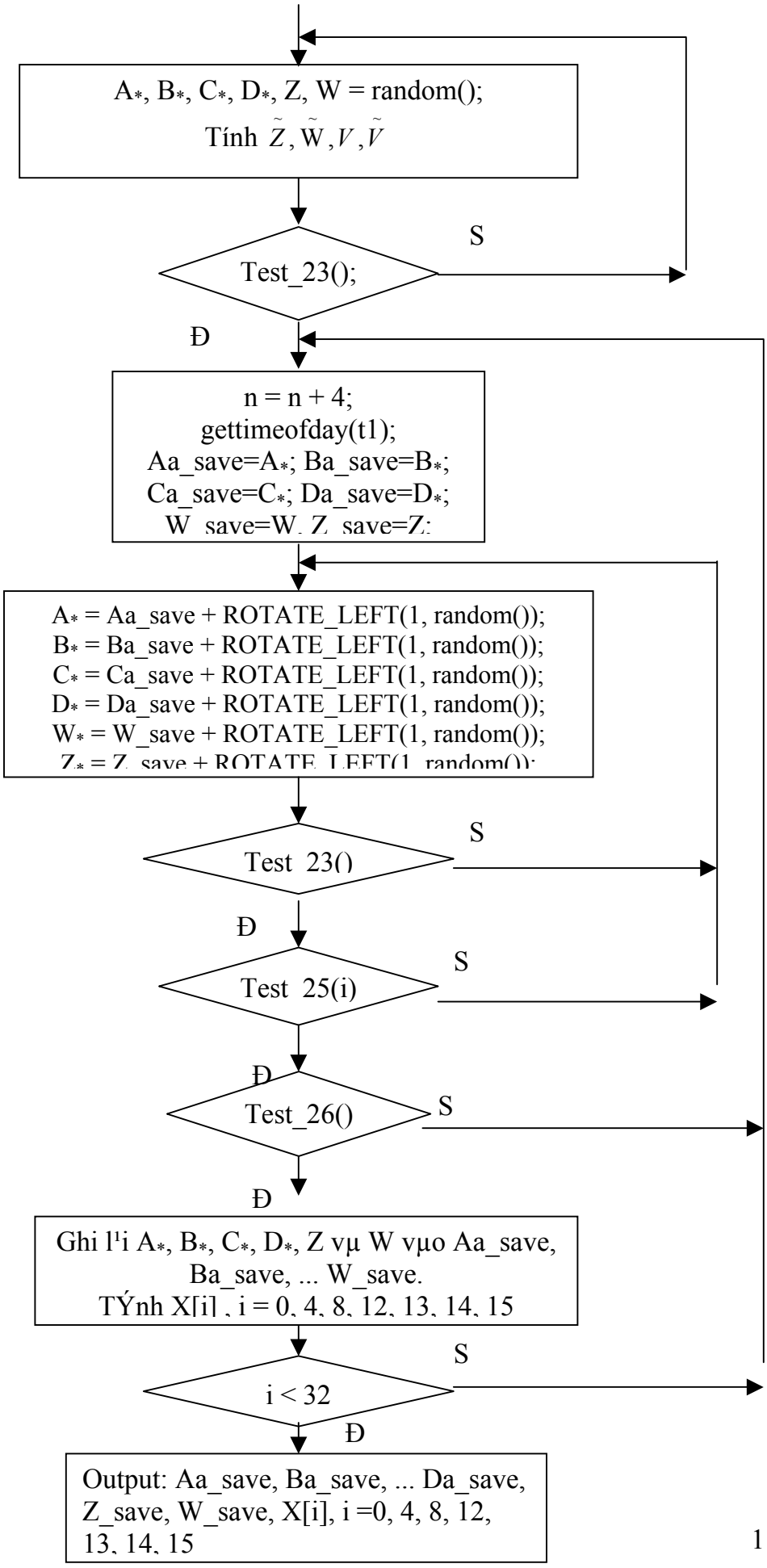
3. Nếu $\Delta_{35} = 0$, thì chúng ta đã tìm được collision. Nếu không thì nhảy về bước 2.

2. Sơ đồ thuật toán

a. Thuật toán tìm kiếm va chạm



b. Thuật toán tìm Inner almost-collision


A*, B*, C*, D*, Z, W = random();
Tính $\tilde{Z}, \tilde{W}, V, \tilde{V}$
Test_23();
S
Đ
n = n + 4;
gettimeofday(t1);
Aa_save=A*; Ba_save=B*;
Ca_save=C*; Da_save=D*;
W_save=W; Z_save=Z;
A* = Aa_save + ROTATE_LEFT(1, random());
B* = Ba_save + ROTATE_LEFT(1, random());
C* = Ca_save + ROTATE_LEFT(1, random());
D* = Da_save + ROTATE_LEFT(1, random());
W* = W_save + ROTATE_LEFT(1, random());
Z* = Z_save + ROTATE_LEFT(1, random());
Test_23()
S
Đ
Test_25(i)
S
Đ
Test_26()
S
Đ
Ghi l¹i A*, B*, C*, D*, Z vµ W vµo Aa_save,
Ba_save, ... W_save.
TÝnh X[i], i = 0, 4, 8, 12, 13, 14, 15
i < 32
S
Đ
Output: Aa_save, Ba_save, ... Da_save,
Z_save, W_save, X[i], i =0, 4, 8, 12,
13, 14, 15

3. Các nhận xét trong quá trình lập trình tấn công MD4

Dựa trên thuật toán được Dobbertin mô tả, chúng tôi đã lập trình cho thuật toán thám, chương trình được viết bằng ngôn ngữ C và chạy trên máy Dell - 350MHz.

- Hiệu chỉnh một số công thức: do soạn thảo, bài báo đã được in với một vài chỗ không chính xác. Chúng tôi đã sửa lại cho đúng logic, đó là các chỗ sau:

1. Ở dòng 20, trang 6. Nguyên bản là thiết lập $\tilde{U}$ = -1 = 0xffffffff, U = 0. Đúng phải là U = -1 = 0xffffffff, $\tilde{U}$ = 0.
2. Biểu thức (22) - trang 6. Nguyên bản là: C = $\tilde{V}^{<<25} - V^{<<25}$, được sửa lại đúng là: C = $V^{<<25} - \tilde{V}^{<<25}$.
3. Biểu thức (24) và (25) - trang 6, 7. Nguyên bản là:

$$F(\tilde{W},\tilde{V},-1) - F(W,V,0) - \tilde{Z}^{<<13} + Z^{<<13} = 0$$

và

$$F(\tilde{W},\tilde{V},-1) - F(W,V,0) - \tilde{Z}^{<<13} + Z^{<<13}$$

được sửa lại cho đúng như sau:

$$F(\tilde{W},\tilde{V},0) - F(W,V,-1) - \tilde{Z}^{<<13} + Z^{<<13} = 0$$

và

$$F(\tilde{W},\tilde{V},0) - F(W,V,-1) - \tilde{Z}^{<<13} + Z^{<<13}.$$

- Thuật toán tìm 'inner almost-collision' được trình bày ở phần 1.2 được tác giả mô tả chạy dưới 1 giây. Song trong lập trình thử nghiệm, thì chương trình chạy khá lâu (vài ngày) vẫn không ra kết quả. Thông thường có thể test biểu thức (25) với 16, 20 và 24 với khoảng thời gian ngắn (vài giây).

- Phát hiện và cải tiến thuật toán:

- Qua seminar nhóm, phân tích và chạy chương trình rất nhiều lần, trên nhiều máy tính khác nhau, kết quả như sau: ở mỗi thời điểm chương trình có thể tìm ngay ra được 'inner almost-collision' dưới một giây (như tác giả đã trình bày). Như vậy có thể "phỏng đoán" thuật toán "xấp xỉ tiếp tục" được sử dụng có ý nghĩa như sau: với mỗi bộ "giá trị cơ sở" (basic value) có thể tìm được bộ "giá trị cơ sở" mới ở một lân cận nhỏ nào đó của cơ sở cũ. Và nếu sau một khoảng nào đó mà không thấy thì xác suất thành công của thuật toán sẽ rất nhỏ. Do đó, thuật toán được cải tiến như sau: với mỗi bộ giá trị cơ sở A_*, B_*, C_*, D_*, Z, W ta chỉ test và tìm "cơ sở" mới sau một khoảng thời gian nhỏ 2 giây. Cải tiến này rất có hiệu quả trong việc tìm ra "inner almost-collision".

- Ngoài ra, còn nhận thấy rằng hàm tạo số ngẫu nhiên ramdom() thường là thanh ghi dịch có phản hồi, nếu được khởi tạo với cùng một "mầm khoá" thì tính ngẫu nhiên là không có. Nói khác đi thì các giá trị "cơ sở" lần sau vẫn giống các kết quả của lần trước. Do đó để đảm bảo tính ngẫu nhiên, chương trình sẽ liên tục thay đổi "mầm khoá" bằng cách lấy thời gian tính theo nanogiây và giây của đồng hồ hệ thống. Như vậy đảm bảo việc tạo các cơ sở là hoàn toàn ngẫu nhiên.

- Với cải tiến mới này, chúng tôi đã tìm được rất nhiều các va chạm khác nhau (trung bình khoảng 1^h15 phút chúng tôi có thể tìm được 1 hầu va chạm bên trong). Chú ý rằng với mỗi một hầu va chạm bên trong tìm được, chúng ta có thể tìm được rất nhiều các va chạm (mà theo tác giả là khoảng 2^{106} va chạm). Các va chạm này được kiểm tra lại với MD4, tất cả đều đúng (2 thông điệp có cùng hash value). Một vài va chạm cụ thể mà chúng tôi tìm thấy được trích ở phần dưới.

- Thử nghiệm với va chạm có nghĩa mà tác giả đưa ra (bản hợp đồng mua bán được sửa đi phần giá cả - khác một con số). Điều này đúng như tác giả đã mô tả, cả 2 thông điệp này đều cho ra một giá trị hash.

- Một vài kết quả thực thi của chương trình thám MD4:

1. Với hầu va chạm tìm được sau:
A^* = 0xde9f958e, B^* = 0x4a65d27d, C^* = 0x63e55380,
D^* = 0x0b73a3e5, Z = 0x5d15702b, W = 0xedfbcfdd

Ta có cặp va chạm thứ 1:

X_0 = 0xa3537919,	X_8 = 0x189bf784,
X_1 = 0x47baaf9a,	X_9 = 0x3c58eacc,
X_2 = 0x1703078c,	X_{10} = 0x2bac77bd,
X_3 = 0x18b21760,	X_{11} = 0x58807999,
X_4 = 0xf03b4df9,	X_{12} = 0x905ad5e6,
X_5 = 0x5a2ef2b7,	X_{13} = 0x57c508d9,
X_6 = 0xfffbdd44,	X_{14} = 0xfffdbf79,
X_7 = 0x1c014184,	X_{15} = 0xc00b9bc6.

và $\tilde{X}_i = X_i$, với i ≠ 12 và $\tilde{X}_{12} = X_{12} + 1$ = 0x905ad5e7.

2. Với hầu va chạm tìm được sau:
A^* = 0x6eb77687, B^* = 0x93c2936e, C^* = 0xa932145e,
D^* = 0x2d224171, Z = 0xaf1cd175, W = 0xffffefdf

Ta có cặp va chạm thứ 2:

X_0 = 0x93547538,
X_1 = 0x6935d3a3,
X_2 = 0x5b7379ad,
X_3 = 0x4bbb4f7c,
X_4 = 0x3f26a09d ,
X_5 = 0x30b664f5,
X_6 = 0x6bc0a73,
X_7 = 0x2819a7c,
X_8 = 0x761bde1d,
X_9 = 0x064bd926,
X_{10} = 0xbaa1401d,
X_{11} = 0x5a827999,
X_{12} = 0xc92afeaf,
X_{13} = 0x3fc28c94,
X_{14} = 0xfffffffd,
X_{15} = 0x00000605.

và $\tilde{X}_i = X_i$, với i ≠ 12 và $\tilde{X}_{12} = X_{12} + 1$ = 0xc92afeb0.

III - TÍNH TOÁN LẠI XÁC SUẤT THÀNH CÔNG Ở BƯỚC 2

Mục đích làm sáng tỏ mục 4 (Differential Attack Modulo 2^{32}) được trình bày trong bài báo "Cryptanalysis of MD4" của Dobbertin. Đây là thủ tục tấn công vi sai để cho phép ta tìm các va chạm cho hàm nén của MD4.

Như đã trình bày ở phần II, tư tưởng chủ đạo cho thám mã MD4 là dựa trên phương pháp "xấp xỉ tiếp tục" (continuous approximation), nhưng vấn đề quyết định tới khả năng thành công của thuật toán cũng cần phải làm sáng tỏ. Đó là:

- việc chọn $\Delta_{19} = (0, 1^{<<25}, -1^{<<5}, 0)$ và
- việc đưa ra bảng 3 với các xác suất cho từng bước trong bảng.

Nếu làm rõ được việc này thì ta có thể học được phương pháp để có thể sử dụng cho việc phân tích, thám mã các hàm hash dựa trên MD4 khác. Để tiện theo dõi quá trình trình bày ở dưới, chúng ta có thể tham khảo Bảng 3(ở trên).

Các ký hiệu được sử dụng trong phần này hơi khác một chút: $X_i (0 \le i \le 15)$ và X'_i $(0 \le i \le 15)$ là 2 message cần tìm, trong đó X'_i bằng với X_i, trừ $X'_{12} = X_{12} + 1$. Và tương tự các (A_j, B_j, C_j, D_j) và (A'_j, B'_j, C'_j, D'_j) ký hiệu cho nội dung 4 thanh ghi (32 bit) sau bước thứ j tương ứng với đầu vào X và X'. Các ký hiệu còn lại vẫn được sử dụng.

Cần thấy rằng để tìm được va chạm thì $\Delta_{35} = 0$. Ta cần tìm giá trị Δ_{19} để $\Delta_{35} = 0$ với một xác suất là cao nhất, đủ để có thể thực hiện việc thám mã MD4. Ta sẽ lần lượt xét và tìm sự sai khác của các thanh ghi qua từng bước của thuật toán.

1. Step 35

Để $\Delta_{35} = 0 \Leftrightarrow A_{35} = A'_{35}, B_{35} = B'_{35}, C_{35} = C'_{35}, D_{35} = D'_{35}$. (1)
Ở bước này chỉ có thanh ghi B_{35} bị thay đổi nội dung, ta có:

$$
\begin{aligned}
& B_{35} = B'_{35} \\
\Leftrightarrow\ & (B_{34} + H(C_{35}, D_{35}, A_{35}) + X_{12} + K_2)^{<<15} = \\
& = (B'_{34} + H(C'_{35}, D'_{35}, A'_{35}) + X_{12} + 1 + K_2)^{<<15}
\end{aligned}
$$

Từ (1) $\Rightarrow B_{34} - B'_{34} = 1$.
hay: $\Delta_{34} = (0, 1, 0, 0)$ với xác suất $p_{35} = 1$.

2. Step 34.
Có: $\Delta_{34} = (0, 1, 0, 0)$. Trong bước này chỉ có nội dung của thanh ghi C thay đổi, nên ta có:

$$
\begin{aligned}
& C_{34} = C'_{34}. \\
\Leftrightarrow\ & (C_{33} + H(D_{34}, A_{34}, B_{34}) + X_4 + K_2)^{<<11} = \\
& = (C'_{33} + H(D'_{34}, A'_{34}, B'_{34}) + X_4 + K_2)^{<<11} \\
\Leftrightarrow\ & C_{33} - C'_{33} = H(D'_{34}, A'_{34}, B'_{34}) - H(D_{34}, A_{34}, B_{34}). \\
& \qquad = H(D_{34}, A_{34}, B_{34} - 1) - H(D_{34}, A_{34}, B_{34}). \qquad (2)
\end{aligned}
$$

Xét vế phải của (2) với 3 biến D, A, B độc lập với nhau (là các thanh ghi 32 bit). Để tìm các giá trị có thể của biểu thức trên và xác suất của nó ta có thể sử dụng phương pháp ước lượng. Ta lấy ngẫu nhiêu 10.000.000 giá trị D, A, B và thực hiện bằng 1 chương trình tính xác suất xuất hiện các giá trị của (2). Kết quả được sắp xếp theo thứ tự xác suất từ cao đến thấp như sau:

```
The number of times take random to test: 10000000

             0  Count: 3334130.        33% 1/(  2,999)       *
            -1  Count: 3331027.        33% 1/(  3,027)
             1  Count:  835112.         8% 1/( 11,813768)
            -2  Count:  831972.         8% 1/( 12,16336)
             3  Count:  208688.         2% 1/( 47,191664)
            -3  Count:  208669.         2% 1/( 47,192557)
            -4  Count:  208545.         2% 1/( 47,198385)
             2  Count:  208011.         2% 1/( 48,15472)
          .. ..
          other                      <<    1/191
```

Ở đây (2) nhận giá trị 0 hoặc -1 với xác suất cao nhất. Ở đây ta chọn đẳng thức bằng -1 để $C_{33} - C'_{33}$ có sự sai khác rất nhỏ. Lúc này (2) là:

$C_{33} - C'_{33} = -1$ với xác suất $p \approx 1/3$
Như vậy, ta có:

$\Delta_{33} = (0, 1, -1, 0)$ với xác suất $p_{34} \approx 1/3$

3. Step 33

Nội dung của thanh ghi D bị thay đổi, ta có:

$$
\begin{aligned}
& D_{33} = D'_{33} \\
\Leftrightarrow \quad & (D_{32} + H(A_{33}, B_{33}, C_{33}) + X_8 + K_2)^{<<9} = \\
& = (D'_{32} + H(A'_{33}, B'_{33}, C'_{33}) + X_8 + K_2)^{<<9} \\
\Leftrightarrow \quad & D_{32} - D'_{32} = H(A_{33}, B'_{33} - 1, C'_{33} + 1) - H(A_{33}, B_{33}, C_{33}) \qquad (3)
\end{aligned}
$$

Tương tự như Step 34, ta xét vế phải của (3) với A_{33}, B_{33}, C_{33} độc lập, ta thu được kết quả như sau:

```
The number of times take random to test: 10000000

            0  Count: 3332810.        33% 1/(  3,0015)     *
            1  Count: 1667715.        16% 1/(  5,9962)
           -1  Count: 1667619.        16% 1/(  5,9962)
            3  Count:  417511.         4% 1/( 23,397247)
            2  Count:  417069.         4% 1/( 23,407413)
           -2  Count:  416475.         4% 1/( 24,4600)
            các trường hợp khác       <    1/95
```

Chọn vế phải của (3) bằng 0, ta có: $D_{32} - D'_{32} = 0$ hay $\Delta_{32} = (0, 1, -1, 0)$ với xác suất $p_{33} \approx 1/3$.

4. Step 32

Thanh ghi A thay đổi, ta có:

$$
\begin{aligned}
& A_{32} = A'_{32}. \\
\Leftrightarrow \quad & (A_{31} + H(B_{32}, C_{32}, D_{32}) + \ldots)^{<<3} = \\
& = (A'_{31} + H(B'_{32}, C'_{32}, D'_{32}) + \ldots)^{<<3} \\
\Leftrightarrow \quad & A_{31} - A'_{31} = H(B_{32} - 1, C_{32} + 1, D_{32}) - H(B_{32}, C_{32}, D_{32}) \qquad (4)
\end{aligned}
$$

Thực hiện hoàn toàn tương tự ta được kết quả sau:

```
The number of times take random to test: 10000000

            0  Count: 3332305.        33% 1/(  3,0303)     *
            1  Count: 1668240.        16% 1/(  5,994341)
           -1  Count: 1665502.        16% 1/(  6,003)
           -3  Count:  417408.         4% 1/( 23,399616)
           -2  Count:  417238.         4% 1/( 23,403526)
            3  Count:  417002.         4% 1/( 23,408954)
            2  Count:  416073.         4% 1/( 24,14248)
            6  Count:  104431.         1% 1/( 95,79055)
```

Ở đây chỉ có 0 xuất hiện với xác suất là cao nhất, ta chọn vế phải của (4) bằng 0, có:

$\Delta_{31} = (0, 1, -1, 0)$, với xác suất $p_{32} \approx 1/3$

Chú ý:

Ở đây tác giả đã chứng minh biểu thức này bằng cách chỉ ra rằng **(R+1)⊕S = R⊕(S+1)** xảy ra với xác suất là 1/3 với R, S là các từ (word) độc lập. Biểu thức này được chứng minh như sau:

Biểu thức này chỉ thoả mãn khi và chỉ khi chính xác một trong các điều kiện sau của R, S xảy ra:

R = *0	và	S = *0
$ = *01	và	S = *01
$ = *011	và	S = *011
….		
$ = 011…11	và	S = 011..11
$ = 111…11	và	S = 111..11

Các dấu '*' đánh dấu dãy bit bất kỳ có độ dài phù hợp. Vì vậy chúng ta có:

$$p_{32} = \frac{1}{2^2} + \frac{1}{4^2} + \frac{1}{8^2} + ... + \frac{1}{2^{62}} + \frac{1}{2^{64}} + \frac{1}{2^{64}} = \frac{1}{3}\left(1 + \frac{1}{2^{63}}\right)$$

Áp dụng đẳng thức đã chứng minh này, ta có:

$$\begin{aligned} H(B'_{32}, C'_{32}, D'_{32}) &= (B_{32} - 1) \oplus (C_{32} + 1) \oplus D_{32} \\ &= (B_{32} - 1 + 1) \oplus C_{32} \oplus D_{32}, \text{ với } p \approx 1/3 \\ &= H(B_{32}, C_{32}, D_{32}), \text{ với } p \approx 1/3 \end{aligned}$$

Tuy nhiên trong các bước khác nhau ta cần có các đẳng thức khác, mà ta chưa thể chỉ ra được cách chứng minh tường minh như trên. Chẳng hạn với Step 33, ta cần chứng minh đẳng thức $R \oplus (S-1) = R \oplus S - 1$, cũng với $p \approx 1/3$. Và từ các bước 31 trở lên tới 20, các đẳng thức sẽ phụ thuộc vào 3 biến và các toán tử AND, OR. Tuy nhiên tác giả nói rằng có thể tìm được xác suất trên bằng phương pháp Monte Carlo đơn giản.

5. Step 31

Có $B_{31} - B'_{31} = 1$

$$\begin{aligned} \Leftrightarrow \quad & (B_{30} + G(C_{31}, D_{31}, A_{31}) + ...)^{<<13} = \\ & = 1 - (B'_{30} + G(C'_{31}, D'_{31}, A'_{31}) + ...)^{<<13} \end{aligned}$$

$$\Leftrightarrow \quad B_{30} - B'_{30} = 1^{<<19} + G(C_{31} + 1, D_{31}, A_{31}) - G(C_{31}, D_{31}, A_{31}) \qquad (5)$$

Xét biểu thức $G(C_{31} + 1, D_{31}, A_{31}) - G(C_{31}, D_{31}, A_{31})$, thực hiện tương tự như các bước ở trên ta có:

```
The number of times take random to test: 10000000

            1  Count: 3334909.        33% 1/(  2,998)
            0  Count: 3331574.        33% 1/(  3,0015)     *
           -1  Count:  833767.         8% 1/( 11,828563)
            2  Count:  832756.         8% 1/( 12,6928)
            4  Count:  208647.         2% 1/( 47,193591)
           -3  Count:  208558.         2% 1/( 47,197774)
           -2  Count:  207949.         2% 1/( 48,18448)
            3  Count:  207770.         2% 1/( 48,27040)
           . . .
```

Chọn giá trị 0 cho biểu thức này, (5) trở thành:

$B_{30} - B'_{30} = 1^{<<19}$ hay
$\Delta_{30} = (0, 1^{<<19}, -1, 0)$ với xác suất $p \approx 1/3$

6. Step 30

$C_{30} - C'_{30} = -1$
$\Leftrightarrow \quad (C_{29} + G(D_{30}, A_{30}, B_{30}) + \ldots.)^{<<9} =$
$=-1 - (C'_{29} + G(D'_{30}, A'_{30}, B'_{30}) + \ldots)^{<<9}$

$$\Leftrightarrow \quad C_{29} - C'_{29} = -1^{<<23} + G(D_{30}, A_{30}, B_{30} - 1^{<<19}) - G(C_{31}, D_{31}, A_{31}) \qquad (6)$$

Xét biểu thức $G(D_{30}, A_{30}, B_{30} - 1^{<<19}) - G(C_{31}, D_{31}, A_{31})$ và thực hiện việc tương tự ta thu được kết quả sau:

```
The number of times take random to test: 10000000

            0  Count: 3335877.        33% 1/(  2,9977)     *
      -524288  Count: 3330779.        33% 1/(  3,0023)
       524288  Count:  833543.         8% 1/( 11,831027)
     -1048576  Count:  833126.         8% 1/( 12,2488)
      1572864  Count:  208995.         2% 1/( 47,177235)
     -1572864  Count:  208517.         2% 1/( 47,199701)
     -2097152  Count:  208331.         2% 1/( 48, 112)
      1048576  Count:  207572.         2% 1/( 48,36544)
     -3670016  Count:   52169.         0% 1/(191,35721)
```

Chọn giá trị 0 cho biểu thức này, ta có:

$\Delta_{29} = (0, 1^{<<19}, -1^{<<23}, 0)$ với xác suất $p_{29} \approx 1/3$.

7. Step 29

Có: $D_{29} = D'_{29}$.

$\Leftrightarrow$ $(D_{28} + G(A_{29}, B_{29}, C_{29}) + \ldots.)^{<<5} =$
$= (D'_{28} + G(A'_{29}, B'_{29}, C'_{29}) + \ldots)^{<<5}$

$\Leftrightarrow$ $D_{28} - D'_{28} = G(A_{29}, B_{29} - 1^{<<29}, C_{29} + 1^{<<23}) - G(A_{29}, B_{29}, C_{29})$ (7)

Thực hiện tính các giá trị có thể với vế phải của đẳng thức (7) ta thu được kết quả sau:

```
The number of times take random to test: 10000000

           0  Count: 1120114.        11% 1/(  8,927)      *
     7864320  Count: 1118482.        11% 1/(  8,9406)
     -524288  Count: 1113614.        11% 1/(  8,9797)
     8388608  Count: 1113495.        11% 1/(  8,9807)
      524288  Count:  286781.         2% 1/( 34,249446)
     7340032  Count:  286216.         2% 1/( 34,268656)
    -8388608  Count:  283803.         2% 1/( 35,66895)
    16252928  Count:  282537.         2% 1/( 35,111205)
     .. ..
```

Ta chọn giá trị cho biểu thức này bằng 0. Khi đó (7) là:

$D_{28} - D'_{28} = 0$ hay

$\Delta_{28} = (0, 1^{<<19}, -1^{<<23}, 0)$ với xác suất $p_{29} \approx 1/9$

8. Step 28.

Có $A_{28} = A'_{28}$.

$\Leftrightarrow$ $(A_{27} + G(B_{28}, C_{28}, D_{28}) + \ldots.)^{<<3} =$
$= (A'_{27} + G(B'_{28}, C'_{28}, D'_{28}) + \ldots)^{<<3}$

$\Leftrightarrow$ $A_{27} - A'_{27} = G(B_{28} - 1^{<<19}, C_{28} + 1^{23}, D_{28}) - G(B_{28}, C_{28}, D_{28})$ (8)

Thực hiện tương tự với vế phải của (8). Ta thu được các kết quả sau:

```
The number of times take random to test: 10000000

           0  Count: 1119764.        11% 1/(  8,93045)    *
     7864320  Count: 1119595.        11% 1/(  8,931801)
     8388608  Count: 1115341.        11% 1/(  8,9656)
     -524288  Count: 1114095.        11% 1/(  8,975895)
      524288  Count:  286932.         2% 1/( 34,244312)
     7340032  Count:  286814.         2% 1/( 34,248324)
    16252928  Count:  283250.         2% 1/( 35,86250)
    -8388608  Count:  283090.         2% 1/( 35,91850)
```

Ta chọn vế phải của (8) bằng 0, lúc này (8) là:

$A_{27} - A'_{27} = 0$ hay

$\Delta_{27} = (0, 1^{<<19}, -1^{<<23}, 0)$ với xác suất $p_{28} \approx 1/9$

9. Step 27

Có $B_{27} - B'_{27} = 1^{<<19}$.

$\Leftrightarrow$ $(B_{26} + G(C_{27}, D_{27}, A_{27}) + \ldots.)^{<<13} =$

$= (B'_{26} + G(C'_{27}, D'_{27}, A'_{27}) + \ldots)^{<<13} + 1^{<<19}$.

$\Leftrightarrow$ $B_{26} - B'_{26} = 1^{<<6} + G(C_{27} + 1^{<<23}, D_{27}, A_{27}) - G(C_{27}, D_{27}, A_{27})$ (9)

Xét biểu thức $G(C_{27} + 1^{<<23}, D_{27}, A_{27}) - G(C_{27}, D_{27}, A_{27})$, thực hiện tương tự như trên ta có kết quả sau:

```
The number of times take random to test: 10000000

       8388608  Count: 3333159.        33% 1/(  3,0001)
             0  Count: 3331763.        33% 1/(  3,0014)     *
      16777216  Count:  833481.         8% 1/( 11,831709)
      -8388608  Count:  833163.         8% 1/( 12,2044)
      33554432  Count:  208731.         2% 1/( 47,189643)
     -16777216  Count:  208505.         2% 1/( 47,200265
```

Lấy giá trị 0 cho biểu thức này, lúc này (9) là:

$B_{26} - B'_{26} = 1^{<<6}$ hay

$\Delta_{26} = (0, 1^{<<6}, -1^{<<23}, 0)$ với xác suất $p_{27} \approx 1/3$.

10. Step 26.

Có $C_{26} - C'_{26} = -1^{<<23}$.

$\Leftrightarrow$ $(C_{25} + G(D_{26}, A_{26}, B_{26}) + \ldots.)^{<<9} =$

$= (C'_{25} + G(D'_{26}, A'_{26}, B'_{26}) + \ldots)^{<<9} - 1^{<<23}$.

$\Leftrightarrow$ $C_{25} - C'_{25} = -1^{<<14} + G(D_{26}, A_{26}, B_{26} - 1^{<<6}) - G(D_{26}, A_{26}, B_{26})$ (10)

Xét biểu thức $G(D_{26}, A_{26}, B_{26} - 1^{<<6}) - G(D_{26}, A_{26}, B_{26})$, tính toán theo cách tương tự, ta có:

```
The number of times take random to test: 10000000

             0  Count: 3333617.        33% 1/(  2,99974)     *
           -64  Count: 3333040.        33% 1/(  3,00026)
```

```
            64   Count:  833875.         8% 1/( 11,827375)
          -128   Count:  831272.         8% 1/( 12,24736)
          -256   Count:  209087.         2% 1/( 47,172911)
          -192   Count:  208544.         2% 1/( 47,198432
```

Chọn giá trị 0 cho biểu thức này, ta sẽ có:

$C_{25} - C'_{25} = -1^{<<14}$ hay

$\Delta_{25} = (0, 1^{<<6}, -1^{<<14}, 0)$ với xác suất $p_{26} \approx 1/3$.

11. Step 25

Có $D_{25} = D'_{25}$.

$\Leftrightarrow$ $(D_{24} + G(A_{25}, B_{25}, C_{25}) + \ldots.)^{<<5} =$

$= (D'_{24} + G(A'_{25}, B'_{25}, C'_{25}) + \ldots)^{<<5}$

$\Leftrightarrow$ $D_{24} - D'_{24} = G(A_{25}, B_{25} - 1^{<<6}, C_{25} + 1^{<<14}) - G(A_{25}, B_{25}, C_{25})$ (11)

Xét biểu thức vế phải của (11). Kết quả tính toán như sau:

```
The number of times take random to test: 10000000

             0   Count: 1112047.        11% 1/(  8,9924)     *
         16384   Count: 1111843.        11% 1/(  8,9940)
           -64   Count: 1110745.        11% 1/(  9,0029)
         16320   Count: 1109409.        11% 1/(  9,0138)
        -16448   Count:  278556.         2% 1/( 35,250540)
         16256   Count:  278056.         2% 1/( 35,268040)
         32704   Count:  277964.         2% 1/( 35,271260)
```

Lấy giá trị 0 cho biểu thức này. (11) là:

$D_{24} - D'_{24} = 0$ hay

$\Delta_{24} = (0, 1^{<<6}, -1^{<<14}, 0)$ với xác suất $p_{25} \approx 1/9$.

12. Step 24

Có $A_{24} = A'_{24}$.

$\Leftrightarrow$ $(A_{23} + G(B_{24}, C_{24}, D_{24}) + \ldots.)^{<<3} =$

$= (A'_{23} + G(B'_{24}, C'_{24}, D'_{24}) + \ldots)^{<<3}$

$\Leftrightarrow$ $A_{23} - A'_{23} = G(B_{24} - 1^{<<6}, C_{24} + 1^{<<14}, D_{24}) - G(B_{24}, C_{24}, D_{24})$ (12)

Xét và tính toán với vế phải của (12) có:

```
The number of times take random to test: 10000000

           -64   Count: 1111760.        11% 1/(  8,99474)
             0   Count: 1111338.        11% 1/(  8,99816)     *
```

```
         16384   Count: 1109959.        11% 1/(  9,0093)
         16320   Count: 1109594.        11% 1/(  9,0123)
         16256   Count:  278239.         2% 1/( 35,261635)
         32704   Count:  278065.         2% 1/( 35,267725)
```

Lấy vế phải của (12) bằng 0, ta có:

$\Delta_{23} = (0, 1^{<<6}, -1^{<<14}, 0)$ với xác suất $p_{24} \approx 1/9$.

13. Step 23

Có $B_{23} = B'_{23} + 1^{<<6}$.

$\Leftrightarrow$ $(B_{22} + G(C_{23}, D_{23}, A_{23}) + \ldots.)^{<<13} =$

$= (B'_{22} + G(C'_{23}, D'_{23}, A'_{23}) + \ldots)^{<<13} + 1^{<<6}$

$\Leftrightarrow$ $B_{22} - B'_{22} = 1^{<<25} + G(C_{23} + 1^{<<14}, D_{23}, A_{23}) - G(C_{23}, D_{23}, A_{23})$ (13)

Xét và tính toán biểu thức $G(C_{23} + 1^{<<14}, D_{23}, A_{23}) - G(C_{23}, D_{23}, A_{23})$ tương tự, ta thu được kết quả sau:

```
The number of times take random to test: 10000000

             0   Count: 3333428.        33% 1/(  2,9999)     *
         16384   Count: 3331739.        33% 1/(  3,001)
         32768   Count:  833840.         8% 1/( 11,827760)
        -16384   Count:  832806.         8% 1/( 12,6328)
         49152   Count:  208990.         2% 1/( 47,177470)
        -49152   Count:  208843.         2% 1/( 47,184379)
```

Ta lấy 0 cho biểu thức này, (13) có dạng:

$B_{22} - B'_{22} = 1^{<<25}$

hay $\Delta_{22} = (0, 1^{<<25}, -1^{<<14}, 0)$ với xác suất $p_{23} \approx 1/3$.

14. Step 22

Có $C_{22} = C'_{22} - 1^{<<14}$.

$\Leftrightarrow$ $(C_{21} + G(D_{22}, A_{22}, B_{22}) + \ldots.)^{<<9} =$

$= (C'_{21} + G(D'_{22}, A'_{22}, B'_{22}) + \ldots)^{<<9} - 1^{<<14}$

$\Leftrightarrow$ $C_{21} - C'_{21} = -1^{<<5} + G(D_{22}, A_{22}, B_{22} - 1^{<<25}) - G(D_{22}, A_{22}, B_{22})$ (14)

Xét và tính toán các giá trị có thể cho biểu thức $G(D_{22}, A_{22}, B_{22} - 1^{<<25}) - G(D_{22}, A_{22}, B_{22})$, ta có kết quả như sau:

```
The number of times take random to test: 10000000

             0   Count: 3334895.        33% 1/(  2,9985)     *
     -33554432   Count: 3332571.        33% 1/(  3,0006)
```

```
      33554432  Count:  834775.         8% 1/( 11,817475)
     -67108864  Count:  833416.         8% 1/( 11,832424)
      67108864  Count:  210048.         2% 1/( 47,127744)
     100663296  Count:  209509.         2% 1/( 47,153077)
    -100663296  Count:  207098.         2% 1/( 48,59296)
```

Lấy 0 cho biểu thức này, (14) có dạng như sau:

$C_{21} - C'_{21} = -1^{<<5}$ hay

$\Delta_{21} = (0, 1^{<<25}, -1^{<<5}, 0)$ với xác suất $p_{22} \approx 1/3$.

15. Step 21

Có $D_{21} = D'_{21}$.

$\Leftrightarrow$ $(D_{20} + G(A_{21}, B_{21}, C_{21}) + \ldots.)^{<<5} =$

$= (D'_{20} + G(A'_{21}, B'_{21}, C'_{21}) + \ldots)^{<<5}$

$\Leftrightarrow$ $D_{20} - D'_{20} = G(A_{21}, B_{21}-1^{<<25}, C_{21} + 1^{<<5}) - G(A_{21}, B_{21}, C_{21})$ (15)

Xét và tính toán các giá trị có thể cho vế phải của biểu thức (15) ta có:

```
The number of satisfy numbers: 1000000

            32  Count:  111534.        11% 1/   8,9658
    -33554432  Count:  110985.        11% 1/   9,0102
             0  Count:  110909.        11% 1/   9,0164      *
    -33554400  Count:  110732.        11% 1/   9,0308
     33554432  Count:   28242.         2% 1/  35
           64  Count:   27963.         2% 1/  35
```

Lấy giá trị 0 cho biểu thức này, (15) có dạng sau:

$D_{20} - D'_{20} = 0$ hay

$(0, 1^{<<25}, -1^{<<5}, 0)$ với xác suất $p_{21} \approx 1/9$.

16. Step 20

Có $A_{20} = A'_{20}$.

$\Leftrightarrow$ $(A_{19} + G(B_{20}, C_{20}, D_{20}) + \ldots.)^{<<3} =$

$= (A'_{19} + G(B'_{20}, C'_{20}, D'_{20}) + \ldots)^{<<3}$

$\Leftrightarrow$ $A_{19} - A'_{19} = G(B_{20} - 1^{<<25}, C_{20} + 1^{<<5}, D_{20}) - G(B_{20}, C_{20}, D_{20})$ (16)

Xét và tính toán các giá trị có thể cho vế phải của biểu thức (16) ta có:

```
The number of times take random to test: 10000000

             0  Count: 1147000.       11%  1/(  8,7183)      *
            32  Count: 1129000.       11%  1/(  8,  85)
```

```
     -33554432  Count: 1104000.       11%  1/(  9,   5)
     -33554400  Count:  109900.       10%  1/(  9,   9)
      33554464  Count:  292000.       2%   1/( 34,  24)
     -67108832  Count:  279000.         2% 1/( 35,  84)
     -33554368  Count:  277000.         2% 1/( 36,  10)
     -33554464  Count:  268000.         2% 1/( 37,  31)
```

Ở đây ta thấy rằng khi chọn $G(B_{20} - 1^{<<25}, C_{20} + 1^{<<5}, D_{20}) - G(B_{20}, C_{20}, D_{20}) = 0$ thì xác suất để $A_{19} - A'_{19} = 0$ cũng gần bằng 1/9. Nói cách khác: $\Delta_{19} = (0, 1^{<<25}, -1^{<<5}, 0)$ với xác suất $p_{20} \approx 1/9$. Đây chính là một điều kiện khi chọn "inner-almost collision", biểu thức (26) trong bài báo:

$$G(B'_{19}, C'_{19}, D_{19}) - G(B_{19}, C_{19}, D_{19})$$

Đến đây ta có thể viết lại thành bảng Δ^*_i được trình bày ở phần đầu. Và xác suất thành công của thuật toán là tích của các xác suất thành phần ở trên.

$$p = \prod_{i=20}^{35} p_i^{i-1} \approx 2^{-24.54}$$

IV - NHẬN XÉT VÀ MỞ RỘNG

1. Nhận xét

Theo các kết quả được tính toán bằng chương trình ở trên, theo ý kiến cá nhân có một vài nhận xét nhỏ sau:

- Với sự sai khác nhỏ (một vài vị trí bit) ở các biến đầu vào của các hàm logic G, H thì đầu ra của chúng sẽ bằng nhau với một xác suất rất cao. Có thể do đó do đó mà tác giả luôn chọn các sai khác là rất nhỏ. Ví dụ:

Trong bước 21 (trình bày ở trên), biểu thức:

$$G(A_{21}, B_{21}-1^{<<25}, C_{21} + 1^{<<5}) - G(A_{21}, B_{21}, C_{21})$$

có thể lấy các giá trị sau với xác suất ngang nhau:

32	với xác suất là 1/8
-33554432	với xác suất là 1/9
0	với xác suất là 1/9
-33554400	với xác suất là 1/9

Rõ ràng biểu thức này có thể lấy giá trị 32 với xác suất cao hơn, nhưng tác giả lại chọn bằng 0.

- Nếu điều này là đúng thì cách chọn như tác giả đã làm sẽ là "con đường" cho xác suất thành công là lớn nhất (như tác giả đã trình bày).

- Để làm rõ thêm "sự nghi ngờ" này, tiến hành thử lấy các giá trị khác (lớn hơn) và có xác suất cao hơn hoặc tương đương. Cụ thể như sau:

Thử nghiệm 1:

Trong bước 23 (mục 13 ở trên): biểu thức $G(C_{23} + 1^{<<14}, D_{23}, A_{23}) - G(C_{23}, D_{23}, A_{23})$ có thể nhận giá trị 16384 cũng với xác suất là 1/3 (bằng với nhận giá trị 0). Nếu lấy giá trị này cho biểu thức, ta có:

$$\Delta_{22} = (0, 1^{<<25} + 16384, -1^{<<14}, 0) \text{ với xác suất } p_{23} \approx 1/3.$$

Ta thực hiện tiếp với bước 22, có:

$$C_{21} - C'_{21} - 1^{<<5} = G(D_{22}, A_{22}, B_{22} - 1^{<<25} - 16384) - G(D_{22}, A_{22}, B_{22})$$

Kiểm tra bằng thực nghiệm, biểu thức ở vế phải có thể nhận các giá trị sau:

```
-33554432  Count:   11275.        11% 1/(  8,  86)
-33570816  Count:   11163.        11% 1/(  8,  95)
   -16384  Count:   11080.        11% 1/(  9,   2)
        0  Count:   11009.        11% 1/(  9,   8)
```

Tiếp tục chọn giá trị -33554432 (giá trị cho xác suất cao nhất) cho biểu thức này, ta có:

$$\Delta_{21} = (0, 1^{<<25} + 16384, -1^{<<5} - 33554432, 0) \text{ với xác suất } p_{22} \approx 1/8{,}86.$$

Thực hiện tương tự với bước 21:

$$D_{20} - D'_{20} = G(A_{21}, B_{21} - 1^{<<25} - 16384, C_{21} + 1^{<<5} + 33554432) - G(A_{21}, B_{21}, C_{21})$$

Kết quả kiểm tra với vế phải là:

```
   -16352  Count:    3798.         3% 1/( 26,  32)
   -16384  Count:    3678.         3% 1/( 27,  18)
       32  Count:    3662.         3% 1/( 27,  30)
        0  Count:    3641.         3% 1/( 27,  46)     *
 33538080  Count:    1910.         1% 1/( 52,  35)
```

Đến đây ta có thể thấy rằng các giá trị xuất hiện với xác suất cao nhất đã giảm đi một cách rõ rệt. Một thử nghiệm khác để làm sáng tỏ vấn đề này.

Thử nghiệm 2:

Trong bước 29 (mục 7 - phần I), vế phải của (7) có thể nhận giá trị 7864320 với xác suất là 1/8,1 (ngang bằng với 0). Với giá trị này, có:

$$\Delta_{28} = (0, 1^{<<19}, -1^{<<23}, 786430) \text{ với xác suất } p_{29} \approx 1/9$$

Với Δ_{28} ta thực hiện tiếp với bước 28, có:

$$A_{27} - A'_{27} = G(B_{28} - 1^{<<19}, C_{28} + 1^{23}, D_{28} - 786430) - G(B_{28}, C_{28}, D_{28})$$

Kiểm tra bằng thực nghiệm với biểu thức ở vế phải của phương trình trên, kết quả như sau:

```
-524286  Count:     156.          1% 1/( 64,  16)
7602178  Count:     147.          1% 1/( 68,   4)
7864322  Count:     144.          1% 1/( 69,  64)
-786432  Count:     141.          1% 1/( 70, 130)
7602176  Count:     135.          1% 1/( 74,  10)
-786430  Count:     134.          1% 1/( 74,  84)
-524288  Count:     122.         1%  1/( 81, 118)
```

Ở ngay sau 1 bước, ta đã thấy các giá trị xuất hiện với xác suất xuất cao nhất cho biểu thức này đã là rất nhỏ.

Như vậy để $(A_{i+1}, B_{i+1}, C_{i+1}, D_{i+1})$ và $(A'_{i+1}, B'_{i+1}, C'_{i+1}, D'_{i+1})$ rất gần nhau (hoặc bằng nhau) với xác suất cao nhất ta phải tìm các giá trị (A_i, B_i, C_i, D_i) và (A'_i, B'_i, C'_i, D'_i) sao cho sự sai khác giữa chúng là rất nhỏ.

2. Mở rộng

Với các nhận xét ở trên, ta có thể tìm được một bộ giá trị Δ_{19} khác với xác suất thành công chấp nhận được, cho phép thám thành công MD4.

Bằng phương pháp thực hiện như ở trên, và sự sai khác so với bảng 1 bắt đầu từ bước 33.

Bảng 2

Bước	Δ^*_i				Hàm	Dịch	p_i^{i-1}	Vào	constant
19	$-1^{<<23}$	$1^{<<25}$	0	0	*	*	*	*	*
20	$-1^{<<26}$	$1^{<<25}$	0	0	G	3	1/3	X_1	K_1
21	$-1^{<<26}$	$1^{<<25}$	0	0	G	5	1/6,01	X_5	K_1
22	$-1^{<<26}$	$1^{<<25}$	0	0	G	9	1/5,998	X_9	K_1
23	$-1^{<<26}$	$1^{<<6}$	0	0	G	13	1/3	X_{13}	K_1
24	$-1^{<<29}$	$1^{<<6}$	0	0	G	3	1/3	X_2	K_1
25	$-1^{<<29}$	$1^{<<6}$	0	0	G	5	1/8,38	X_6	K_1
26	$-1^{<<29}$	$1^{<<6}$	0	0	G	9	1/8,01	X_{10}	K_1
27	$-1^{<<29}$	$1^{<<19}$	0	0	G	13	1/3	X_{14}	K_1

28	-1	$1^{<<19}$	0	0	G	3	1/3,2	X_3	K_1
29	-1	$1^{<<19}$	0	0	G	5	1/8,97	X_7	K_1
30	-1	$1^{<<19}$	0	0	G	9	1/9,4	X_{11}	K_1
31	-1	1	0	0	G	13	1/3,5	X_{15}	K_1
32	0	1	0	0	H	3	1/3,3	X_0	K_2
33	0	1	0	0	H	9	1/3	X_8	K_2
34	0	1	0	0	H	11	1/3	X_4	K_2
35	0	0	0	0	H	15	1	$X_{12}(+1)$	K_2

Xác suất tổng là:

$$p = \prod_{i=20}^{35} p_i^{i-1} \approx 2^{-32}$$

Đây chính là phần 2 (tấn công vi sai của MD4), phần còn lại là tìm inner almost-collision với Δ_{19} = (-$1^{<<23}$, $1^{<<25}$, 0) và right initial value. Ta cũng thu được một hệ các phương trình sau với việc đặt U = -1 = 0xffffffff, $\tilde{U} = 0$, B = 0.

X_{13} = tùy ý (9')
$X_{14} = W^{<<21} - C - F(V,U,B)$ (10')
$X_{15} = Z^{<<13} - B - F(W,V,U)$ (11')
$X_0 = A_*^{<<29} - U - G(Z,W,V) - K1$ (12')
$X_4 = D_*^{<<27} - V - G(A_*, Z, W) - K1$ (13')
$X_8 = C_*^{<<23} - W - G(D_*, A_*, Z) - K1$ (14')
$X_{12} = B_*^{<<19} - Z - G(C_*, D_*, A_*) - X_{13}$ (15')
$D = V^{<<25} - F(U,B,C) - X_{13}$ (16')
$A = U^{<<29} - F(B,C,D) - X_{12}$ (17')

$$Z' = Z - G(C'_*, D'_*, A'_*) + G(C_*, D_*, A_*) + B'^{<<19}_* - B^{<<19}_* - 1 \quad (18')$$
$$W' = W - G(D'_*, A'_*, Z') + G(D_*, A_*, Z) + C'^{<<23}_* - C^{<<23}_* \quad (19')$$
$$V = W'^{<<21} - W^{<<21} \quad (20')$$
$$V' = V - G(A'_*, Z', W') + G(A_*, Z, W) + D'_* - D_* \quad (21')$$
$$C = V^{<<25} - V'^{<<25} \quad (22')$$

$$G(Z,W,V) - G(Z',W',V') = 1 + A'_* - A_* \quad (23')$$
$$F(W',V',0) - F(W,V,-1) - Z'^{<<13} + Z^{<<13} = 0 \quad (24')$$

Đến đây ta có thể áp dụng thuật toán tìm hầu va chạm của Dobbertin với hệ phương trình từ (9') đến (24') này. Phần tìm giá trị khởi đầu đúng (phần 3) cũng được áp dụng như trên không thay đổi về phương trình.

Sử dụng bộ Δ_{19} tìm được ở trên cùng với việc sử dụng thuật toán của Dobbertin, ta cũng rất dễ dàng tìm được các va chạm cho MD4. Ví dụ: ở dưới là các va chạm tìm được sử dụng kết quả tìm được ở trên:

- Cặp message 1:

```
Message 1:
     0xa4bf59ec  0x4f2e8c73  0x6123c8ba  0x50211945
     0xb6fd1653  0x3fe82e34  0x26c50cc9  0x1466d
     0xcc8b9b9a  0xff619d62  0xe81de7dc  0x58825999
     0x5d0bdf18  0x4f315aae  0xffffbff9  0x9cc4d49c

Message 2

     0xa4bf59ec  0x4f2e8c73  0x6123c8ba  0x50211945
     0xb6fd1653  0x3fe82e34  0x26c50cc9  0x1466d
     0xcc8b9b9a  0xff619d62  0xe81de7dc  0x58825999
     0x5d0bdf19  0x4f315aae  0xffffbff9  0x9cc4d49c

Common Message Digest:
     0x6d0938b9 0xd74dd4fe 0x96eaaad3 0xa1b649ec
```

- Cặp message 2:

```
Message 1 :
     0xc8b53517 0x225658df 0x775b60b4 0x6e27a9a5
     0x8b29f758 0x56a8fa0e 0x2d3c5d3b 0xd69d1288
     0xbf46e25b 0xffbee494 0x24993d9d 0x5a825999
     0x2361573c 0x208dcba6 0xffffbff9 0x9c829058

Message 2 :
     0xc8b53517 0x225658df 0x775b60b4 0x6e27a9a5
     0x8b29f758 0x56a8fa0e 0x2d3c5d3b 0xd69d1288
     0xbf46e25b 0xffbee494 0x24993d9d 0x5a825999
     0x2361573d 0x208dcba6 0xffffbff9 0x9c829058

Common Digest Message:

     0x75d3b5e3 0x5af24278 0xdc3f167e 0xc52ed933
```

3. Tổng quát hoá phương pháp tấn công MD4

Ta tìm 2 thông điệp X_i và X'_i (với $0 \le i \le 15$), được thiết lập như sau: $X_i = X'_i$, với $i \ne i_o$ và $X_{i0} = X'_{i0} + \delta$ (khác nhau ở một vài vị trí nhỏ). Thấy rằng nội dung các thanh ghi A, B, C, D là giống nhau từ bước 1 đến bước thứ i_0 (vì X_{i0} là đầu vào

của bước thứ i_0) của vòng 1. Giả sử X_{i0} sẽ lại là đầu vào ở bước m trong vòng 2. Ta chọn vị trí i_0 sao cho khoảng cách từ bước i_0 đến bước m là nhỏ nhất.

Với MD4 i_0 được chọn là 12, nghĩa là $X_i = X'_i$, với $i \neq i_{12}$ và $X_{12} = X'_{12} + \delta$. Ta chia tấn công thành 3 phần chính:

Phần 1: Tấn công vi sai modulo 2^{32}.
Đây là bước chuẩn bị cho việc tìm "inner almost-collision".
Mục đích: tìm Δ_{19} để $\Delta_{35} = 0$ với xác suất đủ lớn để có thể tấn công MD4. Giả sử ta tìm được $\Delta_{19} = (\Delta a, \Delta b, \Delta c, \Delta d)$ với xác suất thành công là p.

Phần 2: Tìm "inner almost-collision"
Với Δ_{19} tìm được, gọi A, B, C, D là giá trị khởi đầu cho bước 12 và A_i, B_i, C_i, D_i, A'_i, B'_i, C'_i, D'_i là nội dung thanh ghi sau bước thứ i. Để dễ quan sát, ta ký hiệu U, V,W, Z là nội dung của các thanh ghi A_{12}, D_{13}, C_{14}, B_{15} và U', V', W', Z' là nội dung của các thanh ghi A'_{12}, D'_{13}, C'_{14}, B'_{15}, tương tự A_*, B_*, C_*, D_* ,A'_*, B'_*, C'_*, D'_*là nội dung thanh ghi sau bước 19 khi đầu vào là X_i và X'_i. Theo định nghĩa về MD4 xét các bước từ 12 đến 19, ta có hệ phương trình sau:

$$\delta = U'^{<<29} - U^{<<29} \quad (1'')$$
$$F(U', B, C) - F(U, B, C) = V'^{<<25} - V^{<<25} \quad (2'')$$
$$F(V', U', B) - F(V, U, B) = W'^{<<21} - W^{<<21} \quad (3'')$$
$$F(W', V', U') - F(W, V, U) = Z'^{<<13} - Z^{<<13} \quad (4'')$$
$$G(Z', W'V') - G(Z, W, V) = U - U' - A_*^{<<29} + A'^{<<29}_* \quad (5'')$$
$$G(A'_*, Z', W') - G(A_*, Z, W) = V - V' - D_*^{<<27} + D'^{<<27}_* \quad (6'')$$
$$G(D'_*, A'_*, Z') - G(D_*, A_*, Z) = W - W' + C'^{<<23}_* - C_*^{<<23} \quad (7'')$$
$$G(C'_*, D'_*, A'_*) - G(C_*, D_*, A_*) = Z - Z' + B'^{<<19}_* - B_*^{<<19} - \delta \quad (8'')$$

và:

$$X_{13} = \text{tùy ý} \quad (9'')$$
$$X_{14} = W^{<<21} - C - F(V,U,B) \quad (10'')$$
$$X_{15} = Z^{<<13} - B - F(W,V,U) \quad (11'')$$
$$X_0 = A_*^{<<29} - U - G(Z,W,V) - K1 \quad (12'')$$
$$X_4 = D_*^{<<27} - V - G(A_*, Z, W) - K1 \quad (13'')$$
$$X_8 = C_*^{<<23} - W - G(D_*, A_*, Z) - K1 \quad (14'')$$
$$X_{12} = B_*^{<<19} - Z - G(C_*, D_*, A_*) - X_{13} \quad (15'')$$
$$D = V^{<<25} - F(U,B,C) - X_{13} \quad (16'')$$
$$A = U^{<<29} - F(B,C,D) - X_{12} \quad (17'')$$

Đặt U = -δ, U' = 0 và B = 0. Phương trình (1) đã thoả mãn, các biểu thức (2)-(8) được biến đổi tương đương và sắp xếp lại như sau:

$$Z' = Z - G(C'_*, D'_*, A'_*) + G(C_*, D_*, A_*) + B'^{<<19}_* - B^{<<19}_* - \delta \quad (18'')$$

$$W' = W - G(D'_*, A'_*, Z') + G(D_*, A_*, Z) + C'^{<<23}_* - C^{<<23}_* \quad (19'')$$

$$V = W'^{<<21} - W^{<<21} \quad (20'')$$

$$V' = V - G(A'_*, Z', W') + G(A_*, Z, W) + D'_* - D_* \quad (21'')$$

$$C = V^{<<25} - V'^{<<25} \quad (22'')$$

$$G(Z, W, V) - G(Z', W', V') = 1 + A'_* - A_* \quad (23'')$$

$$F(W', V', 0) - F(W, V, -1) - Z'^{<<13} + Z^{<<13} = 0 \quad (24'')$$

Như vậy, ta có hệ gồm 16 phương trình từ (9) đến (24) với 20 ẩn số là X_{12}, X_{13}, X_{14}, X_{15}, X_0, X_4, X_8 và A, C, D, W, Z, V, $\tilde{Z}, \tilde{W}, \tilde{V}$, A_*, B_*, C_*, D_*. Sử dụng thuật toán tìm hầu va chạm bên trong của Dobbertin (đã mô tả ở trên) để tìm giá trị của 20 biến trên thoả mãn hệ phương trình. Trong thuật toán tìm hầu va chạm, ta sẽ sử dụng các biểu thức sau:

$$F(W', V', 0) - F(W, V, -1) - Z'^{<<13} + Z^{<<13} \quad (25'')$$

và kiểm tra thêm một biểu thức sau:

$$G(B'_*, C'_*, D'_*) = G(B_*, C_*, D_*) \quad (26'')$$

Phần 3: **Tìm giá trị khởi đầu đúng**

Phần này hoàn toàn giống như đã trình bày ở mục 1.4 (phần II).

Cuối cùng ta cũng có thể đưa ra được thuật toán tổng quát để tìm va chạm cho MD4 như sau:

1. Tính A, B, C, D và X_{12}, X_{13}, X_{14}, X_{15}, X_0, X_4, X_8 để tìm 'inner almost-collision' (từ bước 12 đến 19). Thuật toán chi tiết được mô tả trong phần 2.

2. Chọn X_1, X_2, X_3, X_4 ngẫu nhiên và tính:

$$(A_5, B_5, C_5, D_5) = \text{compress}^{0}_{5}(IV_o; X_0, \ldots, X_5), \quad (27'')$$

$$t = A^{<<29} - A_5 - X_8, \quad (28'')$$

$$X_6 = t^{<<21} - C_5 - F(D_5, A_5, B_5), \quad (29'')$$

$$X_7 = -B_5 - F(t, D_5, A_5) - 1 \quad (30'')$$

$$X_9 = D^{<<25} - D_5 - F(A, -1, t), \quad (31'')$$

$$X_{10} = C^{<<21} - t - F(D, A, -1) \quad (32'')$$

$$X_{11} = B^{<<13} + 1 - F(C, D, A) \quad (33'')$$

$$(A_{35}, B_{35}, C_{35}, D_{35}) = \text{compress}^{20}_{35}(A_{19}, B_{19}, C_{19}, D_{19}); X), \quad (34'')$$

$$(\tilde{A}_{35}, \tilde{B}_{35}, \tilde{C}_{35}, \tilde{D}_{35}) = \text{compress}^{20}_{35}((\tilde{A}_{19}, \tilde{B}_{19}, \tilde{C}_{19}, \tilde{D}_{19}); X), \quad (35'')$$

3. Nếu $\Delta_{35} = 0$, thì chúng ta đã tìm được collision. Nếu không thì nhảy về bước 2.

Va chạm vi sai của SHA-0

Florent Chaboud và Antoine Joux (CRYPTO'98, LNCS 1462, pp. 56-71)

1. Mô tả về SHA

1.1 Tóm tắt lịch sử

Chuẩn hàm băm an toàn (SHS) [7] được Viện Tiêu chuẩn và Công nghệ Quốc Gia đưa ra năm 1993, đó là SHA-0. Nó xuất phát từ MD4 của Rivest [5]. MD4 ra đời năm 1990, còn phiên bản cải tiến của nó là MD5 ra đời năm 1991. Dù sao, một số khối của hàm SHA-0 khác với MD4, nhưng không có một giải thích rõ ràng về sự lựa chọn này. Hai năm sau đó, chuẩn được sửa chữa nhỏ, các hàm được thay đổi một chút [8], đó là SHA-1. Sự thay đổi này được yêu cầu để sửa một điểm yếu kỹ thuật trong SHA, nhưng không có một giải thích nào. Tuy vậy, NSA thông báo rằng họ tìm được một tấn công va chạm tốt hơn nghịch lý ngày sinh.

Một cách độc lập, một vài tấn công trên hàm MD4 gốc, và bản phát triển MD5 của nó [6] đã được công bố [2,4]. Dù sao, những tấn công này không thể áp dụng được với Thuật toán Băm An toàn (Secure Hash Algorithm) trong cả phiên bản thứ nhất và thứ hai bởi việc sử dụng phép mở rộng (expansion).

1.2 Ký hiệu

Các ký hiệu được sử dụng trong bài báo này được định nghĩa trong bảng 1. Bên cạnh đó, chúng tôi còn ký hiệu các ký tự hoa là các từ 32-bits, và $X^{(i)}$ viết tắt cho giá trị của X được dùng ở vòng thứ i của SHA.

Bảng 1: Các ký hiệu

Ký hiệu	Định nghĩa
F_q	Trường hữu hạn với q phần tử
(X,Y,..., Z)	Việc ghép nối các từ 32-bit
$+$	Cộng các từ 32-bit modulo 2^{32}.
$\oplus$	*Hoặc loại trừ* các bit hay các từ 32-bit
$\vee$	Phép toán *hoặc* các bit hay các từ 32-bit
$\wedge$	Phép toán logic and các bit hay các từ 32-bit.
$ROL_l(X)$	Quay l bit của từ 32-bit.
X_i	Bit thứ i của từ 32-bit X, từ trọng số thấp nhất là 0 đến trọng số cao nhất là 31.

1.3 Mô tả SHA

Mô tả hàm băm

Các hàm băm trong họ SHA thao tác với các khối message 512 bit và đầu ra là

một giá trị băm 160 bit. Giá trị băm này được thực hiện bằng việc ghép 5 thanh ghi 32 bit với nhau. Để băm một message, thực hiện một vài bước sau:

1. Đệm thêm vào message được băm bởi số 1, chuỗi số 0 phù hợp và 1 số nguyên 64 bit biểu diễn độ dài của message. Sau bước đệm thêm này, message gồm các khối 512 bit nguyên.
2. Khởi tạo 5 thanh ghi 32 bit A, B, C, D và E với các hằng số cố định:
   - A = 0x67452301
   - B = 0xEFCDAB89
   - C = 0x98BADCFE
   - D = 0x10325476
   - F = 0xC3D2E1F0
3. Với mỗi khối message, copy A, B, C, D và E tương ứng vào AA, BB, CC, DD, và EE. Áp dụng các hàm nén với AA, BB, CC, DD, EE và khối message, ta thu được AA', BB', CC', DD' và EE'. 5 giá trị này được cộng tương ứng với A, B, C, D và E.
4. Đầu ra là sự ghép nối nội dung các thanh ghi A, B, C, D và E.

Trong phần còn lại của bài báo, chúng ta cố gắng tìm các va chạm trên hàm nén, lúc đó va chạm trên hàm băm là tầm thường.

Mô tả hàm nén

Tiếp theo, chúng ta ký hiệu $(W^{(0)}, ...,W^{(15)})$ là 512 bit đầu vào của SHA, được thiết lập từ 16 từ 32 bit. Bước đầu tiên của SHA-0 là thực hiện mở rộng 512 bit này. Kết quả của sự mở rộng này cho ta quan hệ sau:

$$W^{(i)} = W^{(i-3)} \oplus W^{(i-8)} \oplus W^{(i-14)} \oplus W^{(i-16)}, \forall i, 16 \le i < 80. \quad (1)$$

80 từ 32 bit này được dùng để thay thế 5 từ 32-bit trạng thái ký hiệu bởi $A^{(i)}$, $B^{(i)}$, $C^{(i)}$, $D^{(i)}$, $E^{(i)}$. Trạng thái khởi đầu là đầu vào của hàm nén được ký hiệu là $(A^{(0)}$, $B^{(0)}$, $C^{(0)}$, $D^{(0)}$, $E^{(0)})$.

Sự thay đổi trạng thái $<A^{(i)}$, $B^{(i)}$, $C^{(i)}$, $D^{(i)}$,$E^{(i)}>$ được thực hiện bởi phép biến đổi sau, trong đó hàm $f^{(i)}$ và hằng số $K^{(i)}$ được xác định theo bảng 2, và $ADD(U,V,W,X,Y) = U + V + W + X + Y \pmod{2^{32}}$:

for i=0 to 79

$A^{(i+1)} = ADD(W^{(i)}, ROL_5(A^{(i)}), f^{(i)}(B^{(i)}, C^{(i)}, D^{(i)}), E^{(i)}, K^{(i)})$

$B^{(i+1)} = A^{(i)}$

$C^{(i+1)} = ROL_{30}(B^{(i)})$

$D^{(i+1)} = C^{(i)}$

$E^{(i+1)} = D^{(i)}$

Bảng 2: Định nghĩa hàm $f^{(i)}(X,Y,Z)$, và hằng số $K^{(i)}$.

Vòng i	Hàm $f^{(i)}$		Hằng số $K^{(i)}$
	Tên	Định nghĩa	
0-19	IF	$(X \wedge Y) \vee (\tilde{X} \wedge Z)$	0x5A827999
20-39	XOR	$(X \oplus Y \oplus Z)$	0x6ED9EBA1
40-59	MAJ	$(X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$	0x9F1BBCDC
60-79	XOR	$(X \oplus Y \oplus Z)$	0xCA62C1D6

Đầu ra của hàm nén là 160 bit nhận được trong trạng thái cuối cùng ($A^{(80)}$, $B^{(80)}$, $C^{(80)}$, $D^{(80)}$, $E^{(80)}$). Va chạm ở đây được hiểu theo nghĩa chuẩn là tìm 2 dãy từ đầu vào ($W^{(0)}$, ..., $W^{(15)}$) và ($W'^{(0)}$, ..., $W'^{(15)}$) mà cho cùng 160-bit đầu ra ($A^{(80)}$, $B^{(80)}$, $C^{(80)}$, $D^{(80)}$,$E^{(80)}$), sử dụng cùng giá trị ban đầu ($A^{(0)}$, $B^{(0)}$, $C^{(0)}$, $D^{(0)}$, $E^{(0)}$).

Kiến trúc cơ bản của SHA có thể được chỉ ra bởi hình 1. Hộp mở rộng được xem như một ứng dụng tuyến tính từ $(F_2)^{512}$ vào $(F_2)^{2560}$, ánh xạ ($W^{(0)}$, ..., $W^{(15)}$) sang ($W^{(0)}$, ..., $W^{(79)}$). Ánh xạ tuyến tính này là sự khác biệt duy nhất giữa phiên bản đầu tiên và phiên bản thứ 2 của SHA. Chính xác hơn, sự mở rộng SHA-1 thu được bằng việc thay (1) bằng biểu thức sau, khác với (1) bởi quay sang trái 1 bit:

$$W^{(i)} = ROL_1(W^{(i-3)} \oplus W^{(i-8)} \oplus W^{(i-14)} \oplus W^{(i-16)}), \forall i, 16 \le i < 80. \qquad (2)$$

Chúng ta ký hiệu E_0 mở rộng ban đầu được mô tả bởi (1), và E_1 là mở rộng đã sửa đổi được mô tả bởi (2). Kiến trúc chung này định nghĩa họ các hàm băm bằng việc thay đổi hộp mở rộng.

2. Lan truyền các xáo trộn cục bộ trong các hàm băm dạng SHA

2.1 Các phiên bản SHA yếu

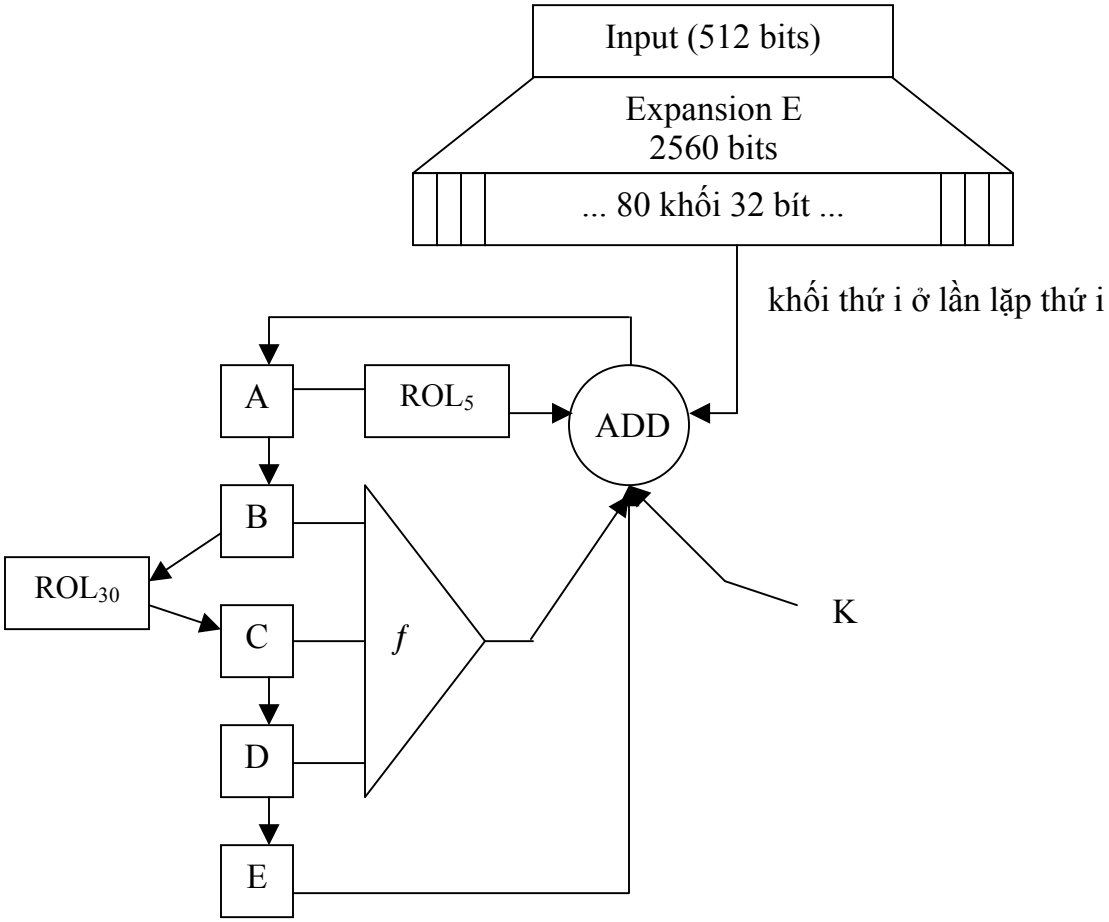
Kiến trúc nguyên thể của SHA

Trước tiên chúng ta nghiên cứu sự lan truyền các xáo trộn cục bộ trong phiên bản tuyến tính hoàn toàn của SHA, để phân biệt giữa một bên là vai trò của kiến trúc nguyên thể các hàm băm và một bên là vai trò của các khối xây dựng cơ bản. Trong hàm nén của hàm băm thuộc họ SHA có 2 nguồn gốc sinh ra tính phi tuyến, đó là các hàm $f^{(i)}$ và hàm ADD.

Do đó, hàm băm đầu tiên chúng ta xét đến là SHI1[1] – hàm nén trong họ SHA được xây dựng từ SHA-0 (do đó sử dụng mở rộng E_0) và bằng việc thay thế hàm ADD bằng phép XOR với 5 biến, và tất cả $f^{(i)}$ là các hàm XOR.

[1] SHI1 là cách chơi chữ của tiếng Pháp gồm những chú chó và chú mèo

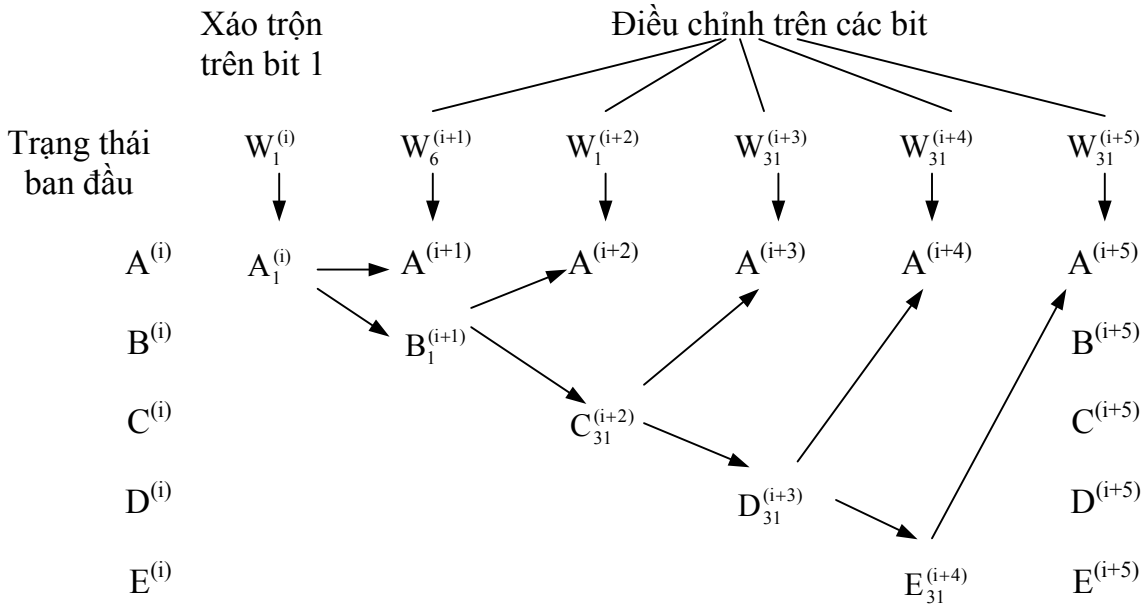
Chúng ta ký hiệu như thông thường với $W^{(i)}$ là từ thứ i của mở rộng ($0 \le i < 80$), và 32 bit của từ này được đánh số là $W_0^{(i)}, ..., W_{31}^{(i)}$.



Hình 1: Kiến trúc SHA

Bây giờ chúng ta giảm các ràng buộc với vector W và tạm thời quên đi rằng nó nhận được từ kết quả của phép mở rộng. Do đó, chúng ta có thể áp dụng sự xáo trộn cục bộ bất kỳ trên một bit bất kỳ của W. Ví dụ, chúng ta có thể phủ định giá trị của W_1^i. Thay đổi này sẽ sửa (correct) bit 1 của $A^{(i+1)}$, bit 1 của $B^{(i+2)}$, bit 31 của $C^{(i+3)}$, bit 31 của $D^{(i+4)}$ và cuối cùng là bit 31 của $E^{(i+5)}$. Nếu chúng ta muốn ngăn cản sự thay đổi hơn nữa, chúng ta cần phủ định các giá trị của bit W_6^{i+1}, W_1^{i+2}, W_{31}^{i+3}, W_{31}^{i+4}, W_{31}^{i+5}. Những điều chỉnh mới này ngăn cản sự thay đổi trên bit 1 của $A^{(i+1)}$ làm thay đổi bit 6 của $A^{(i+2)}$, sự thay đổi bit 1 của $B^{(i+2)}$ làm thay đổi bit 1 của $A^{(i+3)}$, sự thay đổi bit 31 của $C^{(i+3)}$ làm thay đổi bit 31 của $A^{(i+4)}$, sự thay đổi bit 31 của $D^{(i+4)}$ làm thay đổi bit 31 của $A^{(i+5)}$, sự thay đổi bit 31 của $E^{(i+5)}$ làm thay đổi bit 31 của $A^{(i+6)}$. Do đó, phủ định W_1^i, W_6^{i+6}, W_1^{i+2}, W_{31}^{i+3}, W_{31}^{i+4} và W_{31}^{i+5} cho ta 2

đường khác nhau từ $A^{(i)}$, $B^{(i)}$, $C^{(i)}$, $D^{(i)}$ và $E^{(i)}$ tới $A^{(i+6)}$, $B^{(i+6)}$, $C^{(i+6)}$, $D^{(i+6)}$ và $E^{(i+6)}$, và ta thu được một va chạm cục bộ. Điều này được tổng quát hoá trong Hình 2.



Hình 2: Sự lan truyền xáo trộn của SHI1

Chú ý 1: Rõ ràng rằng những gì chúng ta nói với bit 1, có thể tổng quát cho bit bất kỳ khác từ 0 tới 31. Dù sao, mọi chuyện sẽ trở nên rõ ràng trong phần sau (xem phần 1), sự lựa chọn này là tốt nhất cho mục đích của chúng ta. Vì vậy, chúng ta sẽ tập trung vào giá trị này trong suốt phần còn lại của bài báo này.

Vì tất cả đều tuyến tính, nên chúng ta có thể áp dụng đồng thời nhiều va chạm cục bộ khi chúng ta muốn và thu được 2 đường (path) khác nhau từ $A^{(0)}$, $B^{(0)}$, $C^{(0)}$, $D^{(0)}$ và $E^{(0)}$ tới $A^{(80)}$, $B^{(80)}$, $C^{(80)}$, $D^{(80)}$ và $E^{(80)}$. Đường đầu tiên sử dụng W nguyên gốc và đường thứ 2 sử dụng giá trị đã sửa, gọi là W'. Đến đây nảy sinh một câu hỏi là "Chọn các va chạm cục bộ thế nào để quay trở lại với điều kiện là cả W và W' là kết quả của quá trình mở rộng?".

Việc chọn các va chạm cục bộ chỉ đơn giản là xây dựng một vector lỗi (error vector) m_0 gồm 80 bit (được đánh số từ 0 đến 79) với 1 trong vị trí i nếu ta muốn phủ định $W_1^{(i)}$. Dù sao, ta cũng không thể chọn phủ định $W_1^{(i)}$, với $i \geq 75$, vì xáo trộn trong vòng thứ i không bao giờ được điều chỉnh trước vòng i+6, và tất cả các xáo trộn phải được điều chỉnh xong ở vòng thứ 80.

Giả sử $\left(m_0^{(0)},...,m_0^{(79)}\right)$ là một trong các vector lỗi này. Từ đó, ta suy ra *perturbative mask* (tạm dịch là mặt che xáo trộn) trên W, $M_0 = \left\langle M_0^{(-5)},...,M_0^{(79)} \right\rangle$ được định nghĩa như sau:

$\forall$i, -5 ≤ i ≤ -1, $M_0^{(i)} = 0$.

$\forall$i, 0 ≤ i ≤ 79, $M_{0,k}^{(i)} = 0$ nếu k ≠ 1;

$\forall$i, 0 ≤ i ≤ 79, $M_{0,1}^{(i)} = m_0^{(i)}$.

Corrective mask (mặt che điều chỉnh) được kết thúc bởi 5 khối toàn zeroo, bởi vì *corrective mask* bây giờ được suy ra từ *perturbative mask* này bằng cách dịch và quay.

Corrective mask đầu tiên M_1 được suy từ M_0 bằng cách dịch đi một vòng, và quay sang trái 5 bit. Phép quay này được mô tả trong phép biến đổi của SHA (Xem phần 1.3). Vì vậy, nó áp dụng trên các bit có số k=6. Ta có:

$$\forall i, -4 \le i \le 79,\ M_1^{(i)} = ROL_5\left(M_0^{(i-1)}\right) \quad (3)$$

Corrective mask thứ hai M_2 được suy từ M_0 bằng cách dịch 2 vòng và không quay (xem Hình 2).

$$\forall i, -3 \le i \le 79,\ M_2^{(i)} = M_0^{(i-2)} \quad (4)$$

Tương tự, M_3, M_4, M_5 được suy ra từ M_0 tương ứng với việc dịch 3, 4, 5 vòng và áp dụng trên các bit đánh số k = 31.

$$\forall i, -2 \le i \le 79,\ M_3^{(i)} = ROL_{30}\left(M_0^{(i-3)}\right) \quad (5)$$

$$\forall i, -1 \le i \le 79,\ M_4^{(i)} = ROL_{30}\left(M_0^{(i-4)}\right) \quad (6)$$

$$\forall i, 0 \le i \le 79,\ M_5^{(i)} = ROL_{30}\left(M_0^{(i-5)}\right) \quad (7)$$

Bây giờ, điều ta cần là *global differential mask* (mặt che vi sai tổng) M được định nghĩa bởi:

$$\forall i, 0 \le i \le 79,\ M^{(i)} = M_0^{(i)} \oplus M_1^{(i)} \oplus M_2^{(i)} \oplus M_3^{(i)} \oplus M_4^{(i)} \oplus M_5^{(i)}, \quad (8)$$

phải là đầu ra của E_0.

Điều kiện này xảy ra nếu tất cả các mặt che M_k thoả mãn (1), bằng việc kiểm tra xem *perturbative mask* ban đầu có thoả mãn biểu thức sau không:

$$M_0^{(i)} = M_0^{(i-3)} \oplus M_0^{(i-8)} \oplus M_0^{(i-14)} \oplus M_0^{(i-16)}, \forall i, 11 < i < 80, \quad (9)$$

Hơn nữa, vì E_0 không đan xen các bits (xem (1)), chúng ta có thể chia bảng mở rộng thành 32 hộp giống nhau e_0 mở rộng 16 bit thành 80 bit, và định nghĩa (1) được xem như thực hiện trên các bit. Hộp e_0 là đủ nhỏ cho phép liệt kê vét cạn. Số các mặt che (mask) có thể là tương đối nhỏ, như là chỉ có 128 trong số $2^{16} = 65536$

đầu vào có thể, mà thoả mãn (9) và ràng buộc 5 số không (0) cho các vòng từ 75 tới 79, và khi đó ta được mặt che m_0.

Với một mặt che cho trước, ta có thể tìm được M, và bằng cách tính ngược biến đổi tuyến tính E_0, ta tìm được 512 bit mặt che đầu vào tương ứng μ sao cho $M = E_0(\mu)$. Do các hộp mở rộng của hàm SHA được mã (coded) một cách có hệ thống, rõ ràng là $\mu = <M^{(0)}, ..., M^{(15)}>$. Với tất cả các đầu vào $W = <W^{(0)}, ..., W^{(15)}>$, $W' = W \oplus \mu$ có cùng đầu ra thông qua hàm nén tuyến tính SHI1.

Đưa vào các hàm phi tuyến

Từ tất định tới phương pháp xác suất: Bây giờ chúng ta muốn nghiên cứu sự va chạm của các hàm phi tuyến $f^{(i)}$ để đánh giá sự an toàn của hàm băm trong họ SHA. Ta xét hàm thứ hai SHI2, hàm nén trong họ SHA được xây dựng từ SHA-0 (sử dụng mở rộng E_0) và thay thế hàm ADD bằng hàm XOR trên 5 biến. Hàm này có thể xem như SHI1 với các hàm phi tuyến $f^{(i)}$. Dễ dàng thấy rằng trong một số trường hợp $f^{(i)}$ hành động giống như hàm XOR. Do đó, tấn công trước đó có thể sử dụng được. Câu hỏi nảy sinh ở đây là "Khi nào sử dụng được?" và "Xác suất thành công như thế nào?".

Để tính toán xác suất chúng ta cần phân tích chi tiết các hàm IF và MAJ. Do các hàm này thực hiện song song trên 32 bit, nên ta chỉ cần nghiên cứu trên 1 bit đơn lẻ. Giả sử chúng ta nghiên cứu dáng điệu của phép biến đổi từ $f^{(i)}(B^{(i)}, C^{(i)}, D^{(i)})$ vào $f^{(i)}(B'^{(i)}, C'^{(i)}, D'^{(i)})$, bằng cách quan sát cẩn thận các phép quay và mô hình xáo trộn ta có thể thấy các trường hợp khác nhau sau có thể xuất hiện:

1. Không có sự thay đổi ở tất cả các đầu vào, tức là: $B^{(i)} = B'^{(i)}$, $C^{(i)} = C'^{(i)}$, $D^{(i)} = D'^{(i)}$. Trong trường hợp này đầu ra không thay đổi $f(B^{(i)}, C^{(i)}, D^{(i)}) = f(B'^{(i)}, C'^{(i)}, D'^{(i)})$ và $f^{(i)}$ được coi như phép XOR.
2. Có sự khác nhau duy nhất trên các đầu vào tại bit 1 của biến $B^{(i)}$, tức là: $B'^{(i)} = B^{(i)} \oplus 2^1$. Trong trường hợp này $f^{(i)}$ hành động như XOR khi và chỉ khi $f^{(i)}(B^{(i)}, C^{(i)}, D^{(i)}) = f^{(i)}(B'^{(i)}, C'^{(i)}, D'^{(i)}) \oplus 2^1$.
3. Có sự khác nhau duy nhất trên các đầu vào tại bit 31 của biến $C^{(i)}$ hoặc $D^{(i)}$ (XOR). Trong trường hợp này $f^{(i)}$ hành động như phép XOR khi và chỉ khi $f^{(i)}(B'^{(i)}, C'^{(i)}, D'^{(i)}) = f^{(i)}(B^{(i)}, C^{(i)}, D^{(i)}) \oplus 2^{31}$.
4. Có hai khác biệt trong các đầu vào tại bit 31của các biến $C^{(i)}$ và $D^{(i)}$, nghĩa là ta có $C'^{(i)} = C^{(i)} \oplus 2^{31}$ và $D'^{(i)} = D^{(i)} \oplus 2^{31}$. Trong trường hợp đó, $f^{(i)}$ hành động như phép XOR khi và chỉ khi đầu ra của $f^{(i)}$ không thay đổi: $f^{(i)}(B'^{(i)}, C'^{(i)}, D'^{(i)}) = f^{(i)}(B^{(i)}, C^{(i)}, D^{(i)})$.

Bây giờ ta hãy nhìn vào 3 trường hợp cuối cùng với các hàm MAJ và IF. Với hàm MAJ, trường hợp 2 và 3 là giống nhau, đầu ra thay đổi khi và chỉ khi 2 bit đầu vào không thay đổi là ngược nhau (opposite). Điều này xảy ra với xác suất là 1/2.

Trong trường hợp 4, đầu ra không thay đổi khi và chỉ khi 2 bit đầu vào $C_{31}^{(i)}$ và $D_{31}^{(i)}$ thay đổi theo hướng ngược nhau. Điều này xảy ra với xác suất 1/2.

Đối với hàm IF, trong trường hợp 2 đầu ra thay đổi khi và chỉ khi các bit $C_{31}^{(i)}$ và $D_{31}^{(i)}$ là ngược nhau. Điều này xảy ra với xác suất 1/2. Trong trường hợp 3, đầu ra thay đổi khi và chỉ khi bit $B_{31}^{(i)}$ dùng để chỉ ra bit được thay đổi (tức là: $B_{31}^{(i)}$=1 nếu $C'^{(i)}_{31} = C_{31}^{(i)} \oplus 2^{31}$ thay đổi và $B_{31}^{(i)}$=0 nếu $D'^{(i)}_{31} = D_{31}^{(i)} \oplus 2^{31}$ thay đổi), điều này xảy ra với xác suất 1/2. Trong trường hợp 4, đầu ra luôn luôn thay đổi, vì vậy xác suất của điều mong muốn là 0. Điều này có nghĩa rằng, chúng ta cần chọn mẫu xáo trộn sao cho không có 2 xáo trộn liền nhau trong các vòng IF. Chính xác hơn, do các vòng IF xuất hiện từ vòng 0 đến 19 (xem bảng 2), và trường hợp 4 gồm các trạng thái $C_{31}^{(i)}$ và $D_{31}^{(i)}$, nên không có 2 xáo trộn liền nhau có thể xuất hiện trước vòng 16, nhưng có thể có 2 xáo trộn liền nhau trong các vòng 16 và 17, bởi vì sự lan truyền lỗi sẽ xuất hiện với $C^{(i)}$ và $D^{(i)}$ trên vòng 20 (xem Hình 2).

Với tất cả các ràng buộc ở trên, ta có thể tìm được một mẫu với xác suất thành công tổng khoảng $1/2^{24}$. Sau đây chúng ta biểu diễn 80 bit đầu ra tương ứng của hộp e_0. 5 số không (0) ở trước chỉ để nhắc lại rằng mẫu này thoả mãn các ràng buộc được phát triển trong phần 1:

00000 0010001000000010111
01100011100000010100
01000100100100111011
00110000111110000000

Mẫu m_0 này được kết thúc và bắt đầu bởi 5 số 0, và không có 2 xáo trộn liền kề nhau trong 16 vòng đầu tiên.

Bằng cách xây dựng tương tự như mô tả trong phần 1, chúng ta thu được *differential mask* mà có thể áp dụng trên từ đầu vào, và cho một va chạm với xác suất không đáng kể. Chúng ta gọi mặt che này là $\mathcal{M}$.

Việc ước lượng xác suất thành công khá phức tạp, vì 16 vòng đầu tiên không bao gồm trong ước lượng này. Lý do vì sao như thế sẽ xuất hiện khi cài đặt (implement) tìm kiếm va chạm.

Cài đặt tìm kiếm va chạm: Bây giờ chúng ta đã có *differential mask* $\mathcal{M}$ mà chúng ta có thể thử áp dụng trên một từ đầu vào bất kỳ $W = <W^{(0)}, ..., W^{(15)}>$. Để kiểm tra xem có va chạm hay không, người ta phải kiểm tra với tất cả các xáo trộn, nếu sự điều chỉnh được thực hiện tốt, ta nói rằng, hàm $f^{(i)}$ được hành động giống như XOR. Vì mỗi xáo trộn xuất hiện trong ba hàm $f^{(i)}$ (liền nhau) khác nhau, ta cần

phải xét nhiều xác suất sơ đẳng. Trong ví dụ của chúng tôi, có các xáo trộn trong các vị trí 2, 6, 14, 16, 17, 18, 19, 21, 22, 26, 27, 28, 35, 37, 41, 45, 48, 51, 54, 55, 56, 58, 59, 62, 63, 68, 69, 70, 71 và 72. Bảng 3 chỉ trong từng trường hợp mỗi xáo trộn có liên quan tới cho từng bộ ba $f^{(i)}$ tham gia vào.

***Chú ý 2**:* Trong bảng 3, trường hợp 4 với hàm MAJ được tính cho xác suất $1/\sqrt{2}$ cho một trong 2 xáo trộn trong đó. Theo cách này, ta thu được xác suất tổng toàn bộ bằng 1/2 như đã thấy ở trên.

Bảng 3: Xác suất thành công của mặt che $\mathcal{M}$ trong mô hình SHI2.

Xáo trộn trong vòng i	$f^{(i+2)}$	t/hợp	$f^{(i+3)}$	t/hợp	$f^{(i+4)}$	t/hợp	xác suất tổng	logarithm xác suất
2	IF	2	IF	3	IF	3	1/8	3
6	IF	2	IF	3	IF	3	1/8	3
14	IF	2	IF	3	IF	3	1/8	3=2+1 (xem Chú ý 3)
16	IF	2	IF	3	XOR	-	¼	2
17	IF	2	XOR	-	XOR	-	½	1
18,19,21 22,26,27 28,35	XOR	-	XOR	-	XOR	-	1	0
37	XOR	-	MAJ	3	MAJ	3	¼	2
41	MAJ	2	MAJ	3	MAJ	3	1/8	3
45	MAJ	2	MAJ	3	MAJ	3	1/8	3
48	MAJ	2	MAJ	3	MAJ	3	1/8	3
51	MAJ	2	MAJ	3	MAJ	3	1/8	3
54	MAJ	2	MAJ	3	MAJ	4	$¼\sqrt{2}$	2.5
55	MAJ	2	MAJ	4	MAJ	4	¼	2
56	MAJ	2	MAJ	4	XOR	-	$½\sqrt{2}$	1.5
58,59,62 63,68,69 70,71,72	XOR	-	XOR	-	XOR	-	1	0

Do từ đầu vào được truyền mà không có sự sửa đổi thông qua bảng mở rộng, có thể chia tìm kiếm thành 2. Trước tiên, ta tìm $W^{(0)}...W^{(14)}$ sao cho hàm $f^{(i)}$ hành động giống như XOR khi mặt che được áp dụng. Điều này xảy ra với xác suất $1/2^6$, khi hai xáo trộn tham gia vào ở tại các vị trí thứ 2 và 6.

Sau đó, $W^{(0)}...W^{(14)}$ được giữ cố định, chúng ta có thể thử nhiều giá trị của $W^{(15)}$ (tất nhiên ta phải thử ít hơn 2^{32}, trong thực hành một số lớn bất kỳ chẳng hạn

10000 là thoả mãn). Một $W^{(15)}$ như vậy có thể dẫn tới một va chạm sau 80 vòng nếu tất cả các vòng khác đều cư xử tốt. Như ta đã thấy trong bảng 3, điều này xảy ra với xác suất $1/2^{26}$. Vì phần đầu của xây dựng được thực hiện 1 lần cho nhiều $W^{(15)}$, xác suất thứ hai đưa ra giá trị thực của việc liệt kê.

***Chú ý 3**:* Ước lượng đầu tiên này cho ta xác suất tổng bằng $1/2^{26}$ thay cho xác suất yêu cầu. Nhưng chúng ta có thể cải tiến phương pháp này hơn nữa và bỏ đi một số xác suất từ xáo trộn của vòng 14. Hàm đầu tiên liên quan tới xáo trộn này là hàm IF trong vòng 16. Hàm này hành động tốt nếu các bit $C_1^{(16)}$ và $D_1^{(16)}$ khác nhau. Các bit này được biết trong vòng 14, vì chúng được copy từ $A_3^{(14)}$ và $A_3^{(13)}$. Điều này cho phép chúng ta chuyển xác suất bằng 1/2 từ phần hai của liệt kê sang phần thứ nhất. Điều này làm giảm xác suất xuống $1/2^{25}$.

Hàm thứ hai liên quan tới xáo trộn của vòng 14 là hàm IF trong vòng 17. Hàm này xử sự tốt nếu bit $B_{31}^{(17)}$ là 1. Vì bit này được copy từ $A_{31}^{(16)}$, ta có thể kiểm tra sự điều chỉnh của nó ngay sau khi chọn $W^{(15)}$, và nếu cần thiết thay đổi bit 31 của $W^{(15)}$ trước khi bắt đầu kiểm tra. Việc này làm giảm xác suất xuống như công bố là $1/2^{24}$.

***Chú ý 4**:* Trong trường hợp SHI2, tìm kiếm va chạm là rất nhanh và có thể thực hiện dưới nửa phút. Dưới đây là một va chạm ví dụ:

```
1a6191b0  3c4a331c  1f228ea2  403b7609
062ec496  48611ca8  583401bc  399879d0
2270fdbd  2a8090f0  4b12fd98  473cc7a1
002831a9  50fe1535  61ac0d3d  f26700ec
```

và

```
1a6191b0  3c4a331c  1f228ea0  403b7649
062ec494  c8611ca8  d83401be  b9987990
2270fdbd  aa8090f0  cb12fd98  c73cc7a1
002831a9  50fe1535  61ac0d3f  f26700ac
```

Cả 2 đều cho giá trị băm sau 80 vòng của hàm SHI2:

```
1334f224  21a3efc9  b667d2b2  2890013b  56013ca9
```

Đưa vào phép cộng

Cuối cùng, trước khi tiếp tục với SHA-0 và SHA-1, chúng ta nghiên cứu ảnh hưởng của phép cộng ADD trên lược đồ tấn công của chúng ta. Ta xét một hàm thứ ba SHI3, là hàm nén trong họ SHA được xây dựng từ SHA-0 (do đó sử dụng

mở rộng E_0) và bằng việc thay thế các hàm phi tuyến IF và MAJ bằng hàm XOR. Cái đó có thể xem như SHI1 với phép cộng ADD được đặt sang một bên.

Điểm mới ở đây là xáo trộn có thể dẫn tới có nhớ. Nếu ta có thể ngăn cản điều này xảy ra, thì tất cả sẽ hành động tốt như trước đó. Đầu tiên, dường như mỗi bit xáo trộn và mỗi bit điều chỉnh có thể dẫn tới nhớ. Điều này có ý rằng một xác suất cơ bản là $1/2^6$ cho mỗi xáo trộn, và do đó không cho ta tấn công khả dụng nào. Tuy nhiên, nên nhớ rằng chúng ta chọn để áp dụng xáo trộn trên bit 1 của $W^{(i)}$ nên có 3 điều chỉnh trên các bit ở vị trí 31 (W_{31}^{i+3}, W_{31}^{i+4} và W_{31}^{i+5}) . Vì không thể có bit nhớ nào từ bit 31, điều này chia đôi logarithm của xác suất cơ bản, điều này sẽ giải thích cho sự lựa chọn của chúng tôi ở trên.

Chúng ta có thể rút gọn sự kiện này hơn nữa, giả sử rằng $W_1^{(i)}$ là 0 và nó bị chuyển thành 1 trong $W'^{(i)}_1$, nếu không có bit nhớ xuất hiện (xác suất 1/2) thì $A_1^{(i+1)}$ là 0 (và $A'^{(i+1)}_1$ là 1). Tiếp theo sự thay đổi này trong tính toán của $A^{(i+2)}$, ta thấy rằng $W_6^{(i+1)}$ sẽ là 1 (và $W'^{(i+1)}_6$ là 0), ngược lại thì sự điều chỉnh đều dẫn tới nhớ. Nếu điều kiện này xảy ra thì điều chỉnh sẽ luôn luôn xảy ra mà không có nhớ. Điểm khó nhất là điều chỉnh thay đổi trong tính toán của $A^{(i+3)}$. Như trước nói, chúng ta chọn cố định $W_1^{(i+2)}$ là 1 (và $W'^{(i+2)}_1$ là 0). Sau đó việc điều chỉnh sẽ hành động tốt nếu bit đầu tiên trong kết quả của hàm XOR bằng với $B_1^{(i+2)}$ (tức là $A_1^{(i+1)}$). Điều này đúng mỗi khi $C_1^{(i+2)} = D_1^{(i+2)}$ (với xác suất 1/2).

Cũng chính các lập luận này chỉ ra rằng các xác suất cũng sẽ như thế khi $W_1^{(i)}$ là 1 (và thay đổi thành 2 trong $W'^{(i)}_1$). Thực ra, vấn đề quan trọng là thay đổi từ 0 sang 1 (tăng) cần phải được điều chỉnh bằng thay đổi từ 1 sang 0 (giảm) và thay đổi từ 1 sang 0 phải được điều chỉnh bằng việc thay đổi từ 0 sang 1. Xác suất cơ bản để xét được tạo nên từ thừa số 1/2 để bảo đảm rằng xáo trộn ban đầu không gây ra có nhớ, và thừa số 1/2 khác để bảo đảm rằng XOR giữ thay đổi theo cùng hướng.

Có hai rắc rối kỹ thuật nảy sinh trong trường hợp này, rắc rối thứ nhất là chúng ta cần xây dựng W theo cách mà $W_1^{(i)}$, $W_6^{(i+1)}$ và $W_1^{(i+2)}$ sẽ thoả mãn các ràng buộc (phi tuyến) ở trên. Vì E_0 không đan xen các bit, chúng ta xây dựng W_1 và W_6 ngay khi bắt đầu và giữ chúng cố định cho đến cuối. Điều rắc rối thứ hai đến là do không gì ngăn cản chúng ta nhận được sự thay đổi trong $W_1^{(i)}$, và một thay đổi khác trong $W_1^{(i+2)}$, trong trường hợp này ta thu được các điều kiện khác nhau trên W_1 và W_6 nhưng xác suất cơ bản bằng 1/4 vẫn xảy ra.

Trong thực hành, chúng tôi đã có thể tìm được một mẫu với xác suất $1/2^{44}$ (được tính như trong trong trường hợp SHI2)[2]. Mẫu này là:

0000 01000010100100011110

0101100000111000000

0000110000011011000

0001100010110110000

và chúng ta sẽ ký hiệu $\mathcal{M}'$ là mặt che vi sai gán với nó.

***Chú ý 5**:* Trong mẫu thứ hai này, ta không có điều kiện trên các xáo trộn liền kề, vì ta coi $f^{(i)}$ luôn luôn là hàm XOR. Do đó, ta có thể chú ý rằng mẫu này có 2 bit xáo trộn liền kề trên các vòng 15 và 16.

Kết hợp với mẫu này, các điều kiện trên bit 1 và 6 của W và mở rộng E_0 bắt ta chọn các giá trị sau cho những bit này:

Bit 1: 01110010000000011000

1010110101111000110

1101010111110110101 0

0000100111110101011 1

Bit 6: 00010000000110100000

1011000110100111001 1

0110110101111100001 0

0000101111011011101 11

***Chú ý 6**:* Sau một vài ngày tính toán, chúng tôi có thể tìm được một va chạm cụ thể cho SHI3:

53c29e14	44fe051b	4a8ce882	576e1943
0c0abc30	3806260d	76cbeb2f	1b8379a8
0da433ac	6337b011	1041e2a9	20b44364
1a3f8b30	8e7a4622	25e81245	289acb2b

và

[2] Ta có thể cải tiến tiến trình liệt kê tác động các xáo trộn của vòng 16 và 17 và kết hợp với điều chỉnh để thành công. Chi tiết về cải tiến khá phức tạp nên chúng tôi không trình bày ở đây, nhưng sẽ đăng trong một bài báo khác. Điều này dẫn tới thời gian thực hiện là 2^{40}, đã được kiểm tra lại bằng chương trình của chúng tôi.

53c29e14 44fe0519 4a8ce8c2 576e1941
8c0abc30 b806260d f6cbeb2d 1b8379a8
0da433ac e337b051 9041e2ab 20b44366
9a3f8b30 8e7a4622 a5e81245 a89acb29

Cả 2 đều cho giá trị băm sau 80 vòng của hàm SHI3 như sau:

983d1f8e e619f190 2e94fa09 0b0d479c 4c536e3e

2.2 Trường hợp SHA-0 thực sự

Chúng ta đã nghiên cứu SHI1, SHI2 và SHI3, bây giờ ta quay trở lại với trường hợp SHA-0. Trong trường hợp này, tất cả các xáo trộn đã được đưa vào mà không có nhớ, như trong SHI3. Hơn nữa, ta cần khảo sát sâu hơn trong phân tích các hàm IF và MAJ, mà chúng ta đã tiến hành với SHI2.

Chúng ta bắt đầu với hàm IF. Như trong SHI2, ta phải xét trường hợp 2, 3 và 4. Trường hợp 4 luôn không thể chấp nhận được trong mẫu tấn công. Trong trường hợp 3, mọi việc vẫn còn nguyên như vậy: thay đổi phải thông qua hàm IF, và nó xảy ra với xác suất 1/2. Trong trường hợp 2, thay đổi phải thông qua hàm IF. Hơn nữa, như trong trường hợp SHI3, hướng của nó phải được giữ nguyên. Hai điều kiện này được thoả mãn với xác suất 1/4.

Với hàm MAJ, ta có nhận xét rằng MAJ không bao giờ đảo hướng thay đổi, vì vậy trường hợp 2 và 3 vẫn không thay đổi, và mỗi trường hợp dẫn tới một xác suất bằng 1/2. Tuy nhiên, trường hợp 4 đưa tới một thay đổi thú vị.

Một điều mới, so với SHI2, là như trong SHI3, ta có thêm các tính chất sau:

$$C_{31}^{(i+3)} = A_1^{(i+1)} = \mathrm{W}_1^{(\mathrm{i})},$$
$$C_{31}^{(i+4)} = A_1^{(i+2)} = \mathrm{W}_1^{(\mathrm{i+1})}.$$

Điều này có nghĩa rằng trong trường hợp 4, MAJ hành động như XOR ngay khi biểu thức sau xảy ra,

$$W_1^{(i)} \neq W_1^{(i+1)} \qquad (10)$$

bởi vì kết quả của MAJ không thay đổi nếu và chỉ nếu $C_{31}^{(i+3)}$ và $D_{31}^{(i+4)}$ thay đổi ngược chiều nhau. Vì vậy, khi có xáo trộn trong vòng i và i+1 với $36 \leq i \leq 55$, nếu ta thêm ràng buộc (10) lên W_1, thì xác suất cơ bản của trường hợp 4 cho hàm MAJ là 1. Các điều kiện này được thêm vào điều kiện trước đó như mô tả trong SHI3, khi xây dựng W_6.

Kết hợp tất cả các ràng buộc này lại, ta có thể tìm được hai mẫu tốt, với xác suất thành công $1/2^{68}$ (tương ứng với $1/2^{69}$). Các mẫu này là:

0000 00010000000100100000
00100001101101111110
11010010000101010010
10100010111001100000 c = 68

0000 00100010000000101111
01100011100000010100
01000100100100111011
00110000111110000000 c = 69

Ta có thể xây dựng các mặt che vi sai (*differential mask*) được suy ra từ mỗi mẫu theo cách xây dựng trong phần 1. Mẫu thứ hai đã được ký hiệu là M trong phần 1.Chúng ta ký hiệu mẫu thứ nhất là M''.

Chú ý 7: Việc tính toán các xác suất có thể thực hiện từ bảng 5 và 4. Như đã trình bày trong Chú ý 3, xáo trộn trong vòng 14 nằm trên ranh giới giữa hai liệt kê. Đóng góp của nó vào xác suất thành công tổng là 1/2.

Bảng 4: Xác suất thành công của mặt che M cho SHA-0

Xáo trộn trong vòng i	$f^{(i+2)}$	t/hợp	$f^{(i+3)}$	t/hợp	$f^{(i+4)}$	t/hợp	xác suất tổng	logarithm xác suất
2	IF	2	IF	3	IF	3	1/32	5
6	IF	2	IF	3	IF	3	1/32	5
14	IF	2	IF	3	IF	3	1/32	4+1
16	IF	2	IF	3	XOR	-	1/16	4
17	IF	2	XOR	-	XOR	-	1/8	3
18,19,21 22,26,27 28,35	XOR	-	XOR	-	XOR	-	1/4	2
37	XOR	-	MAJ	3	MAJ	3	1/16	4
41	MAJ	2	MAJ	3	MAJ	3	1/16	4
45	MAJ	2	MAJ	3	MAJ	3	1/16	4
48	MAJ	2	MAJ	3	MAJ	3	1/16	4
51	MAJ	2	MAJ	3	MAJ	3	1/16	4
54	MAJ	2	MAJ	3	MAJ	4	1/8	3
55	MAJ	2	MAJ	4	MAJ	4	1/4	2
56	MAJ	2	MAJ	4	XOR	-	1/4	2
58,59,62 63,68,69 70,71,72	XOR	-	XOR	-	XOR	-	1/4	2

***Chú ý 8**:* Cho trước mẫu $\mathcal{M}$'' (tương ứng $\mathcal{M}$), thì W_1 và W_6 được chọn theo các ràng buộc, việc tìm va chạm còn lại không thay đổi (xem phần 1). Độ phức tạp thực hiện (running complexity) thu được là 2^{68} (tương ứng 2^{69}). Dù sao, hãy cẩn thận hơn khi thể hiện (implement) thuật toán tìm kiếm va chạm, ta có thể bỏ qua xác suất còn lại gây ra bởi xáo trộn trong vòng 14. Sau đó chúng ta thu được độ phức tạp thực hiện bằng 2^{67} (tương ứng với 2^{68}). Hơn nữa trong trường hợp của $\mathcal{M}$, ta cũng có thể chặn các xác suất gây ra bởi các xáo trộn trong vòng 16 và 17. Việc này làm giảm hơn nữa xác suất thành công của $\mathcal{M}$ xuống giá trị yêu cầu 2^{61}.

Mẹo cuối cùng này cũng có thể được dùng trong mô hình SHI2. Vì vậy, thay vì xác suất $1/2^{24}$ thu được trong Chú ý 3, ta có thể thu được xác suất $1/2^{20}$.

Bảng 4: Xác suất thành công của mặt che $\mathcal{M}$'' cho SHA-0

Xáo trộn trong vòng i	$f^{(i+2)}$	t/hợp	$f^{(i+3)}$	t/hợp	$f^{(i+4)}$	t/hợp	xác suất tổng	logarithm xác suất
3	IF	2	IF	3	IF	3	1/32	5
11	IF	2	IF	3	IF	3	1/32	5
14	IF	2	IF	3	IF	3	1/32	4+1
22,27,28 30,31,33 34,35	XOR	-	XOR	-	XOR	-	1/4	2
36	XOR	-	XOR	-	MAJ	4	1/4	2
37	XOR	-	MAJ	4	MAJ	4	1/4	2
38	XOR	-	MAJ	4	MAJ	3	1/8	3
40	MAJ	2	MAJ	3	MAJ	4	1/8	3
41	MAJ	2	MAJ	4	MAJ	3	1/8	3
43	MAJ	2	MAJ	3	MAJ	3	1/16	4
46	MAJ	2	MAJ	3	MAJ	3	1/16	4
51	MAJ	2	MAJ	3	MAJ	3	1/16	4
53	MAJ	2	MAJ	3	MAJ	3	1/16	4
55	MAJ	2	MAJ	3	MAJ	3	1/16	4
58,60,62 66,68,69 70,73,74	XOR	-	XOR	-	XOR	-	1/4	2

***Chú ý 9**:* Trong phần giữa của 20-vòng thứ hai, khối mẫu $\mathcal{M}$ với xác suất $1/2^{69}$ (tìm kiến cơ bản) hoặc $1/2^{61}$ (tìm kiếm đã cải tiến), chúng tôi đã may mắn tìm được một nhóm 5 số không (0) (thực tế là 6 nhưng 5 là đủ cho mục đích của chúng ta). Điều

này cho phép chúng ta dừng tấn công sau nhóm này, với va chạm từng phần trên 35 vòng của SHA. Ở dưới là một va chạm từng phần như vậy:

```
78fb1285  77a2dc84  4035a90b  b61f0b39
4a4d1c83  186e8429  74326988  7f220f79
a08e7920  16a3e469  2ed4213d  4a75b904
38bef788  2274a40c  4c14e934  cee12cec
```

và

```
78fb1285  77a2dc84  4035a909  b61f0b79
4a4d1c81  986e8429  7432698a  ff220f39
a08e7922  96a3e469  aed4213d  ca75b904
38bef788  2274a40c  4c14e936  cee12cac
```

kết quả thu được (của cả 2) sau 35 vòng của SHA-0:

```
7b907fb9  d050108b  88d6e6d6  5c70d4a3  7e06a692
```

Xác suất để tìm một va chạm như vậy là $1/2^{22}$ nếu sử dụng tìm kiếm va chạm cơ bản, hoặc $1/2^{14}$ nếu sử dụng tìm kiếm va chạm đã cải tiến.

3. Trường hợp SHA-1

Trong trường hợp SHA-1 các bit được đan xen với nhau và do đó ta không thể chia mở rộng thành 32 mở rộng con. Dù sao, tính bất biến (invariance) bằng việc dịch chuyển vẫn đúng. Vì vậy, ta vẫn có thể suy ra 5 *corrective mask* (mặt che điều chỉnh) từ một *perturbative mask* (mặt che xáo trộn), sử dụng cách xây dựng của phần 1.

Chính xác hơn, cho trước một *perturbative mask* M_0 là đầu ra của E_1. Biểu thức (3) và (7) vẫn xảy ra, và mặt che xây dựng được M được xác định bởi (8) lại là đầu ra của E_1.

Việc tìm *perturbative mask* M_0 có thể thực hiện bằng các công cụ lý thuyết mã (coding theory tools) [3], bởi mặt che được xem như một từ mã có trọng số thấp (low-weight codeword) của mở rộng. Thực hiện tìm kiếm như vậy trên E_1 dẫn tới một số từ mã rất ngắn so với số chiều của mã (dimension of code). Tuy nhiên, với xác suất rất cao, không có từ mã có trọng số nhỏ hơn 100 tồn tại trong E_1, mà thoả mãn các ràng buộc (xem phần 1), trong khi đó vẫn tồn tại từ mã có trọng số 27 trong E_0.

Do mỗi bit của perturbative mask M_0 đem lại ít nhất thừa số 1/4 trong xác suất thành công tổng, tấn công của chúng ta vì thế sẽ không có hiệu quả với SHA-1.

Dù sao, vẫn còn một bài toán mở là xem các *differential mask* (mặt che vi sai) có tồn tại hay không trong trường hợp của SHA-1, bởi vì tấn công của chúng tôi xây dựng các mặt che rất đặc biệt.

4. Kết luận

Chúng tôi đã phát triển một kiểu tấn công mới trên các hàm SHA, mà kết quả thu được tốt hơn tấn công nghịch lý ngày sinh cổ điển trên SHA-0. Tấn công này có liên quan tới thám mã vi sai nổi tiếng [1] trong đó nó thực hiện tìm kiếm một số loại mặt che đặc trưng (characteristic masks) mà có thể cộng với từ đầu vào thì với xác suất không tầm thường sẽ không làm thay đổi đầu ra của hàm nén. Mở rộng của SHA-1 dường như được thiết kế để chống lại kiểu tấn công này, do đó nó tăng mức độ tin cậy trong chuẩn này.

Lời cảm ơn

Chúng tôi muốn cảm ơn Matthew Robshaw và những trọng tài về những phê bình, nhận xét có giá trị và các cải tiến cho bài báo này.

Tài liệu tham khảo

1. E. Biham, và A.Shamir. Cryptanalysis of Full 16-Round DES, CRYPTO'92 LNCS 740, pp 487-496, 1993. 71
2. B. den Boer, và A. Bosselasers. Collisions for the compression function of MD5, EUROCRYPT'93 LNCS 773, pp 293-304, 1994. 56
3. A. Canteaut, và F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: Application to primitive narrow-sense BCH codes of length 511, IEEE Trans. Inform. Thoery, IT-44(1), pp 367-378, Jan. 1998. 70
4. H. Dobbertin. Cryptanalysis of MD4, Fast SoftEncryption LNCS 1039, pp 53-69, 1996. 56
5. R. Rivest. The MD4 Message-Digest Algorithm, CRYPTO'90 LNCS 537, pp303-311, 1991. 56
6. R. Rivest. The MD5 Message-Digest Algorithm, Network Working Group Request for Comments: 1321, April 1992. http://theory.lcs.mit.edu/~rivest/Rivest-MD5.txt 56
7. Secure Hash Standard. Federal Information Processing Standard Publication #180, U.S. Deparment of Commerce, National Institute of Standards and Technology, 1993. 56, 57, 71
8. Secure Hash Standard. Federal Information Processing Standard Publication #180-1, U.S. Deparment of Commerce, National Institute of Standards and Technology, 1995 (phần tiếp theo của [7]). 56

Phân tích SHA-1 trong chế độ mã hoá

Helena Handchuh, Lars R.Knudsen, và Matthew J.Robshaw
(CT-RSA 2001, LNCS 2020, pp.70-83)

1. Giới thiệu

Nhiều hàm băm phổ biến ngày nay đều dựa trên MD4 [8]. MD4 được xây dựng cho các cài đặt phần mềm nhanh trên các máy 32-bit và có đầu ra 128 bit. Vì kết quả của Dobbertin [5,4] người ta không còn sử dụng MD4 để băm (một cách an toàn) nữa, vì các va chạm được tìm thấy trong khoảng 2^{20} tính toán hàm nén. Năm 1991 MD5 đã được giới thiệu như là là một phiên bản mạnh hơn của MD4. Các dạng khác bao gồm RIPEMD-128, và RIPEMD-160. SHA được công bố như chuẩn FIPS năm 1993.

SHA được giới thiệu bởi Viện Tiêu chuẩn và Công nghệ Quốc gia Mỹ (NIST) năm 1993, với tên gọi là SHA-0. Năm 1995 có một thay đổi nhỏ với SHA-0, dẫn tới SHA-1. Chúng ta sẽ nói tới chuẩn này trong phần mô tả chi tiết của thuật toán [10].

Tấn công tốt nhất đã biết với SHA-0 khi sử dụng làm hàm băm là của Chabaud và Joux [3]. Họ chỉ ra rằng trong khoảng 2^{61} ước lượng của hàm nén, người ta có thể tìm được 2 thông điệp có cùng giá trị băm. Tấn công brute-force khai thác nghịch lý ngày sinh cũng yêu cầu khoảng 2^{80} ước lượng. Hiện chưa có tấn công nào thông báo trên SHA-1 trong các tài liệu mở. Trong bài báo này chúng tôi chỉ xét tới SHA-1.

1.1 Sử dụng SHA trong chế độ mã hoá

SHA chưa khi nào được dùng cho mã hoá. Dù sao, hàm nén có thể được dùng cho mã hoá. Mỗi bước trong 80 bước của SHA-1 (được chia thành 4 vòng, mỗi vòng 20 bước) đều có thể tính ngược được theo năm biến A, B, C, D, E được dùng để nén. Do đó, nếu ta đưa khoá bí mật vào trong message và bản rõ như là giá trị khởi đầu, ta thu được một hàm có ngược từ hàm nén đơn giản bằng việc bỏ qua phép cộng ngược cuối cùng với đầu. Đây là chế độ mã hoá của SHA được xét đến ở đây. Mã khối thu được có tên là SHACAL và đã được đề nghị với NESSIE bởi Naccache và Helena Handschuh. Do đó SHACAL-1 là mã khối 160 bit sử dụng 512 bit khoá và SHACAL-2 là mã khối 256 bit sử dụng 512 bit khoá. Các khoá ngắn hơn có thể sử dụng bằng cách thêm vào khoá các số 0 để thành xâu 512 bit. Tuy nhiên SHACAL không sử dụng khoá ngắn hơn 128 bit.

Mã khối SHACAL là một trong các ứng cử được đệ trình cho dự án NESSIE, gồm 2 phiên bản SHACAL-1 và SHACAL-2. Thuật toán này dựa trên chuẩn hàm băm

SHA được dùng trong chế độ mã hoá. Sức mạnh chính của họ mã khối này là sự thừa hưởng từ phân tích kỹ lưỡng trên chính bản thân hàm băm này. Chúng tôi khẳng định rằng không có điểm yếu tiềm ẩn nào được cài vào trong mã khối này, và chúng tôi tin rằng các nguyên tắc thiết kế là mạnh. Những gì mà chúng tôi đã biết là SHACAL chưa được đăng ký bản quyền.

SHA được NIST đưa ra năm 1993, và được gọi là SHA-0. Năm 1995 SHA-0 có sự thay đổi rất nhỏ và được gọi là SHA-1. Năm 2000, NIST đã công bố một thuật toán mới với tên gọi là SHA-2; 3 hàm băm mới được mô tả tương ứng với việc tạo bản tóm lược có độ dài 256, 384 và 512 bit. Người ta chọn ra 2 primitive trong số 5 primitive trên cho thuật toán SHACAL, đó là SHA-1 và SHA-2 với 256 bit tóm lược. Tuy rằng SHACAL được mô tả ở 2 dạng với NESSIE, nhưng nó vẫn đúng để tạo SHACAL cho các khối 384 và 512 bit.

Ở đây chúng ta chỉ quan tâm đến SHACAL-1 (SHACAL-1 được xây dựng từ SHA-1, còn SHACAL-2 được xây dựng từ SHA-256). Tính hiệu quả tính toán được đánh giá cho SHACAL-1 là 2480 cycles cho phép mã 20 byte, 2320 cycles cho phép giải mã 20 byte và 2280 cycles để thiết đặt một khoá 64 bit. Các phép đo thời gian đã được thực hiện cho máy PC dùng bộ xử lý Pentium III tốc độ 800 Mhz. Với SHACAL-1 thì 20 triệu lần mã mất 62 giây, 20 triệu lần dịch mất 58 giây và 20 triệu lần thiết đặt khoá mật 57 giây.

1.2 Mô tả SHACAL

Như vậy ta có thể mô tả thuật toán SHACAL như sau:

Các ký hiệu:

+	Phép cộng modulo 2^{32} của các từ 32 bit.
$ROTL_i(W)$	Quay từ 32 bit W sang trái i vị trí.
$S_i(W)$	Quay từ 32 bit W sang phải i vị trí.
$R_i(W)$	Dịch từ 32 bit W sang phải i vị trí.
$\oplus$	Phép toán XOR theo bit
&	Phép toán AND theo bit.
|	Phép toán OR theo bit
~	Phép toán NOT theo bit

Các hàm được định nghĩa trong hàm nén của SHA-1:

$f_{if}(X,Y,Z) = (X \& Y) \mid (\sim X \& Z)$

$f_{xor}(X,Y,Z) = (X \oplus Y \oplus Z)$

$f_{maj}(X,Y,Z) = ((X \& Y) \mid (X \& Z) \mid (Y \& Z))$

Thuật toán mã hoá

1. Đầu vào (bản rõ) gồm 160 bit được chia thành 5 khối 32bit và lần lượt gán cho 5 thanh ghi (32 bit) A, B, C, D và E. Khoá $K = [W^0 \| W^1 \| W^2 \| ... \|$

W^{15}] - gồm 15 word.

2. Thực hiện mã hoá như sau:
   - Mở rộng 512 bit khoá thành 2560 bit khoá:
     $W^i = ROTL_1(W^{i-3} \oplus W^{i-8} \oplus W^{i-14} \oplus W^{i-16})$, $16 \le i \le 79$.
   - for (i=0; i<80; i++) thực hiện
     $A^{i+1} = W^i + ROTL_5(A^i) + f^i(B^i, C^i, D^i) + E^i + K^i$.
     $B^{i+1} = A^i$.
     $C^{i+1} = ROTL_{30}(B^i)$
     $D^{i+1} = C^i$.
     $E^{i+1} = D^i$.

     Trong đó:

     $f^i = f_{if}$, với $0 \le i \le 19$
     $f^i = f_{xor}$, với $20 \le i \le 39$ và $60 \le i \le 79$
     $f^i = f_{maj}$, với $40 \le i \le 59$

     Các hằng số K^i được định nghĩa như sau:

     K^i = 0x5A827999, với $0 \le i \le 19$
     K^i = 0x6ED9EBA1, với $20 \le i \le 39$
     K^i = 0x8F1BBCDC, với $40 \le i \le 59$
     K^i = 0xCA62C1D6, với $60 \le i \le 79$.

3. Đầu ra (bản mã) là giá trị của 5 thanh ghi A, B, C, D, E sau 80 bước.

Thuật toán giải mã

1. Đầu vào (bản mã) gồm 160 bit được chia thành 5 khối 32bit và lần lượt gán cho 5 thanh ghi (32 bit) A, B, C, D và E. Khoá K = [W^0 || W^1 || W^2 || ... || W^{15}] - gồm 15 word.
2. Thực hiện giải mã như sau:
   - Mở rộng 512 bit khoá thành 2560 bit khoá:
     $W^i = ROTL_1(W^{i-3} \oplus W^{i-8} \oplus W^{i-14} \oplus W^{i-16})$, $16 \le i \le 79$.
   - for (i=80; i>0; --i) thực hiện
     $A^i = B^{i+1}$,
     $B^i = ROTL_2(C^{i+1})$,
     $C^i = D^{i+1}$,
     $D^i = E^{i+1}$,
     $E^i = A^{i+1} - W^i - ROTL_5(A^i) - f^i(B^i, C^i, D^i) - K^i$.
3. Đầu ra (bản rõ) là giá trị của 5 thanh ghi A, B, C, D, E sau 80 bước.

1.3 Tấn công SHA trong chế độ mã hoá

Hai tấn công tốt nhất đã biết trên các hệ thống tương tự với SHA trong chế độ mã hoá là *thám mã tuyến tính* [7] và *thám mã vi sai* [1]. Có rất nhiều các dạng khác nhau của 2 tấn công này được đề nghị trong các tài liệu nhưng nguyên lý cơ bản là gần giống nhau. Cũng vậy, nhiều tấn công khác với các lược đồ mã hoá đã được

gợi ý nhưng chúng đều ít thông dụng hơn 2 tấn công ở trên. Hơn nữa chúng tôi cho rằng các thuộc tính khoá yếu tiềm ẩn, các tấn công khoá liên quan (related key attacks) [2] hoặc các tấn công tương tự có thể chuyển thành các tấn công va chạm (collision attacks) một cách hiệu quả trên hàm nén; do đó chúng ta kết luận rằng không có biện pháp trực tiếp nào tấn công SHA trong chế độ mã hoá. Trong báo cáo này chúng tôi chỉ xét thám mã tuyến tính và thám mã vi sai. Các tấn công này được áp dụng với SHACAL, nhưng như chúng ta sẽ thấy, độ phức tạp tấn công dựa trên những tiếp cận này là hoàn toàn không thực tế.

SHA sử dụng xen lẫn 2 nhóm toán tử, phép cộng modular 2^{32} và XOR (cộng từng bit theo modulo 2). Nếu chúng ta sử dụng biểu diễn các từ theo nhị phân, ví dụ: $A = a_{w-1}2^{w-1} + ... + a_1 2 + a_0$, và tương tự với S, biểu diễn nhị phân của tổng $Z = A + S$ được tính bằng công thức sau:

$$z_j = a_j + s_j + \sigma_{j-1} \text{ và } \sigma_j = a_j s_j + a_j s_{j-1} + s_j \sigma_{j-1}, \quad (1)$$

mà σ_{j-1} ký hiệu bit nhớ và $\sigma_{-1} = 0$ ([9]). Công thức này sẽ được dùng trong phần tiếp theo nhiều lần.

2. Thám mã tuyến tính

Thám mã tuyến tính cố gắng xác định một dãy các xấp xỉ tuyến tính A_i với các thành phần chức năng khác nhau trong mã khối, đó là các S-box, phép cộng số nguyên, các toán tử logic hoặc bất cứ toán tử gì. Các xấp xỉ tuyến tính riêng lẻ sau đó được kết hợp lại để tạo xấp xỉ cho phần lớn hơn của thủ tục mã hoá. Tổ hợp các xấp xỉ tuyến tính chỉ đơn giản là XOR theo bit vì vậy xấp xỉ cuối cùng là $A_1 \oplus A_2 \oplus ... \oplus A_n$.

Trong phân tích tiếp theo chúng tôi sẽ chỉ xét các xấp xỉ theo từng bit riêng biệt thông qua các toán tử khác nhau. Kinh nghiệm thực tế chỉ ra rằng cố gắng sử dụng các xấp xỉ tuyến tính nặng hơn (heavier linear approximation) sẽ sớm gặp phải rắc rối. Trong khi có thể tưởng tượng được rằng với một số toán tử thì các xấp xỉ tuyến tính nặng hơn sẽ có độ lệch (bias) riêng biệt lớn hơn, nhưng thường khó sử dụng chúng hơn như một phần của tấn công và như vậy chúng thường không hữu ích. Chúng ta sẽ sử dụng ký hiệu e_i để biểu thị mặt nạ có 1 bit duy nhất được sử dụng để tạo nên xấp xỉ tuyến tính. Cho nên e_i là từ 32 bit mà có các số 0 ở tất cả các vị trí bit trừ bit thứ i. Ta sẽ qui định vị trí bit có trọng số nhỏ nhất là bit 0.

Trong tất cả các vòng có 4 phép cộng số nguyên. Tuy nhiên, hai trong số này là cộng với các hằng số; một với khoá và một là hằng số của vòng. Trước tiên ta tạm thời bỏ hai phép cộng này, nhưng thực tế giá trị của khoá có tác động quan trọng đến độ lệch của xấp xỉ.

Ngay cả khi không có xem xét này, sử dụng các xấp xỉ tuyến tính qua hai (hoặc nhiều hơn) phép cộng liên tiếp đã là vấn đề phức tạp. Như một ví dụ, chúng tôi muốn xét phép cộng với hai phép cộng số nguyên $x = (a+b) + c$. Xét phép cộng số

nguyên thứ nhất y = a + b một cách riêng biệt. Khi đó xác suất để có các xấp xỉ tuyến tính $a[i] \oplus b[i] = y[i]$ $(0 \le i \le 31)$ là 2^{-i}. Nếu sau đó ta xét phép cộng số nguyên thứ hai x = y + c có thể gợi cho chúng ta sử dụng trực tiếp Bổ đề Pilling-up, nhưng điều đó sẽ cho ta các kết quả sai lầm.

Ví dụ, trong vị trí bit i = 2, Bổ đề Pilling-up có thể phát biểu rằng xấp xỉ xảy ra với độ lệch $2^{-3} \times 2^{-3} \times 2 = 2^{-5}$. Nhưng chú ý rằng đầu ra của phép cộng số nguyên thứ nhất được sử dụng trực tiếp như là đầu vào cho phép cộng số nguyên thứ hai, do đó hai phép toán này là không độc lập nhau. Thay vào đó, nếu ta đánh giá các biểu thức logic một cách trực tiếp sử dụng 3 bit có trọng số thấp nhất của a, b và c thì ta thấy rằng độ lệch trong thực tế là 2^{-3}.

Trong trường hợp của SHA-1 chúng ta còn có tình huống phức tạp hơn nữa. Chúng ta có một dãy phép cộng cần phải xấp xỉ x = (a + b) + k + c, mà k là hằng số phụ thuộc vòng (hoặc khoá). Xấp xỉ mà chúng ta dự tính sử dụng là $x[i] = a[i] + b[i] + k[i] + c[i]$ $(0 \le i \le 31)$. Độ lệch mà ta quan sát được sẽ phụ thuộc vào giá trị của k.

Chúng ta hãy xét một trường hợp đơn giản hơn, x = k + y. Hình dung rằng chúng ta tạo xấp xỉ $x[i] = k[i] + y[i]$ $(0 \le i \le 31)$, mà y[i] là bít phụ thuộc bản rõ và k[i] là bit khoá cố định. Rõ ràng nếu chúng chỉ xét bit có trọng số thấp, i = 0, thì xấp xỉ luôn luôn xảy ra. Với bit i = 1, xấp xỉ luôn xảy ra nếu k[0] = 0, nhưng chỉ với xác suất 0.5, nghĩa là độ lệch bằng 0, nếu k[0] = 1. Nếu ta sử dụng bit $i \ge 1$ cho xấp xỉ thì các số nguyên k thoả mãn $(k \,\&\, (2^i - 1)) = 0$ cho độ lệch lớn nhất, vì khi đó sẽ không có các bit nhớ trong các vị trí bit thấp hơn i, và xấp xỉ luôn luôn xảy ra, xem công thức (1). Chú ý rằng số các khoá "yếu hơn" này cho ta độ lệch lớn nhất là phụ thuộc vào vị trí bit i. Khi i = 2 ta có một trong bốn khoá đó cho độ lệch lớn nhất. Nếu i = 30 thì ta chỉ có một trong 2^{30} khoá cho độ lệch lớn nhất này. Chúng ta cũng chú ý rằng một số giá trị của k cho ta độ lệch bằng 0. Chính là các giá trị của k thoả mãn $(k \,\&\, (2^i - 1)) = 2^{i-1}$. Với các giá trị như vậy không có các bit nhớ cho các vị trí nhỏ hơn i-1. Nhưng vì k[i-1] = 1 trong trường hợp này, nên sẽ có một bit nhớ trong vị trí i nếu và chỉ nếu y[i-1] = 1. Nếu y biến đổi qua tất cả các giá trị (cách tiếp cận thường thấy trong thám mã tuyến tính) thì xấp xỉ $x[i] = k[i] + y[i]$ xảy ra với xác suất 0.5, do đó độ lệch bằng 0.

2.1 Tất cả các vòng

Cấu trúc vòng của SHA-1 có nghĩa rằng trong tất cả bốn vòng chúng ta có thể dễ dàng xác định họ các xấp xỉ tuyến tính mà luôn đúng với cả bốn bước. Chúng ta sử dụng Γ để ký hiệu một mẫu bit tổng quát được dùng trong xấp xỉ và x^c để ký hiệu phép quay trái của từ 32-bit x đi c vị trí.

A	B	C	D	E	độ lệch
Γ	-	-	-	-	
		↓			1/2
-	Γ	-	-	-	
		↓			½
-	-	Γ^{30}	-	-	
		↓			½
-	-	-	Γ^{30}	-	
		↓			½
-	-	-	-	Γ^{30}	

Đây là một xấp xỉ tuyến tính "hoàn thiện" (perfect) thông qua 4 bước bất kỳ của SHA-1. Mở rộng xấp xỉ này chúng ta sẽ cần tính tới ảnh hưởng của các hàm logic khác nhau được dùng trong các vòng khác nhau.

2.2 Các vòng 2 và 4

Trong các vòng này hàm logic f_{xor} được dùng để tổ hợp các từ là phép XOR theo bit $b \oplus c \oplus d$. Hàm này đưa ra một số khó khăn cho các nhà thám mã khi cố gắng điều khiển số bit được dùng trong các xấp xỉ.

Trong vòng 2 và 4 ta có thể mở rộng xấp xỉ tuyến tính "hoàn thiện" cơ sở mà ta đã chỉ ra cho tất cả các vòng theo cách sau. Điều này cho ta một xấp xỉ tuyến tính có tác động qua bảy bước và xảy ra với xác suất 1 (tức là độ lệch là 1/2). Để tiếp tục ta đặt $\Gamma = e_0$.

	A	B	C	D	E	độ lệch
i	e_2	-	-	-	-	
			↓			1/2
i+1	-	e_2	-	-	-	
			↓			½
i+2	-	-	e_0	-	-	
			↓			½
i+3	-	-	-	e_0	-	
			↓			½
i+4	-	-	-	-	e_0	
			↓			½
i+5	e_0	e_{27}	e_{30}	e_0	e_0	
			↓			½
i+6	e_0	$e_{27} \oplus e_0$	$e_{30} \oplus e_{25}$	$e_{30} \oplus e_0$	-	
			↓			½
i+7	-	e_0	$e_{25} \oplus e_{30}$	$e_{25} \oplus e_{30}$	$e_{30} \oplus e_0$	

Ta giả thiết rằng đây là xấp xỉ tuyến tính "hoàn thiện" dài nhất thông qua các bước trong vòng 2 và 4. Nếu ta sử dụng xấp xỉ này để tấn công thì ta sẽ cần mở rộng nó. Nếu ta xét chỉ phép mở rộng có thể ở phía trên thì ta có xấp xỉ tuyến tính một - bước (one-step) sau:

A	B	C	D	E
e_{29}	e_2	e_2	e_2	e_2
		↓		
e_2	-	-	-	-

Ở phía chân của xấp xỉ tuyến tính bảy-bước (seven-step) ta cần sử dụng xấp xỉ một-bước sau:

A	B	C	D	E
-	e_0	$e_{25} \oplus e_{30}$	$e_{25} \oplus e_{30}$	$e_{30} \oplus e_0$
		↓		
$e_{30} \oplus e_0$	$e_{27} \oplus e_{25}$	e_{28}	$e_{25} \oplus e_0$	$e_{25} \oplus e_0$

Sử dụng các kỹ thuật đã đề cập trong phần dẫn nhập, chúng ta ước lượng rằng độ lệch lớn nhất cho xấp xỉ tuyến tính chín-bước (có tính đến giá trị có thể tốt nhất cho dữ liệu khoá) là nhỏ hơn $2^{-2} \times 2^{-2} \times 2 = 2^{-3}$ và nhiều hơn $2^{-3} \times 2^{-3} \times 2 = 2^{-5}$. Độ lệch này có thể áp dụng cho một trong 2^{32} khoá vì ta cần một điều kiện khoá trên xấp xỉ trong bước 1 và một điều kiện khoá trên xấp xỉ trong bước 9. Với xấp xỉ một trong 2^2 khoá sẽ không có độ lệch cho xấp xỉ tuyến tính này. Giá trị trung bình của độ lệch ta cần phải nằm giữa $2^{-3} \times 2^{-3} \times 2 = 2^{-5}$ và $2^{-4} \times 2^{-4} \times 2 = 2^{-7}$. Các thử nghiệm cho thấy độ lệch đó sử dụng các điều kiện khoá tốt nhất là khoảng $2^{-4.0}$ và độ lệch trung bình đó trên tập tất cả các khoá là $2^{-5.6}$. Với 1 trong 4 khoá không có độ lệch trong xấp xỉ.

Chúng ta đã xác định xấp xỉ chín bước. Để làm thuận tiện hơn phân tích tổng quát của ta, ta sẽ thêm 1 bước vào xấp xỉ 9 bước này. Ta có thể thêm 1 bước ở đầu hoặc ở cuối. Với các nhà thám mã, việc thêm xấp xỉ 1 bước sau vào phần đầu của xấp xỉ đã có dường như dễ dàng hơn.

A	B	C	D	E
$e_{24} \oplus e_2$	$e_{29} \oplus e_4$	$e_{29} \oplus e_2$	$e_{29} \oplus e_2$	e_{29}
		↓		
e_{29}	e_2	e_2	e_2	e_2

Tiếp theo các phương pháp trên, chúng tôi sẽ ước lượng rằng độ lệch lớn nhất đó (dưới những điều kiện khoá thuận lợi nhất cho người phân tích) nằm trong khoảng

(2^{-4},2^{-7}) và độ lệch trung bình đó nằm trong khoảng (2^{-7}, 2^{-10}). Với một chút trội hơn 1 trong 4 khoá sẽ không có độ lệch. Các thử nghiệm chứng minh rằng các giá trị khoá tốt nhất (mà có thể xuất hiện với 1 trong $2^{29+30+2}$ khoá ngẫu nhiên) cho độ lệch của $2^{-5.4}$ nhưng độ lệch cho khoá trung bình đó được thực hiện tốt hơn một chút mong muốn với độ lệch $2^{-6.7}$. Vì trường hợp của các giá trị khoá tốt nhất là rất hiếm, chúng tôi đề nghị sử dụng 2^{-6} như một biểu diễn thận trọng cho độ lệch của xấp xỉ tuyến tính bước mười này trong các vòng 2 và 4.

2.3 Vòng 1

Như phân tích của chúng tôi trong vòng 2 và 4, ta xét mở rộng tốt nhất cho xấp xỉ "hoàn thiện" 4 bước cơ sở mà áp dụng trong tất cả các vòng. Hàm logic ở đây f_{if} là bc ⊕ (1 ⊕ b)d. Không có các xấp xỉ hoàn thiện nào thông qua phép toán này, mặc dù có một vài xấp xỉ với độ lệch 2^{-2}.
Ta có thể thấy mở rộng 4-bước sau cho xấp xỉ tuyến tính cơ sở đã có:

A	B	C	D	E	
-	-	-	-	e_0	
		↓			¼
e_0	e_{27}	-	e_0	-	
		↓			½
-	e_0	e_{25}	-	e_0	
		↓			¼
e_0	e_{27}	-	$e_{25}⊕e_0$	-	
		↓			½
-	e_0	e_{25}	-	$e_{25}⊕e_0$	

Độ lệch cho mở rộng này có thể được tính là 2^{-3}. Trong mở rộng tiếp theo ta cần xấp xỉ qua phép cộng tại một vị trí bit khác với bit có trọng số nhỏ nhất. Ta sẽ coi như độ lệch của xấp xỉ này có thể khoảng 2^{-2}.

Mở rộng 2-bước sau cho phép ta thực hiện xấp xỉ 10-bước cho các bước trong vòng 1 mà xảy ra với độ lệch không lớn hơn 2^{-6} trong trường hợp tốt nhất và trung bình trong khoảng (2^{-7}, 2^{-8}).

A	B	C	D	E
-	e_0	e_{25}	-	$e_{25}⊕e_0$
		↓		
$e_{25}⊕e_0$	$e_{27}⊕e_{20}$	-	e_0	-
		↓		
-	$e_{25}⊕e_0$	$e_{25}⊕e_{18}$	-	e_0

Thực nghiệm đã khẳng định xấp xỉ tuyến tính 10-bước. Độ lệch trung bình là $2^{-7.2}$ và với các điều kiện khoá tốt nhất (xảy ra cho 1 trong 2^{25} khoá ngẫu nhiên) độ lệch qua 20 lần thử là $2^{-6.4}$.

Chúng ta sẽ sử dụng 2^{-6} như ước lượng cho độ lệch của xấp xỉ tuyến tính 10-bước này cho các bước trong vòng 1.

2.4 Vòng 3

Một lần nữa chúng ta xét các mở rộng cho xấp xỉ tuyến tính cơ sở áp dụng cho tất cả các vòng. Hàm logic ở đây f_{maj} là bc ⊕ cb ⊕ bd. Không có xấp xỉ hoàn thiện nào với hàm này, mặc dù có một vài xấp xỉ với độ lệch 2^{-2}.

Ngay lập tức ta có thể thấy mở rộng 4-bước sau cho xấp xỉ tuyến tính cơ sở đang có:

A	B	C	D	E	
-	-	-	-	e_0	
		↓			¼
e_0	e_{27}	-	e_0	-	
		↓			½
-	e_0	e_{25}	-	e_0	
		↓			¼
e_0	e_{27}	-	e_{25}	-	
		↓			½
-	e_0	e_{25}	-	e_{25}	

Độ lệch cho mở rộng này có thể được tính là 2^{-3}. Để mở rộng hơn nữa ta cần xấp xỉ phép cộng tại vị trí bit khác với bit có trọng số nhỏ nhất. Ta sẽ coi như độ lệch của xấp xỉ này khoảng 2^{-2} cho phép cộng số nguyên đặc biệt này.

Mở rộng 2-bước sau cho phép ta thực hiện xấp xỉ 10-bước cho các bước trong vòng 1 mà xảy ra với độ lệch không lớn hơn 2^{-5} trong trường hợp tốt nhất (cho nhà phân tích) và trung bình trong khoảng $(2^{-6}, 2^{-7})$:

A	B	C	D	E
-	e_0	e_{25}	-	e_{25}
		↓		
e_{25}	e_{20}	e_{30}	-	-
		↓		
-	e_{25}	e_{18}	e_{30}	-

Các thực nghiệm xác nhận xấp xỉ tuyến tính 10-bước này và với các điều kiện khoá tốt nhất (xảy ra với 1 trong 2^{25} khoá ngẫu nhiên) độ lệch là $2^{-5.6}$ và với trường hợp trung bình độ lệch trung bình là $2^{-6.4}$.

Ta sẽ sử dụng 2^{-5} là ước lượng cho độ lệch của xấp xỉ tuyến tính 10-bước với các bước trong vòng 3.

2.5 Liên kết giữa các vòng

Xấp xỉ tuyến tinh 10-bước mà chúng ta đã xác định trong vòng 2 và 4 là đúng với 40 bước của SHA-1 đầy đủ. Do đó chúng ta ước lượng rằng khi sử dụng xấp xỉ này độ lệch cao nhất là $(2^{-6})^4 \times 2^3 = 2^{-21}$. Đây tất nhiên là một ước lượng thận trọng cao. Trong số nhiều giả thiết có lợi cho nhà phân tích mã có giả thiết là xấp xỉ tuyến tính 10-bước này có thể ghép nối với chính chúng. Nhưng nó không thể. Việc mở rộng xấp xỉ này ở cả 2 phía dường như cung cấp độ suy giảm khắt khe trong độ lệch của xấp xỉ tuyến tính có thể khai thác được.

Với vòng 1 chúng ta có thể ước lượng thận trọng rằng 20 bước có thể được xấp xỉ bằng một xấp xỉ tuyến tính với độ lệch không quá $(2^{-6})^2 \times 2 = 2^{-11}$. Cũng như vậy ta có thể ước lượng rằng 20 bước trong vòng 3 có thể được xấp xỉ bằng một xấp xỉ với độ lệch không quá $(2^{-5})^2 \times 2 = 2^{-9}$.

Với các điều kiện có lợi nhất cho nhà thám mã (các điều kiện mà ta tin là không thể được thoả mãn thực) nếu SHA-1 được xấp xỉ sử dụng một xấp xỉ tuyến tính thì độ lệch sẽ không vượt quá $2^{-21} \times 2^{-11} \times 2^{-9} \times 2^2 = 2^{-39}$. Chú ý rằng các điều kiện khoá cần thiết cho độ lệch tốt nhất với các xấp xỉ trong vòng 1 và 3 rất hiếm khi xảy ra và vì vậy chúng ta bỏ qua trường hợp này và suy ra rằng độ lệch đại đa số chắc như dưới 2^{-40}. Mặt khác, chú ý rằng xấp xỉ đã đưa ra đó có độ lệch 0 với nhiều khoá và vì vậy các xấp xỉ khác có thể được nhà phân tích dùng trong những trường hợp này đem lại độ lệch làm việc suy giảm (reduced working bias).

Do đó tấn công thám mã tuyến tính trên SHA-1 sử dụng ít hơn 2^{80} bản rõ đã biết là không thể thực hiện được.

3. Thám mã vi sai

Điều gì làm thám mã vi sai khó thực hiện trên SHA-1? Thứ nhất, việc sử dụng cả phép cộng modulo và phép hoặc-loại trừ (XOR), và thứ hai là các hàm phi tuyến f_{if}, f_{xor}, f_{maj}.

Trước tiên chúng ta hãy xét quan hệ giữa các vi sai của phép XOR và phép cộng số nguyên. Phép cộng số nguyên của một từ hằng số S với các từ 32-bit A và B, A và B chỉ khác nhau ở một vài bit, không nhất thiết dẫn tới việc tăng số bit khác nhau trong các tổng A + S và B + S. Điều này có thể được chỉ ra bởi trường hợp đặc biệt sau: Giả sử các từ A và B chỉ khác nhau ở bit thứ i, i<31. Khi đó A + S và B + S cũng chỉ khác nhau ở bit thứ i, điều này xảy ra với xác suất 1/2. Sử dụng công thức (1) ta thấy rằng A + S và B + S khác nhau ở chính xác 2 bit (liền nhau) với xác suất 1/4. Có một trường hợp quan trọng và đặc biệt cần xét, là khi A và B chỉ khác nhau ở bit có trọng số cao nhất, vị trí 31. Trong trường hợp đó A + S và B + S cũng chỉ khác nhau trong bit có trọng số cao nhất.

Các hàm f_{if}, f_{xor}, f_{maj} đều thực hiện theo từng bit. Do đó, ta có thể dễ dàng tìm ra xem các vi sai tại các đầu ra của mỗi hàm phụ thuộc như thế nào vào các vi sai của 3 đầu vào. Đó chính là, ta có thể xét 3 đầu vào (mỗi đầu vào 1 bit) và đầu ra ío một bit. Bảng 1 chỉ ra điều này cho cả 3 hàm. Ký hiệu của bảng như sau. Ba cột đầu tiên biểu diễn 8 vi sai có thể cho các đầu vào 1 bit x, y, z. Ba cột tiếp theo chỉ ra các vi sai ở đầu ra của mỗi hàm. '0' ký hiệu rằng vi sai luôn là 0, '1' ký hiệu vi sai luôn là 1 và '0/1' ký hiệu rằng một nửa số trường hợp vi sai sẽ bằng 0 và nửa số trường hợp còn lại vi sai sẽ bằng 1. Chú ý rằng hàm f_{xor} là tuyến tính trong các đầu vào, tức là vi sai tại các đầu ra có thể được xác định từ các vi sai tại các đầu vào. Tuy nhiên, như chúng ta sẽ thấy, f_{xor} giúp cho phức tạp thám mã vi sai với SHA.

Bảng 1: Phân phối của các vi sai exor thông qua các hàm f.

x	y	z	f_{xor}	f_{if}	f_{maj}
0	0	0	0	0	0
0	0	1	1	0/1	0/1
0	1	0	1	0/1	0/1
0	1	1	0	1	0/1
1	0	0	1	0/1	0/1
1	0	1	0	0/1	0/1
1	1	0	0	0/1	0/1
1	1	1	1	0/1	1

Trong phần tiếp theo chúng ta xét một số đặc trưng cho tất cả các vòng và với mỗi vòng.

3.1 Tất cả các vòng

Bảng 2. Đặc trưng 5 bước

A	B	C	D	E	prob
e_{26}	0	0	0	e_{31}	
		↓			1
0	e_{26}	0	0	0	
		↓			1
?	0	e_{24}	0	0	
		↓			1
?	?	0	e_{24}	0	
		↓			1
?	?	?	0	e_{24}	
		↓			1
?	?	?	?	0	

Đặc trưng của hình 2 xảy ra với xác suất 1 qua 5 bước (bất kỳ) của 1 trong 4 vòng bất kỳ. Dấu chấm hỏi (?) chỉ ra một giá trị không biết. Do đó, một cặp bản rõ chỉ khác nhau tại các từ đầu tiên ở vị trí bit 26 và ở từ thứ 5 tại vị trí bit 31, đem đến bản rõ sau 5 bước là bằng nhau tại các từ thứ năm. Vi sai ở các từ khác của bản rõ sẽ phụ thuộc vào vòng đặc biệt được xét và bản rõ tham gia.

3.2 Vòng 1 và 3

Đầu tiên ta xét đặc trưng 5 bước của phần trước. Với các hàm f_{if} và f_{maj} nó cho đặc trưng qua 5 bước như sau:

A	B	C	D	E	prob
e_{26}	0	0	0	e_{31}	
		↓			1
0	e_{26}	0	0	0	
		↓			½
0	0	e_{24}	0	0	
		↓			½
0	0	0	e_{24}	0	
		↓			½
0	0	0	0	e_{24}	
		↓			½
e_{24}	0	0	0	0	

Đặc trưng này có thể được ghép với đặc trưng 3 bước ở phần đầu và đặc trưng 2 bước ở phần cuối, thu được đặc trưng 10 bước sau.

A	B	C	D	E	prob
0	e_1	e_{26}	0	0	
		↓			¼
0	0	e_{31}	e_{26}	0	
		↓			¼
0	0	0	e_{31}	e_{26}	
		↓			¼
e_{26}	0	0	0	e_{31}	
		↓			1
0	e_{26}	0	0	0	
		↓			½
0	0	e_{24}	0	0	
		↓			½
0	0	0	e_{24}	0	
		↓			½
0	0	0	0	e_{24}	
		↓			½
e_{24}	0	0	0	0	

		↓			½
e_{29}	e_{24}	0	0	0	
		↓			¼
e_2	e_{29}	e_{22}	0	0	

Đặc trưng 10 bước này có xác suất 2^{-13}. Như đã chỉ rõ, việc mở rộng đặc trưng này cho nhiều bước hơn, 20 bước chẳng hạn, sẽ bao gồm các bước với trọng số Hamming trong các vi sai của 5 từ lớn hơn so với 10 bước đầu tiên ở trên.

Ta phỏng đoán rằng đặc trưng ở trên là một trong các đặc trưng với xác suất cao nhất qua 10 bước, và một đặc trưng bất kỳ nào như vậy qua 20 bước của vòng 1 và vòng 3 sẽ có xác suất nhỏ hơn 2^{-26}.

3.3 Vòng 2 và 4

Đối với thám mã vi sai, hàm f_{xor} được dùng trong vòng 2 và 4 được coi là sự khác biệt quan trọng so với các hàm được dùng trong vòng 1 và 3. Chú ý đầu tiên là nếu ta thay tất cả phép cộng modulo bằng phép XOR, các bước trong vòng 2 và 4 là tuyến tính đối với vi sai XOR, nói cách khác, cho trước vi sai đầu vào người ta có thể với xác suất 1 xác định được vi sai ở đầu ra sau một số bước bất kỳ, tối đa là 20. Như đã chỉ ra ở trên, việc sử dụng lẫn phép cộng modulo và XOR chỉ có một ít ảnh hưởng cho các cặp bản rõ với vi sai có trọng số Hamming thấp. Do đó đặc trưng tốt cho các bước này cần có trọng số Hamming thấp chạy qua càng nhiều bước càng tốt. Trước hết, xét đặc trưng 5 bước của bảng 2. 4 bước đầu tiên được suy ra như trong bảng 3:

Bảng 3.

A	B	C	D	E	prob
e_{26}	0	0	0	e_{31}	
		↓			1
0	e_{26}	0	0	0	
		↓			½
e_{26}	0	e_{24}	0	0	
		↓			½
$e_{24,31}$	e_{26}	0	e_{24}	0	
		↓			1/16
$e_{4,24,26,29}$	$e_{24,31}$	e_{24}	0	e_{24}	

Ở đây chúng ta đã sử dụng ký hiệu $e_{a_1,\ldots,a_r}$ cho $e_{a_1} \oplus e_{a_2} \oplus \ldots \oplus e_{ar}$. Có thể thấy rằng cho đặc trưng này thì trọng số Hamming của vi sai ở các từ bản mã sẽ tăng lên cho các bước sau. Khác đi, hãy xét đặc trưng được chỉ trong bảng 4.

Đặc trưng này được tìm bởi máy tính. Trong tất cả các vi sai đầu vào có thể cho đến vi sai 1 bit tại mỗi một trong số 5 từ đầu vào, tổng cộng $33^5 - 1$ đặc trưng, 9 bước cuối cùng của đặc trưng trên có trọng số Hamming thấp nhất trong các vi sai bản mã của tất cả các bước. Với tìm kiếm này chúng tôi đã thay phép cộng modulo bằng XOR. Đặc trưng 9 bước có thể được ghép nối với đặc trưng một bước ở phần đầu, như đã chỉ ra ở trên. Trong SHA thực xác suất của 10 bước này xấp xỉ 2^{-26}, mà chúng tôi đã dùng trong ước lượng ở trên cho cách chạy của vi sai XOR sau phép cộng modulo. Điều này có thể **không** đưa ra cận cho đặc trưng tốt nhất qua 10 bước của SHA, nhưng một tìm kiếm đầy đủ dường như không thể hiện (implement) được, hơn nữa có đủ bằng chứng để kết luận rằng không có đặc trưng xác suất cao qua 20 bước của vòng 2 và 4. Chúng tôi ước lượng rằng đặc trưng tốt nhất như vậy sẽ có xác suất nhỏ hơn 2^{-32}.

3.4 Liên kết giữa các vòng

Sử dụng các ước lượng cho các đặc trưng tốt nhất cho các vòng 1, 2, 3 và 4 của phần trước, ta được một ước lượng của đặc trưng tốt nhất cho toàn bộ 80 bước của SHA, là $2^{-26} * 2^{-32} * 2^{-26} * 2^{-32} = 2^{-116}$. Chúng tôi nhấn mạnh rằng ước lượng này là được rút ra rất thận trọng. Trước hết, các ước lượng cho mỗi vòng đều rất thận trọng, và thứ hai là không có sự bảo đảm là các đặc trưng có xác suất cao cho mỗi vòng lại có thể ghép nối lại cho toàn bộ mã pháp. Do đó chúng tôi kết luận rằng thám mã vi sai SHA dường như yêu cầu một khối lượng bản rõ chọn lọc là không thực.

Bảng 4

A	B	C	D	E	prob
e_1	e_3	e_1	e_{11}	$e_{1,3,11}$	
		↓			1/16
e_6	e_1	e_1	e_1	e_{11}	
		↓			¼
e_1	e_6	e_{31}	e_1	e_1	
		↓			¼
e_{31}	e_1	e_4	e_{31}	e_1	
		↓			¼
e_{31}	e_{31}	e_{31}	e_4	e_{31}	
		↓			½
e_{31}	e_{31}	e_{29}	e_{31}	e_4	
		↓			¼
e_{29}	e_{31}	e_{29}	e_{29}	e_{31}	
		↓			¼
e_2	e_{29}	e_{29}	e_{29}	e_{29}	
		↓			¼
e_7	e_2	e_{27}	e_{29}	e_{29}	
		↓			1/16

$$
\begin{array}{ccccc}
e_{2,12,27} & e_7 & e_0 & e_{27} & e_{29} \\
 & & \downarrow & & \qquad 1/32 \\
e_{17,27,29} & e_{2,12,27} & e_5 & e_0 & e_{27}
\end{array}
$$

4. Kết luận

Trong phần trước chúng tôi đã suy ra rằng tấn công thám mã tuyến tính trên SHA-1 như một hàm mã hoá yêu cầu ít nhất 2^{80} bản rõ đã biết và tấn công vi sai yêu cầu ít nhất 2^{116} bản rõ chọn lọc. Chú ý rằng chúng ta đang xét các xấp xỉ tuyến tính và các đặc trưng vi sai có thể xây dựng được một rõ ràng. Cũng rất tốt nếu có các xấp xỉ và đặc trưng trên SHA-1 mà chưa được phát hiện bởi kiểu phân tích này. Thay vào đó, người ta cũng có thể tìm kiếm bằng phương pháp brute-force. Vì chưa có một đường tắt nào được biết đối với tìm kiếm này, nên xác suất này cần phải được xem như không có thể xảy ra cũng như không có giá trị thực hành.

Các kỹ thuật của chúng tôi trong việc kiến thiết các xấp xỉ và các đặc trưng là không theo thể thức, nhưng dựa trên các kinh nghiệm thực hành. Chúng tôi đã rất cẩn thận trong các đánh giá và cảm thấy rất tin tưởng khẳng định rằng tấn công thám mã vi sai hoặc tuyến tính sử dụng ít hơn 2^{80} khối bản rõ là không thể thực hiện được. Chúng tôi cũng chú ý rằng ở điểm này dù sao mã khối 160-bit bắt đầu lộ thông tin bản rõ khi được dùng để mã hoá nhiều văn bản với cùng một khoá.

Cuối cùng chúng tôi đề cập rằng các xem xét thám mã bổ sung chẳng hạn các dạng thám mã tuyến tính, xấp xỉ tuyến tính bội (multiple linear approximations), và các kiểu thám vi sai khác nhau là không thể có sự khác biệt đáng kể với các ước lượng và phân tích của chúng tôi. Do đó không có sự khác biệt về thực hành với kết luận mà chúng tôi đưa ra.

Tài liệu tham khảo

1. E. Biham, A.Shamir. Differential Cryptanalysis of the Data Encryption Standard, Spinger-Verlag, 1993.
2. E. Biham, New types of cryptanalysis attacks using ralated keys. In Advances in Cryptology: EUROCRYPT'93, LNCS 765, trang 398-409. Springer-Verlag, 1994.
3. F. Chaubaud và A. Joux. Differential collisions in SHA-0. In H. Krawczyk, editor, Advances in Cryptology: CRYPTO'98, LNCS 1452, trang 56-71. Springer Verlag, 1999.
4. H. Dobbertin. Cryptanalysis of MD5 compress. Presented at the rump session of EUROCRYPT'96, May 1996.
5. H. Dobbertin. Cryptanalysis of MD4. Journal of Cryptology, vol. 11, n. 4, trang 253-271, Spinger-Verlag, 1998.

6. A. J. Menezes, P. C. van Oorschot, và S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
7. M. Matsui, Linear Cryptanalysis method for DES cipher. Trong *Advances in Cryptology* – EUROCRYPT'93, LNCS 765, trang 386-397. Springer-Verlag, 1993.
8. R.L. Rivest. The MD4 message digest algorithm. Advances in Cryptology – CRYPTO'90, LNCS 537, trang 303-311. Springer Verlag, 1991.
9. R.A. Rueppel. Analysis and Design of Stream Ciphers. Springer Verlag, 1986.
10. US Department of Commerce, N.I.S.T. Secure Hash Algorithm. n FIPS 180-1, 1995.

Các hàm băm dựa trên mã khối: phương pháp thiết kế

Bart Preneel, René Govaerts, Joos Vandewalle
CRYPTO'93, pp. 368-378

Tóm tắt: Việc thiết kế các hàm băm dựa trên mã khối được nghiên cứu khi độ dài của giá trị băm bằng với độ dài khối của mã khối và khi độ dài khoá xấp xỉ bằng độ dài khối. Mô hình tổng quát được trình bày, và chỉ ra rằng mô hình này bao gồm 9 lược đồ đã xuất hiện trước đây trong tài liệu in. Trong mô hình tổng quát đó, tồn tại 64 lược đồ có thể, và chỉ ra rằng 12 trong chúng là an toàn, chúng có thể đưa về 2 lớp dựa trên các biến đổi tuyến tính của các biến. Tính chất của 12 lược đồ này tương ứng với tính yếu của mã khối nằm trong đã được nghiên cứu. Cũng phương pháp này có thể mở rộng để nghiên cứu các hàm băm có khoá (MAC's) dựa trên mã khối và các hàm băm dựa trên phép tính modulo. Cuối cùng, một tấn công mới được trình bày đối với lược đồ được đề nghị bởi R. Merkle.

1. Mở đầu

Các hàm băm là các hàm nén đầu vào có độ dài bất kỳ thành một xâu có độ dài cố định. Nó là khối kiến trúc cơ bản cho các ứng dụng mật mã giống như bảo vệ tính toàn vẹn dựa trên "dấu vân tay" hay lược đồ chữ ký số. Các yêu cầu mật mã đã được đặt lên hàm băm là [4, 5, 19, 27, 28]:

tính một chiều: theo nghĩa này, khi cho X và h(X), sẽ "khó" tìm được một tạo ảnh thứ hai, tức là bản tin X'≠ X sao cho h(X') = h(X),

chống va chạm: "khó" tìm được va chạm, tức là hai biến khác nhau sao cho có cùng giá trị băm.

Lý do chính để kiến thiết hàm băm dựa trên mã khối là cực tiểu hoá việc thiết kế và nỗ lực cài đặt. Thiết kế một cấu trúc an toàn là bài toán khó; điều này được minh hoạ bằng một số lớn các lược đồ đã bị phá vỡ [23, 27].

Kiến thiết đầu tiên cho các hàm băm dựa trên mã khối là các hàm băm nhằm sử dụng DES [10]. Trong trường hợp này, kích thước của giá trị băm (64 bit) bằng với độ dài khối của mã khối và kích thước của khoá (56 bit) là xấp xỉ bằng độ dài khối. Các hàm băm dạng này sẽ được nghiên cứu trong tóm tắt mở rộng này. Các kiến thiết về sau cho các hàm băm chống va chạm đã được phát triển dựa trên DES; trong trường hợp này đòi hỏi rằng kích thước của giá trị băm ít nhất là 112..128 bit (vì tấn công ngày sinh [33]). Các ví dụ về các lược đồ không bị phá vỡ có thể tìm thấy trong [20, 22]. Công trình gần đây xem xét việc kiến thiết các hàm băm nếu độ dài khoá gấp đôi độ dài khối [17], và nếu khoá được coi là không đổi [26]. Các kiến thiết ban đầu vẫn còn là thú vị bởi vì đối với một số ứng dụng, một hàm băm một chiều là đủ. Hơn nữa, chúng có thể dẫn đến một hàm băm chống lại va chạm nếu mã khối cùng với độ dài khoá đủ lớn là có thể.

2. Mô hình tổng quát

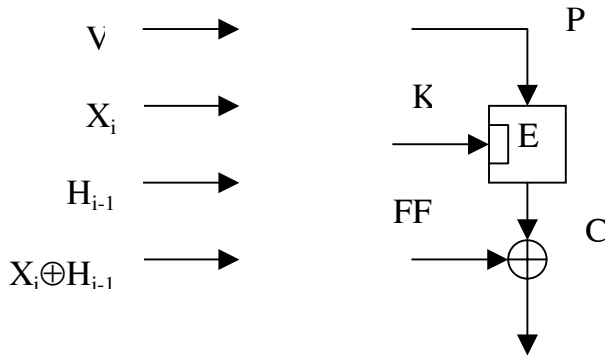
Phép mã bản rõ X bằng khoá K được ký hiệu là E(K,X). Phép giải mã tương ứng áp cho bản mã C được ký hiệu là D(K,C). Trừ khi khẳng định khác đi, giả thiết rằng mã khối không có điểm yếu. Độ dài khối , tức là kích thước của bản rõ và mã theo bit được ký hiệu bằng n và kích thước khoá theo bit được ký hiệu bằng k. Đối tượng của hàm băm lặp được chia thành t khối từ X_1 đến X_t. Nếu độ dài tổng thể không là bội của n, đối tượng cần phải được kéo dài theo một qui tắc xác định. Hàm băm h có thể mô tả theo đó như sau:

$$H_i = f(X_i, H_{i-1}), i=1, 2,..., t.$$

ở đây f là hàm vòng, H_0 bằng với giá trị khởi đầu (IV), nó được chỉ ra cùng với lược đồ, H_t là giá trị băm. Tỷ lệ R của hàm băm dựa trên mã khối được định nghĩa như là số các phép mã cần xử lý một khối n bit.

Mô hình tổng quát cho hàm vòng của hàm băm sẽ được nghiên cứu trong tóm tắt mở rộng này được mô tả ở hình 1. Để cho đơn giản, ta giả thiết rằng k = n. Mã khối có 2 đầu vào, chính là khoá K và bản rõ P, một đầu ra C. Người ta có thể chọn làm các đầu vào một trong 4 giá trị: X_i, H_{i-1}, $X_i \oplus H_{i-1}$ và hằng số V. Cũng có thể thay đổi đầu ra C bằng phép feedforward bằng phép cộng theo modulo 2 với một trong 4 khả năng có thể. Điều này dẫn đến 4^3= 64 lược đồ khác nhau. Sau đây, chúng tôi xin giả thiết rằng w.l.o.g rằng V bằng 0 (???) .

Phép tính EXOR được chọn vì nó được sử dụng trong các đề xuất được tổng quát hoá ở đây; có thể chỉ ra rằng nó có thể được thay bởi một phép toán bất kỳ là phép hoán vị dễ tính ngược (easy-to-invert permutation) của một trong các đầu vào khi đầu vào thứ hai được cố định. Các hạn chế chính của mô hình này là chỉ có 1 phép DES được sử dụng trong một vòng và bộ nhớ trong của hàm băm cũng chỉ giới hạn đến một khối n-bit.



Hình 1. Cấu hình khi độ dài giá trị băm bằng độ dài khối. P, K và FF có thể chọn từ tập $\{V, X_i, H_{i-1}, X_i \oplus H_{i-1}\}$

3. Phân loại tấn công

5 Tấn công đối với hàm vòng $f(X_i, H_{i-1})$ có thể nhận thấy:

Direcrt Attack (D): cho H_{i-1} và h_i, dễ tìm được X_i. Tất cả các lược đồ mà bị tổn thương bởi tấn công trực tiếp về nguyên tắc có thể được sử dụng để mã, khi mà phép mã X_i cho kết quả bởi h_i. Tất nhiên, các mode CBC và CFB thuộc lớp này.

Permutation Attack (P): trong trường hợp này H_i có thể viết như là $H_{i-1} \oplus f'(X_i)$ với f' là hàm một chiều: X_i không thể khôi phục từ H_i và H_{i-1}, nhưng giá trị băm là độc lập đối với thứ tự của các khối bản tin, điều đó có nghĩa là một tạo ảnh thứ hai hay va chạm dễ tìm thấy. Hơn nữa, có thể chèn một khối tin 2 lần. Các tấn công này là tầm thường, nếu như H_i chỉ phụ thuộc tuyến tính vào H_{i-1}.

Forward Attack (F): Cho H_{i-1}, H'_{i-1}, và X_i (chú ý rằng điều đó có nghĩa là H_i cố định), dễ tìm được X'_{i-1} sao cho $f(X'_i, H'_{i-1}) = f(X_i, H_{i-1}) = H_i$. Trong trường hợp này người ta có thể dễ dàng kiến thiết được một tạo ảnh thứ hai cho một giá trị băm đã cho, nhưng không cần thiết phải dễ tìm được tạo ảnh cho một phần tử đã cho trong miền giá trị.

Backward Attack (B): cho H_i, dễ tìm được cặp (X_i, H_{i-1}) sao cho $f(X_i,H_{i-1}) = H_i$. Trong trường hợp này sẽ là đơn giản việc tìm tạo ảnh (hay tạo ảnh thứ hai) cùng với giá trị khởi đầu ngẫu nhiên; tạo ảnh (hay tạo ảnh thứ hai) có thể tìm thấy nhờ tấn công gặp nhau ở giữa.

Fixed Point Attack (FP): tìm H_{i-1} và X_i sao cho $f(X_i,H_{i-1})=H_{i-1}$. Tấn công này không thực nguy hiểm: nếu hàm băm thoả mãn tính một chiều thì sẽ khó sinh ra bản tin dẫn đến giá trị cụ thể H_{i-1}.

Thứ tự của các tấn công này có một ý nghĩa quan trọng: khả năng của tấn công trực tiếp có nghĩa là tấn công backward và forward là cũng có thể, nhưng chiều ngược lại không đúng. Trong trường hợp của tấn công hoán vị, người ta có thể áp dụng tấn công backward bằng cách trước hết chọn X_i và sau đó tính H_{i-1}. Có thể chỉ ra rằng nếu cả hai tấn công forward và backward (hoặc permutation) là có thể thì tấn công direct cũng có thể. Phép chứng minh có trong bài báo đầy đủ.

4. Phân tích 64 lược đồ

Bảng 1 chỉ ra các tấn công có thể cho mỗi một trong số 64 lược đồ của mô hình tổng quát. Các tấn công được chỉ ra bởi chữ cái đầu của nó, trong khi đó "-" có nghĩa là hàm vòng f bị yếu một cách tầm thường như là một kết quả độc lập với một trong các đầu vào. Nếu không có 1 trong 5 dạng tấn công có thể áp dụng được, thì √ được đặt vào chỗ đó.

choice of FF	choice of K	choice of P			
		X_i	H_{i-1}	$X_i \oplus H_{i-1}$	V
V	X_i	-	B^{13}	B^{25}	-
	H_{i-1}	D^1	-	D^{26}	-
	$X_i \oplus H_{i-1}$	B^2	B^{14}	F^{27}	F^{41}
	V	-	-	D^{28}	-
X_i	X_i	-	B^{15}	B^{29}	-
	H_{i-1}	$\sqrt{}^3$	D^{16}	$\sqrt{}^{30}$	D^{42}
	$X_i \oplus H_{i-1}$	FP^4	FP^{17}	B^{31}	B^{43}
	V	-	D^{18}	B^{32}	-
H_{i-1}	X_i	P^5	FP^{19}	FP^{33}	P^{44}
	H_{i-1}	D^6	-	D^{34}	-
	$X_i \oplus H_{i-1}$	FP^7	FP^{20}	B^{35}	B^{45}
	V	D^8	-	D^{36}	-
$X_i \oplus H_{i-1}$	X_i	P^9	FP^{21}	FP^{37}	P^{46}
	H_{i-1}	$\sqrt{}^{10}$	D^{22}	$\sqrt{}^{38}$	D^{47}
	$X_i \oplus H_{i-1}$	B^{11}	B^{23}	F^{39}	F^{48}
	V	P^{12}	D^{24}	F^{40}	D^{49}

Bảng 1. Các tấn công đối với 64 lược đồ khác nhau. Các lược đồ được đánh số theo như các chỉ số phía trên.

Các lược đồ 18 và 28 tương ứng với CFB mode và CBC mode cho phép mã như đã chỉ ra trong [11, 14]; việc các mode này là hữu ích cho các hàm băm có khoá nhưng không đủ để kiến thiết các hàm băm một chiều như đã chỉ ra bởi S. Akl trong [1]. Điều này cũng có liên quan đến việc khả năng bảo vệ tính toàn vẹn được đưa ra các mode này là có giới hạn. Lược đồ 13 đã được đề xuất bởi Rabin năm 1978 [28]: tuy nhiên, R. Merkle đã chỉ ra rằng tấn công backward là có thể, từ đó suy ra rằng có thể tìm được một tạo ảnh bằng tấn công gặp ở giữa.

Đề xuất tiếp theo là lược đồ 14, cái đó thuộc về W. Bitzer trong [7,9]. R. Winternitz [31] đã chỉ ra rằng tấn công gặp nhau ở giữa bởi R. Merkle cũng áp dụng được cho lược đồ này. Ông cũng chỉ ra rằng các lược đồ 13 và 14 là bị tổn thương bởi tấn công khoá yếu: với khoá yếu K_w, DES thoả mãn tính chất $E(K_w,E(K_w,X))=X, \forall X$. Chèn hai lần khoá yếu vào như là khối tin sẽ làm cho hashcode không thay đổi trong mọi lược đồ.

Lược đồ an toàn đầu tiên (lược đồ 3) được đề xuất bởi S. Matyas, C. Meyer và J. Oseas trong [18]. Nó "đối ngẫu" với lược đồ 19, nó được qui cho D. Davies trong [31, 32], và được qui cho C. Meyer bởi D. Davies trong [8]. D. Davies đã khẳng định trong thông báo cá nhân tới các tác giả của bài báo này là ông không đề xuất ra lược đồ. Tuy vậy, lược đồ này được mọi người biết đến như là lược đồ Davies-Meyer (ví dụ, xem [23]). Việc lược đồ này bị tổn thương bởi tấn công điểm bất định được chỉ ra trong [25]. Lược đồ 10 đã được đề xuất bởi các tác giả và được

nghiên cứu trong [30]. Nó xuất hiện một cách độc lập trong [24] như là một mode của N-hash. Năm 1990, cũng chính lược đồ này được đề nghị bởi Nhật bản cho ISO/IEC [15]. Chuẩn quốc tế ISO/IEC 10118 Part 2 [16] chỉ ra hàm băm dựa trên mã khối bao gồm lược đồ 3. Lược đồ 20 được đề xuất như là một kiểu sử dụng cho mã khối LOKI trong [3]. Cuối cùng cũng chú ý rằng lược đồ 40 (cùng với tính bị tổn thương của nó bởi tấn công forward) được mô tả trong [18].

Tinh thần của phương pháp này là tất cả các lược đồ dựa trên mô hình tổng quát đã được phân loại một lần cho tất cả. Ưu điểm thứ hai là các tính chất của 12 lược đồ "an toàn" có thể được so sánh. Trước hết, sự phân loại tiếp theo sẽ được làm dựa vào biến đổi tương đương.

5. Các lớp tương đương

Một số lớn các lược đồ có thể được phân loại tiếp theo bởi xem xét các biến đổi tuyến tính của các đầu vào. Lớp các lược đồ nhận được từ 1 lược đồ bởi ánh xạ tuyến tính của các biến và được gọi là các lớp tương đương.

- Trong 7 lớp tương đương, hàm vòng phụ thuộc vào hai đầu vào độc lập (X_i và H_{i-1}), và 6 biến đổi có thể, vì có 6 ma trận nghịch đảo 2 x 2 trên GF(2). Có thể chỉ ra rằng trong 2 trường hợp, hàm vòng là an toàn hoặc bị tổn thương bởi tấn công điểm bất động, và trong 5 trường hợp hàm vòng bị tổn thương bởi tấn công trực tiếp, bởi tấn công hoán vị hoặc tấn công backward.

- Trong 7 lớp tương đương, hàm vòng phụ thuộc vào một đầu vào độc lập duy nhất. Vì có 3 đầu vào có thể, chính là X_i, H_{i-1} và $X_i \oplus H_{i-1}$ tương ứng với 3 vecto khác không độ dài 2 trên GF(2). Nếu hàm vòng phụ thuộc tổng của hai đầu vào, nó không phải là yếu một cách tầm thường. Tuy nhiên, nó bị tổn thương bởi tấn công trực tiếp (2 trong 7 trường hợp) hoặc tấn công forward (5 trong 7 trường hợp).

- Trong 1 lớp tương đương, hàm vòng đơn giản là hằng số.

CI	Characterization	class size	-	D	P	B	F	FP	√
0	FF = P, (P ≠ K)	6						4	2
	FF = P ⊕ K, (P ≠ K)	6						4	2
	FF = K, (P ≠ K)	6		2		4			
	P = K, (FF ≠ K)	6		2	2	2			
	FF = P = K	3	2				1		
1	FF = V, (P ≠ K)	6		2		4			
	P = V, (FF ≠ K)	6		2	2	2			
	K = V, (FF ≠ K)	6		4	1	1			
	FF = V, (P= K)	3	2				1		
	P = V, (FF = K)	3	2				1		
	K = V, (P = FF)	3	2				1		

2	FF = P =V FF = K = V P = K = V	3 3 3	2 2 2	 1 1			1		
3	FF = P = K = V	1	1						
Total		64	15	14	5	13	5	8	4

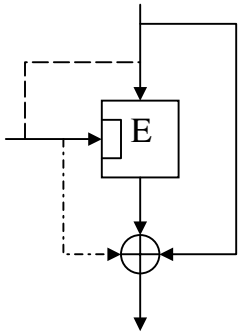
Bảng 2. Tổng hợp 15 phương án, được sắp xếp theo số các CI của các đầu vào hằng số.

Bảng 2 mô tả các lớp tương đương. Việc phân loại tiếp theo được dựa vào số CI của các hằng số đầu vào trong các lựa chọn. Để mô tả lớp, quan hệ được đưa ra giữa bản rõ P, khoá K và feedforward FF.

Chúng ta có thể kết luận rằng chỉ có 4 lược đồ là an toàn, còn 8 lược đồ chỉ bị tổn thương bởi tấn công điểm bất động. Cả 12 lược đồ đó được liệt kê (và được đánh số lại) trong Bảng 3 và được biểu diễn đồ hoạ trong hình 2. Vai trò của X_i và H_{i-1} tại đầu vào có thể được chọn một cách tự do, và mũi tên chấm-gạch là tuỳ chọn (nếu nó được đưa vào, thì khoá được cộng modulo 2 với bản mã). Còn đường gạch-gạch, thì có 3 khả năng có thể: nó có thể bỏ qua hoặc nó có thể chỉ từ khoá đến bản rõ hoặc từ bản rõ đến khoá. Hai lớp tương đương này là an toàn, các đại diện đơn giản nhất của chúng là lược đồ của Matyas và một số người khác (số 1) và lược đồ của Miyaguchi và các tác giả (số 3). Cho mỗi một trong các lược đồ này có thể viết một "chứng minh an toàn" dựa trên mô hình hộp đen của thuật toán mã, như đã được làm cho lược đồ Davies-Mayer (số 5) trong [32]. ý tưởng chính là việc tìm một giả tạo ảnh (pseudo-preimage) cho giá trị băm đã cho ít nhất sẽ khó như việc giải phương trình $H_i = f(X_i, H_{i-1})$ với giá trị đã cho của H_i. Số trung bình của việc tính f() được chỉ ra là 2^{n-1}.

no.	function expression
1	$E(H_{i-1},X_i)\oplus X_i$
2	$E(H_{i-1},X_i \oplus H_{i-1})\oplus X_i\oplus H_{i-1}$
3	$E(H_{i-1},X_i)\oplus X_i \oplus H_{i-1}$
4	$E(H_{i-1},X_i \oplus H_{i-1})\oplus X_i$
5	$E(X_i, H_{i-1})\oplus H_{i-1}$
6	$E(X_i,X_i \oplus H_{i-1})\oplus X_i\oplus H_{i-1}$
7	$E(X_i, H_{i-1})\oplus X_i\oplus H_{i-1}$
8	$E(X_i,X_i \oplus H_{i-1})\oplus H_{i-1}$
9	$E(X_i \oplus H_{i-1}, X_i)\oplus X_i\oplus$
10	$E(X_i \oplus H_{i-1}, H_{i-1})\oplus H_{i-1}$
11	$E(X_i \oplus H_{i-1}, X_i)\oplus H_{i-1}$
12	$E(X_i \oplus H_{i-1}, H_{i-1})\oplus X_i$

Bảng 3. Danh sách của 12 lược đồ an toàn cho hàm băm một chiều dựa trên mã khối và feeforward.



Hình 2. Cấu hình an toàn cho hàm băm một chiều dựa trên mã khối và feedforward

Trong bài báo đầy đủ, 12 lược đồ này sẽ được so sánh một cách chi tiết hơn dựa trên tính tổn thương đối với tấn công điểm bất định, các tấn công dựa trên tính yếu của mã khối nằm trong (trong trường hợp này là DES), và tấn công vi sai [2]. Tính hiệu quả của chúng cũng được so sánh. Các kết quả chính được tổng kết ở Bảng 4: cột 5 là đầu ra của hàm vòng nếu khoá là khoá yếu và bản rõ là một trong các điểm bất định tương ứng, cột 6 là √ nếu người ta có thể khai thác tính chất bù (complementation), cột cuối cùng chỉ ra xem biến nào cần được thay đổi nếu tấn công vi sai cùng với khoá cố định được sử dụng. (người ta có thể nghĩ về tấn công vi sai đối ngẫu, khi bản rõ được cố định và sai khác khoá đã cho được áp dụng; điều này không được xem xét ở đây).

no.	fixed points		properties if E=DES			differential attack
	X_i	H_{i-1}	rate R	K_w	compl.	
1	-	-	1	0	√	X_i
2	-	-	1	0	√	X_i
3	-	-	1	K_w	-	X_i
4	-	-	1	K_w	-	X_i
5	K	D(K,0)	n/k	0	√	H_{i-1}
6	K	D(K,K) ⊕ K	n/k	0	√	H_{i-1}
7	K	D(K,K)	n/k	K_w	-	H_{i-1}
8	K	D(K,0) ⊕ K	n/k	K_w	-	H_{i-1}
9	D(K,K)	D(K,K) ⊕K	1	0	√	X_i, H_{i-1}
10	D(K,0)⊕K	D(K,0)	n/k	0	√	H_{i-1}
11	D(K,0)	D(K,0)⊕K	1	K_w	-	X_i, H_{i-1}
12	D(K,K)⊕ K	D(K,K)	n/k	K_w	-	X_i, H_{i-1}

Bảng 4. Các tính chất của 12 lược đồ an toàn: các điểm bất động, các tính chất nếu DES được sử dụng như mã khối ở trong, và các biến có thể được thay đổi trong trường hợp tấn công vi sai.

Nếu k ≠ n, câu hỏi đặt ra là khả năng sử dụng các biến X_i và H_{i-1} có độ dài max(n,k) để tối đa hoá mức độ an toàn. ý tưởng là các bít không được sử dụng trong khoá hoặc như là bản rõ cần ảnh hưởng đến đầu ra thông qua một số EXOR. Khẳng định sau chỉ ra rằng điều đó là không thể đối với hàm băm tuân theo mô hình tổng quát của chúng ta; nó được chứng minh trong bài báo đầy đủ.

Khẳng định 1: Mức an toàn của hàm băm 1 chiều được xác định bởi số nhỏ nhất giữa k và n, trong đó k là độ dài của khoá còn n là độ dài khối.

6. Tăng cường của Merkle đối với lược đồ Rabin

Để tránh tấn công về phía sau trong trường hợp lược đồ Rabin (lược đồ 13), R. Merkle đã đề xuất mã bản tin theo mode CBC hoặc CFB (cùng với khoá ngẫu nhiên không bí mật K và giá trị khởi đầu IV) trước khi áp hàm băm vào [6]. Điều này gây ra hiệu suất giảm: tỷ lệ bằng 2. ý tưởng là đưa vào sự phụ thuộc giữa các khối tham gia vào hàm băm. Có thể chỉ ra rằng tấn công gặp ở giữa có thể được thay đổi để áp dụng cho mở rộng này:

- Tạo ra tập r bản tin sao cho khối mã cuối cùng của phép mã CBC cùng với giá trị khởi đầu IV và khoá K là bằng IV'; điều này có thể dễ làm bằng cách chọn tương ứng khối rõ cuối cùng (hoặc bằng tấn công gặp nhau ở giữa)
- Tạo ra tập thứ hai gồm r bản tin và mã những bản tin này theo CBC mode với giá trị khởi đầu IV' và khoá K
- Vì hai phần của bản tin bây giờ là độc lập, người ta có thể sử dụng hai tập các bản tin "đã mã hoá" trong một tấn công gặp ở giữa đơn thuần.

Điều đó chỉ ra rằng việc tìm tạo ảnh chỉ yêu cầu $O(2^{n/2})$ phép mã.

7. Mở rộng

Cũng phương pháp này có thể áp vào các hàm băm có khoá (hoặc Message Authentication Code) dựa trên mã khối. Kết quả chính trong trường hợp này là hàm vòng

$$f= E(K, X_i \oplus H_{i-1}) \oplus X_i$$

là được ưu tiên với các mode CBC và CFB, chúng đã được chỉ ra trong phần lớn các chuẩn (xem [13]). Chú ý rằng một bảo vệ phụ là cần thiết ở cuối. Chi tiết hơn có thể đọc ở bài báo đầy đủ.

Thiết kế của các hàm băm dựa trên phép bình phương cũng có ích từ quan điểm kiến thiết này. Bởi vì không có khoá, nên chỉ có 16 trường hợp được xem xét. Phần lớn các hàm băm trong trường hợp này dựa độ an toàn của nó vào sự việc lấy căn bậc 2 là khó đối với người không biết phân tích của modulo. 7 trong số các lược đồ như vậy là yếu một cách hiển nhiên, 4 (trong số 9 cái còn lại) đã xuất hiện trong tài liệu in. Kết luận chính là hàm vòng

$$f = (X_i \oplus H_{i-1})^2 \bmod N \oplus X_i$$

là có hứa hẹn nhất. Để nhận được hàm băm an toàn, một lược đồ có dư thừa cần phải được chỉ ra. Độ dư thừa là cần thiết để loại bỏ việc khai thác cấu trúc đại số

giống như các điểm bất định của phép bình phương modulo và các số "nhỏ", khi đó không có sự rút gọn không xảy ra [12, 27].

Cuối cùng, cũng cần chỉ ra rằng cấu trúc tương tự có thể được khai thác trong các hàm băm đặc chủng giống như họ MD4 [29] và Snefru [21]. Việc phân tích đối với tấn công vi sai và các điểm bất định có thể được chuyển sang cho các lược đồ này.

8. Kết luận

Việc kiến thiết các hàm băm mật mã dựa trên các mã khối rõ ràng là bài toán khó. Trong tóm tắt mở rộng này, phép xử lý tổng quát đã được phát triển cho trường hợp đơn giản nhất, chính là độ dài của giá trị băm bằng với độ dài khối và độ dài khoá. Phương pháp này cho phép nhận ra các lược đồ an toàn và so sánh chúng theo một số chỉ tiêu. Sẽ là hữu ích việc nghiên cứu hàm băm có khoá, hàm băm dựa trên phép tính số học modulo và các hàm băm chuyên dụng.

Tài liệu tham khảo

1. S.G. Akl, "On the security of compressed encodings", *Advances in Cryptology, Proc. Crypto'83*, D. Chaum, Ed., Plenum Press, New York, 1984, pp.209-230.
2. E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems", *Journal of Cryptology*, Vol.4, No.1, 1991, pp.3-72.
3. L. Brown, J. Pieprzyk, and J. Seberry, "LOKI- a cryptographic primitive for authentication and secrecy applications," *Advances in Cryptology, Proc. Auscrypt'90*, *LNCS 453*, J. Seberry and J. Pieprzyk, Eds., Springer-Verlag, 1990, pp. 229-236.
4. I. B. Damgard, "Collision free hash functions and public key signature schemes," *Advances in Cryptology, Proc. Eurocrypt'87, LNCS 304*, D. Chaum and W.L. Price, Eds., Springer-Verlag, 1988, pp.203-216.
5. I. B. Damgard, "A design principle for hash functions," *Advances in Cryptology, Proc. Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp.416-427.
6. D. Davies and W. L. Price, "The application of digital signature based on public key cryptosystems," *NPL Report* DNACS 39/80, December 1980.
7. D. Davies, "Applying the RSA digital signature to electronic mail," *IEEE Computer*, Vol. 16, February 1983, pp.55-62.
8. D. Davies and W. L. Price, "Digital signature, an update," *Proc. 5th International Conference on Computer Communication*, October 1984, pp.845-849.
9. D. Denning, "Digital signatures with RSA and other public-key cryptosystems," *Communications ACM*, Vol. 27, April 1984, pp. 388-392.
10. FIPS 46, "*Data Encryption Standard*," Federal Information Processing Standard, National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.

11. FIPS 81, "*DES Modes of operation*," Federal Information Processing Standard, National Bureau of Standards, U.S. Department of Commerce, Washington D.C., December 1980.
12. M. Girault, "Hash-functions using modulo-n operations," *Advances in Cryptology, Proc. Eurocrypt'87, LNCS 304*, D. Chaum and W.L. Price, Eds., Springer-Verlag, 1988, pp.217-226.
13. ISO/IEC 9797, "*Information technology-Data cryptographic techniques- Data integrity mechanisms using a cryptographic check function employing a block cipher algorithm,*" 1993.
14. ISO/IEC 10116, "*Information technology- Security techniques-Modes of operation of an n-bit block cipher algorithm,*" 1991.
15. "*Hash functions using a pseudo random algorithm,*" ISO-IEC/JTC1/SC27/WG2 N98, Japanese contribution, 1991.
16. ISO/IEC 10118, "*Information technology-Security techniques- Hash-functions- Part 1: General and Part 2: Hash-functions using an n-bit block cipher algorithm*," 1993.
17. X. Lai and J.L. Massey, "Hash functions based on block ciphers," *Advances in Cryptology, Proc. Eurocrypt'92, LNCS 658*, R.A. Rueppel, Ed., Springer-Verlag, 1993, pp.55-70
18. S.M. Matyas, C.H. Meyer, and J. Oseas, "Generating strong one-way functions with cryptographic algorithm," *IBM Techn. Disclosure Bull.*, Vol.27, No.10A, 1985, pp.5658-5659.
19. R. Merkle, "*Secrecy, Authentication, and Public Key Systems*," UMI Research Press, 1979.
20. R. Merkle, "One way hash functions and DES," *Advances in Cryptology, Proc. Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp.428-446.
21. R. Merkle, "A fast software one-way hash function," *Journal of Cryptology*, Vol.3, No.1, 1990, pp.43-58.
22. C. H. Meyer and M. Schilling, "Secure program load with Manipulation Detection Code," *Proc. Securicom* 1988, pp.111-130.
23. C. Mitchell, F. Piper, and P. Wild, "Digital signatures," in "*Contemporary Cryptology: The Science of Information Integrity*," G. J. Simmons, Ed., IEEE Press 1991, pp.325-378.
24. S. Miyaguchi, M. Iwata, and K. Ohta, "New 128-bit hash function," *Proc. 4th International Joint Workshop on Computer Communcations* , Tokyo, Japan, July 13-15, 1989, pp.279-288.
25. S. Miyaguchi, K. Ohta and M. Iwata, "Confirmation that some hash functions are not collision free," *Advances in Cryptology, Proc. Eurocrypt'90, LNCS 473*, I.B. Damgard, Ed., Springer-Verlag, 1991, pp.326-343.
26. B. Preneel, R. Govaerts, and J. Vandewalle, "On the power of memory in the design of collision resistant hash functions," *Advances in Cryptology, Proc. Auscrypt'92, LNCS 718*, J. Seberry and Y. Zheng, Eds., Springer-Verlag, 1993, pp.105-121.

27. B. Preneel, “Cryptographic hash functions, “ Kluwer Academic Publishers, 1994.
28. M. O. Rabin, “Digitalized signatures,” in “Foundations of Secure Computation,” R. Lipton and R. DeMillo, Eds., Academic Press, New York, 1978, pp. 155-166.
29. R. L. Rivest, “The MD4 message digest algorithm,” *Advances in Cryptology, Proc. Crypto’90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp.303-311.
30. K. Van Espen and J. Van Mieghem, “*Evaluatie en Implementatie van Authentiseringsalgoritmen (Evaluation and Implementation of Authentication Algorithms-in Dutch)*,” ESAT laboratorium, Katholieke Universiteit Leuven, Thesis grad. eng., 1989.
31. R. S. Winternitz, “Producing a one-way hash function built from DES,” *Advances in Cryptology, Proc.Crypto’83,* D. Chaum, Ed., Plenum Press, New York, 1984, pp.203-207.
32. R.S. Winternitz, “A secure one-way hash function built from DES,” *Proc.IEEE symposium on Information Security and Privacy 1984*, 1984, pp.88-90
33. G, Yuval, “How to swindle Rabin,” *Cryptologia*, Vol.3, 1979, pp. 187-189.

Nguyên tắc thiết kế cho hàm băm

Ivan Bjerre Damgard, Eurocrypt 91, pp. 416-427

Tóm tắt: Chúng tôi chỉ ra rằng nếu tồn tại một hàm không va chạm về mặt tính toán f từ m bit vào t bit với m > t, thì tồn tại hàm không va chạm theo quan niệm tính toán h ánh xạ các bản tin có độ dài đa thức bất kỳ vào chuỗi t-bit.

Giả sử n là độ dài của bản tin. h có thể kiến thiết sao cho nó có thể đánh giá trong thời gian tuyến tính theo n khi sử dụng 1 bộ vi xử lý, hoặc là mất O(log(n)) thời gian khi sử dụng O(n) bộ vi xử lý, việc tính hàm f được coi như là 1 bước. Cuối cùng, với mỗi hằng số k và n lớn, việc tăng tốc theo hệ số k trên kiến thiết thứ nhất là có thể khi sử dụng k bộ vi xử lý.

Ngoài việc đưa ra nguyên tắc thiết kế mạng nói chung cho hàm băm, kết quả của chúng tôi cho một cái nhìn tổng quát đối với các kiến thiết rõ ràng là không có liên quan với nhau được đề xuất trước đây. Nó cũng đề nghị các thay đổi đối với các kiến thiết đã được đề nghị khác để làm cho việc chứng minh tính an toàn của nó là khá dễ dàng.

Chúng tôi đưa ra 3 thí dụ cụ thể của việc kiến thiết, dựa trên phép bình phương lấy modulo, bộ tạo bít giả ngẫu nhiên Wolfram [Wo], và bài toán balô.

1. Mở đầu và các tài liệu có liên quan

Hàm băm h được gọi là *không va chạm*, nếu nó ánh xạ các bản tin có độ dài bất kỳ vào các chuỗi có độ dài cố định, sao cho việc tìm x, y để h(x) = h(y) là khó. Chú ý rằng chúng ta tập trung ở đây vào các hàm băm việc tính toán được công bố công khai, tức là các hàm không bị kiểm soát bởi khoá.

Nhu cầu của các hàm này để đảm bảo toàn vẹn dữ liệu, và để sử dụng với các lược đồ chữ ký số là đã được biết- ví dụ xem [Da], [De], [DP].

Một số kiến thiết của các hàm băm đã được đề xuất, ví dụ như dựa trên DES [Wi], [DP] hoặc trên RSA [DP], [Gir]. Kiến thiết trong [Da] đầu tiên là để cho tính không va chạm có thể chứng minh được, giả thiết rằng các phép toán thành phần mà nó dựa vào là an toàn, đó chính là tính một chiều của phép bình phương lấy modulo trong trường hợp này, còn một cách tổng quát hơn, đó là sự tồn tại của cặp claw-free của các hoán vị. Ví dụ sau ở trong [Gi]. Không may, tính chất lý thuyết thú vị đó có nghĩa là sự suy giảm tính hiệu quả so với các đề xuất khác: thời gian cần thiết cho các hàm này đúng bằng việc dùng RSA cho cả bản tin.

Bài toán kiến thiết hàm băm không va chạm chứng minh được và nhanh hãy còn chưa có lời giải.

Nhiều cái khó để đưa ra một chứng minh cho các kiến thiết đã biết là do mọi cái trở nên phức tạp khi độ dài của bản tin được băm tăng lên. Mặt khác, hàm băm sẽ không sử dụng được, nếu chúng ta không cho phép băm các bản tin có độ dài bất kỳ.

Trong bài báo này, chúng ta chỉ ra rằng cái khó đó có thể loại bỏ, nếu như một kiến thiết tốt được sử dụng: it turns out thay ability to cut just 1 bit off the length of a message in a collision free way implies ability to hash message of arbitrary lengths. Phép chứng minh đưa ra một nguyên lý thiết kế mạnh nền tảng, nó có thể sử dụng như là một chỉ dẫn khi thiết kế các hàm băm mới hay khi cải tiến một cái đã có.

Kiến thiết của chúng ta rất giống với phương pháp "meta" do Merkle phát kiến ra một cách độc lập [Me]; khi so sánh, thì kiến thiết này chứa đựng một số phần tử thêm cho phép một phép chứng minh hình thức có thể mà không cần đến một số giả thuyết thêm đối với các tham số của hàm số.

Cũng có nhiều cái tương tự với các phương pháp được sử dụng độc lập bởi Naor và Yung [NaYu]. Họ cũng chứng minh rằng các hàm băm độ dài cố định có thể được soạn ra để nhận được bản nén cho thông báo có độ dài đa thức bất kỳ. Điều đó được làm cho cả loại hàm băm của chúng ta và cho các hàm băm cùng với tính chất yếu hơn một chút: kẻ thù cho phép chọn x, và sau đó chọn một trường hợp cá biệt (instance) ngẫu nhiên từ họ hàm băm. Naor và Yung kiến thiết các hàm kiểu này từ một hoán vị 1 chiều bất kỳ, và dùng nó để xây dựng các lược đồ chữ ký số.

Từ quan điểm thực tế, kiến thiết của chúng ta là trực tiếp và hiệu quả hơn. Đó là vì để cho kiến thiết hàm băm trong [NaYu] làm việc với các hàm băm có tính chất yếu hơn, họ cần phải chọn một trường hợp riêng độc lập mới của hàm băm độc dài cố định cho mỗi khối tin để xử lý.

Cuối cùng, công trình độc lập rất gần đây của Impagliazzo và Naor [ImNa] cần được nhắc đến. Họ chứng minh rằng hàm băm được kiến thiết từ bài toán knapsack theo như cách ở trong mục 4.3 của bài báo này là an toàn theo nghĩa của [NaYu] nếu như knapsack được dùng đem lại hàm một chiều.

2. Chuẩn bị

Trước hết, chúng ta định nghĩa khái niệm họ các hàm băm không va chạm.

Định nghĩa 2.1

Họ hàm băm không va chạm độ dài cố định $\mathcal{F}$ là họ vô hạn các tập hữu hạn $\{F_m\}_{m=1}^{\infty}$ và hàm t: N$\rightarrow$ N sao cho t(m) < m với mọi m $\in$ N. Phần tử của F_m là hàm f: $\{0,1\}^m \rightarrow \{0,1\}^{t(m)}$, và được gọi là một trường hợp riêng của $\mathcal{F}$ có độ dài

m. $\mathcal{F}$ cần phải thoả mãn các tính chất sau:

1. Có thuật toán thời gian đa thức xác suất (theo m) Θ sao cho, khi cho giá trị của m, chọn được một trường hợp riêng của $\mathcal{F}$ có kích thước m một cách ngẫu nhiên.
2. Với mỗi trường hợp riêng $f \in \mathcal{F}$ bất kỳ và $x \in \{0,1\}^m$, f(x) có thể tính trong thời gian đa thức.
3. Cho một trường hợp riêng $f \in \mathcal{F}$ được chọn ngẫu nhiên như trong 1), khó tìm được $x, y \in \{0,1\}^m$ sao cho f(x) = f(y) và $x \neq y$.

Một cách hình thức hơn: với thuật toán thời gian đa thức xác xuất bất kỳ Δ (theo m), và đa thức bất kỳ P, xét tập con các trường hợp riêng f có độ dài m mà đối với nó thuật toán Δ với xác xuất ít nhất 1/P(m) cho ra $x \neq y$ nhưng lại có f(x)=f(y). Giả sử ε(m) là xác xuất mà Θ chọn một trong các trường hợp riêng đó. Thế thì như một hàm số của m, ε(m) tiến tới không nhanh hơn bất kỳ một phân số nào có mẫu là đa thức.

Các điều kiện 1) và 2) nói rằng $\mathcal{F}$ là hữu ích trong thực hành: các trường hợp riêng có thể được chọn, và giá trị hàm được tính có hiệu quả. Điều kiện 3) khẳng định tính không va chạm.

Một trong các điều ngụ ý cơ bản của 3) là các hàm số trong $\mathcal{F}$ có tính một chiều như sau:

Bổ đề 2.1

Giả sử $\mathcal{F}$ là họ hàm không va chạm, f là trường hợp riêng có kích thước m. Giả sử P_f là phân bố xác suất trên $\{0,1\}^{t(m)}$ được sinh ra bằng cách chọn x ngẫu nhiên và phân bố đều trong $\{0,1\}^m$ và tính hàm f(x).

Khi đó không có thuật toán tính ngược f trên miền ảnh được chọn theo P_f thành công với xác suất lớn hơn $\frac{1}{2}+\frac{1}{P(m)}$ với mọi đa thức P.

Nếu P_f là phân bố đều trên tập ảnh của f hoặc nếu m-t là O(m) thì không có thuật toán tính ngược thành công với xác suất lớn hơn $\frac{1}{P(m)}$.

Chứng minh:

Giả sử bổ đề là sai. Cho thuật toán Δ có thể tính ngược f với xác xuất ít nhất ½+1/P(m), chúng ta chọn x phân bố đều và cho f(x) như là đầu vào của Δ. Nếu Δ là thành công, chúng ta được y sao cho f(x) = f(y). Giả sử A là sự kiện rằng tạo ảnh của f(x) có kích cỡ ít nhất bằng 2. $\{0,1\}^m$ ít nhất lớn gấp 2 lần so với ảnh của f, và A xảy ra với xác suất lớn hơn ½. Cho nên, theo giả thuyết cho Δ, nó thành công với xác xuất ít nhất 1/P(m) khi A xảy ra. Rõ ràng là, việc Δ chọn phần tử nào trong số các tạo ảnh của f(x) để đưa cho chúng ta là không tương quan với việc

chọn x (khi cho f(x)). Cho nên x ≠ y với xác xuất ít nhất bằng ½, nếu như A xảy ra và Δ là thành công.

Với khẳng định thứ hai, chú ý rằng nếu P_f là phân bố đều, thì thành công của Δ là không tương quan với việc xảy ra của A. Nếu m-t = O(m), thì A xảy ra với xác xuất áp đảo. Trong trường hợp ngược lại, Δ cho chúng ta va chạm với xác xuất cơ bản là 1/P(m).

Cuối cùng, chúng ta định nghĩa khái niệm về họ hàm băm không va chạm.

Định nghĩa 2.2

Họ hàm băm không va chạm $\mathcal{H}$ là họ vô hạn các tập hữu hạn $\{H_m\}_{m=1}^{\infty}$ và hàm bị chặn đa thức t: N→ N. Phần tử của H_m là hàm h: $\{0,1\}^m \rightarrow \{0,1\}^{t(m)}$, và được gọi là trường hợp riêng của $\mathcal{H}$ có kích thước m. $\mathcal{H}$ cần phải có các tính chất sau:

1. Cho giá trị của m, có thuật toán thời gian đa thức xác suất (theo m) Θ với đầu vào m chọn được một trường hợp riêng của $\mathcal{H}$ có kích thước m một cách ngẫu nhiên.
2. Với mỗi trường hợp riêng $h \in H_m$ và $x \in \{0,1\}^*$, h(x) là dễ tính, tức là tính được trong thời gian đa thức theo cả m và |x|.
3. Khi cho một trường hợp riêng $h \in \mathcal{H}$ được chọn ngẫu nhiên như trong 1), khó tìm được $x, y \in \{0,1\}^*$, sao cho h(x)=h(y) và x ≠ y.

Một cách hình thức: Với mỗi thuật toán Δ thời gian đa thức xác suất, và đa thức bất kỳ P, xét tập con các trường hợp riêng h kích thước m, mà đối với nó Δ có xác xuất ít nhất 1/P(m) cho ra x ≠ y sao cho h(x) =h(y). Giả sử ε(m) là xã suất để cho Θ chọn một trong các trường hợp riêng đó. Như một hàm số của m, ε(m) sẽ tiến tới 0 nhanh hơn bất kỳ một phân số nào có mẫu số là đa thức.

3. Các kiến thiết cơ sở

Kết quả chính trong phần này là họ các hàm băm không va chạm có thể được kiến thiết từ họ hàm băm không va chạm kích thước cố định.

Định lý 3.1

Giả sử $\mathcal{F}$ là họ hàm băm không va chạm độ dài cố định ánh xạ m bit vào t(m) bit. Khi đó tồn tại họ hàm băm không va chạm $\mathcal{H}$ ánh xạ chuỗi có độ dài bất kỳ vào các chuỗi t(m)-bit.

Giả sử h là một trường hợp riêng của $\mathcal{H}$ có kích thước m. Việc tính h với đầu vào có kích thước n có thể làm trong nhiều nhất n/(m-t(m)+1)+1 bước khi dùng 1 bộ xử lý (chúng ta tính hàm trong $\mathcal{F}$ như là một bước).

Chứng minh.
Với mỗi trường hợp riêng f $\in \mathcal{F}$ với kích thước m, chúng ta sẽ kiến thiết trường hợp riêng của h $\in \mathcal{H}$ kích thước m. Đặt t=t(m). Với các chuỗi bit a, b, chúng ta ký hiệu a||b là phép nối của a và b.

Việc kiến thiết được chia ra làm 2 trường hợp: trước hết là trường hợp m-t > 1, sau đó là trường hợp m-t=1.

Chúng ta mô tả xem hàm h được tính như thế nào đối với đầu vào x $\in \{0,1\}^*$:
Chúng ta chia x ra thành các khối m-t-1 bit. Nếu khối cuối cùng là chưa đủ thì ta thêm 0 vào. Giả sử d là số các số 0 cần thiết. Giả sử các khối được ký hiệu bởi $x_1,x_2,...,x_{n/(m-t-1)}$, với n=|x| (là độ dài sau khi padding).

Chúng ta thêm vào dãy này một khối nữa $x_{n/(m-t-1)\_+1}$ có (m-t-1) bit, nó chứa biểu diễn nhị phân của d, nó có tiền tố là một số thích hợp các số 0.

Sau đó, ta định nghĩa dãy các khối có t bit h_0, h_1,...bởi:
$h_1 = f(0^{t+1}||x_1)$
$h_{i+1}= f(h_i ||1||x_{i+1})$
Cuối cùng, đặt $h(x)= h_{n/(n-m)+1}$.

Việc kiểm tra rằng $\mathcal{H}$ thoả mãn các điều kiện 1 và 2 trong Định nghĩa 2.2 là dễ. Còn với điều kiện 3, giả thuyết ngược lại rằng chúng ta có thuật toán Δ có thể tìm x≠ x’ sao cho h(x) =h(x’).

Giả sử h_i, x_i (tương ứng $h_i^{'}, x_i^{'}$) là các kết quả trung gian trong việc tính h(x) (tương ứng h(x’)).

Nếu |x| ≠ |x’| mod (m-t) thì sẽ có $x_{n/(m-t)+1}$≠ $x'_{n'/(m-t)+1'}$ cho nên h(x)=h(x’) cho chúng ta ngay lập tức va chạm của f. Cho nên bây giờ chúng ta giả thiết rằng |x| =|x’| mod (m-t) và không mất tính tổng quát rằng |x’| ≥ |x|.

Hãy xem xét đẳng thức:
$h(x)= f(h_{n/(m-t)}||x_{n/(m-t)+1}) = f(h'_{n'/(m-t)}||x'_{n'/(m-t)+1}) = h(x')$

Nếu $h_{n/(m-t)}||x_{n/(m-t)+1} \neq h'_{n'/(m-t)}||x'_{n'/(m-t)+1}$, chúng ta có va chạm của f, và thế là xong. Nếu không, chúng ta xét thay vào đó đẳng thức
$f(h_{n/(m-t)-1}||x_{n/(m-t)}) = f(h'_{n'/(m-t)-1}||x'_{n'/(m-t)})$

và lặp lại lập luận. Rõ ràng là quá trình này cần phải dừng bằng cách hoặc là tạo ra va chạm của f (vì x ≠ x’), hoặc là thiết lập ra đẳng thức
$0^{t+1}||x_1 = h'_i||1||x'_{i+1,}$

điều này là không thể.

Tóm lại, chúng ta có một phép suy dẫn, nó biến đổi Δ thành một thuật toán tìm va chạm cho f. Giả sử Δ thao tác nhiều nhất T(m) bit. Thế thì x và x' cần phải có độ dài nhỏ hơn T(m).

Khi đó cả phép suy dẫn chiếm thời gian O(T(m)F(m)), với F(m) là thời gian cần thiết để tính một giá trị của f, đặc biệt, nếu T là đa thức, thì cả phép suy dẫn sẽ trong thời gian đa thức. Điều này cuối cùng mâu thuẫn với điều kiện 3 trong Định nghĩa 2.1.

Cuối cùng, chúng ta bàn đến trường hợp m-t=1. Dễ dàng, nhưng là giải pháp không hiệu quả đó là việc mã prefix-free tất cả các bản tin trước khi đem băm, sau đó sử dụng kiến thiết tương tự như ở trên, thay đổi định nghĩa của h bằng

$h_1 = f(0^t||x_1)$
$h_{i+1} = f(h_i||x_{i+1})$

Ở đây, tất nhiên x_i là các khối 1-bit. Cái đó có thể chứng minh là an toàn theo như cùng một cách như trên.

Nếu f thoả mãn điều kiện thức hai trong Bổ đề 2.1, thì có một giải pháp hiệu quả hơn: chúng ta chọn chuỗi y_0 dài t-bit theo phân bố đều và định nghĩa:

$h_1 = f(y_0||x_1)$
$h_{i+1}= f(h_i||x_{i+1})$

lần này không cần phải mã prefix free cho x. Lập luận ở trên sẽ chỉ ra rằng va chạm cho h hoặc sẽ đem lại va chạm cho f hoặc tạo ảnh của y_0. Nhưng khi đó chúng ta sử dụng Bổ đề 2.1.

Các chú ý:

- Phiên bản cuối của phép kiến thiết sử dụng Bổ đề 2.1, nó sẽ chỉ không làm việc khi m-t=1, nhưng sẽ làm việc khi P_f là gần với phân bố đều, hoặc m-t = 0(m). Điều này cần nhận biết bởi vì phiên bản này cho phép băm nhiều hơn 1 bit mỗi lần áp dụng hàm f so với kiến thiết tổng quát, nên nó hiệu quả hơn một chút.
- Mẹo trong chứng minh ở trên là việc ghép khối thêm vào bản tin chỉ cần để đảm bảo rằng chúng ta có thể nhận thấy sự khác biệt giữa các bản tin mà cần phải thêm vào d số 0 với các bản tin kết thúc đơn giản bằng d số 0 trước khi padding. Trong nhiều ứng dụng, hoàn toàn chấp nhận được là các số 0 phía sau trong khối cuối cùng là bỏ qua được, trong trường hợp đó phần đó của kiến thiết có thể bỏ qua.

Chúng ta hãy xem mối liên quan giữa Định lý 3.1 và các hàm băm đã biết. Trong [Da], các hàm băm được kiến thiết từ các cặp claw-free các hoán vị, tức là các

hoán vị (f_0, f_1) có cùng miền xác định D, sao cho việc tìm $x \neq y$ và $f_0(x) = f_1(y)$ là bài toán khó. Chúng ta có thể kiến thiết trường hợp riêng trong họ hàm không va chạm từ cặp (f_0, f_1) bằng cách định ra hàm f: D x $\{0,1\} \rightarrow$ D bởi

$$f(x, b) = f_b(x)$$

với $x \in D$ và $b \in \{0, 1\}$. Sử dụng Định lý 3.1 đối với họ hàm được định nghĩa như vậy sẽ dẫn đến chính xác cá hàm băm được trình bày ở [Da], ngoại trừ việc chúng ta bỏ nhu cầu thêm prefix free cho bản tin được sử dụng ở đây (P_f là phân bố đều trong trường hợp này).

Trong ví dụ thứ hai, xem xét một trong các ý tưởng đầu tiên để thiết kế hàm băm từ mã kinh điển, bắt nguồn từ Rabin: Giả sử E là thuật toán mã, nó mã bản tin có độ dài t bit bằng khoá có độ dài k bit. Giả sử m=t+k. Chúng ta chia bản tin m thành các khối k bit là $x_1,...x_n$. Chúng ta chọn khối h_0 có t bit cố định một cách ngẫu nhiên và đặt $h_{i+1} = E_{x_{i+1}}(h_i)$. h(x) được định nghĩa bằng h_{n+1}.

Cái đó khớp với điều kiện của Định lý .1 bởi việc lấy $f(a,b)=E_a(b)$ với b là khối t bit còn a là khối k-bit. Không may, hàm f này không phải là không va chạm: phép mã bản tin bất kỳ với khoá bất kỳ và giải mã cùng với khoá *khác* sẽ đem lại va chạm của f với xác xuất cao. Cái đó không có nghĩa hàm là yếu. Cái đó có nghĩa là, phép chứng minh độ an toàn không thể chỉ dựa vào tính chất của riêng hàm f, mà còn phụ thuộc vào cấu trúc tổng thể của hàm.

Với ví dụ cụ thể, điểm yếu có thể tìm thấy, tuy nhiên: nếu DES được sử dụng như E, sao cho t chỉ bằng 64, thì đã biết rằng với hàm f được kiến thiết như trên, sẽ cho phép kẻ địch khi có x và h(x) thì tìm được $y \neq x$ sao cho h(x) = h(y), bằng cách dùng "nghịch lý ngày sinh". Đó là khẳng định mạnh hơn so với việc nói rằng hàm băm không có tính không va chạm.

Trong ISO, hiện tại đang đề xuất việc chuẩn hoá một sửa đổi của lược đồ này, với f được định nghĩa là $f(a,b)= E_a(b) \oplus b$. Với phiên bản này của b, có thể hy vọng là chúng ta có thể sử dụng Định lý 3.1 để chứng minh tính không va chạm của hàm toàn thể bằng cách chỉ xem xét hàm f: Cho c, không dễ giải phương trình f(a,b)=c với ẩn số a và b, nếu E là một thuật toán mạnh, và có đủ lý do để tin rằng phiên bản này của f là không va chạm.

3.1 Các hàm băm song song

4. Các kiến thiết cụ thể

Sau đây, chúng tôi đưa ra 3 kiến thiết cụ thể cho các hàm không va chạm với độ dài đầu vào cố định. Những hàm này có thể biến thành các hàm băm bằng cách áp dụng trực tiếp Định lý 3.1.

4.1 Dựa trên bình phương lấy modulo

Trước hết chúng ta đưa ra thiết kế dựa trên độ khó của việc lấy căn bậc 2 modulo số lớn là tích của hai số nguyên tố. Kiến thiết có nhiều điểm tương tự với hàm được mô tả ở trong [Gir], nhưng điểm khác cơ bản là các hàm ở trong [Gir] không cho phép áp dụng Định lý 3.1.

Giả sử n=pq với p và q là các số nguyên tố lớn. Giả sử độ dài của n là s bit. Trong thực tế, có thể nghĩ s=512. Sau đó, chọn I là một tập con nào đó của các số 1,2,..., s. Với chuỗi có s-bit bất kỳ, $y=y_1,y_2,...,y_s$, giả sử $f_I(y)$ là phép nối của các y_j với $j \in I$.

Cuối cùng, chúng ta có thể định nghĩa hàm không va chạm dự tuyển f từ m bit vào t bit bằng cách đặt m=s-8, t=|I|, và định nghĩa

$$f(x) = f_I((3F|x)^2 \bmod n)$$

với 3F|x là ký hiệu nối của byte 3F với x. Từ phép nối suy ra rằng tất cả các đầu vào của hàm bình phương modulo nhỏ hơn n/2 nhưng đủ lớn để đảm bảo việc có lấy modulo. Chúng ta tránh các va chạm đơn giản do $x^2 = (-x)^2 \bmod n$, và cố gắng tìm các va chạm bằng cách chọn ví dụ các luỹ thừa 2 nhỏ như đầu vào.

Chúng ta cũng cần chọn I sao cho f an toàn và hiệu quả. Bài toán tìm va chạm cho f có thể phát biểu lại: tìm các số $x \neq y$ sao cho bình phương modulo n của chúng trùng nhau tại các vị trí thuộc I. Girault [Gir] chỉ ra cách làm điều đó như thế nào nếu I là tập tương đối bé (nhơ hơn 64) các bít lớn nhất hay các bit nhỏ nhất.

Điều đó có nghĩa là chúng ta nên chọn các vị trí có thể phát tán đều trên toàn bộ s khả năng có thể, vì phương pháp của Girault và những cái có liên quan [GTV] sẽ hỏng. Mặt khác, có những lý do thực tế để không sử dụng các vị trí hoàn toàn ngẫu nhiên, nhưng ít nhất là gộp chúng vào các byte. Hơn nữa, |I| cần phải chọn không quá bé, để chống lại "va chạm ngày sinh".

Cụ thể, nếu s= 512, thì đề nghị một lựa chọn tốt là |I|= 128, và f_I lấy ra một byte trong số 4 byte.

Hàm này sẽ băm được đến 376 bit / bình phương modulo (??), các đó trên máy IBM P/S II model 80 cho tốc độ đến 100 Kbit/s. Một phần cứng đặc biệt cho tốc độ lên đến vùng Mbit/sec.

4.2 Dựa vào Bộ tạo bit giả ngẫu nhiên của Wolfram

Đề nghị thứ hai dựa vào bộ tạo bit giả ngẫu nhiên được đề xuất bởi Wolfram [Wo]. Nói chung, bộ tạo bit giả ngẫu nhiên là thuật toán lấy đầu vào là mầm ngắn hoàn toàn ngẫu nhiên, và kéo dài nó thành các chuỗi đầu ra dài, dường như ngẫu nhiên. Một cách trực giác, rõ ràng là bộ tạo như vậy cần phải theo nghĩa nào đó áp

dụng hàm một chiều vào mầm để có đầu ra: nếu mầm là dễ tìm khi cho một đoạn đầu ra thì cả đầu ra sẽ tính được và như thế thì không còn ngẫu nhiên nữa.

Tuy nhiên, tính một chiều nhìn chung không suy ra tính không va chạm của hàm được kiến thiết trực tiếp từ bộ tạo- việc phân tích trường hợp cụ thể là rất cần thiết.

Bây giờ chúng ta xem xét thuật toán được đề nghị bởi Wolfram: ta định nghĩa hàm g từ các chuỗi n bit sang các chuỗi n bit: giả sử $x=x_0, x_1,..., x_{n-1}$ thì bit thứ i của g(x) là

$$g(x)_i = x_{i-1} \oplus (x_i \text{ V } x_{i+1})$$

với phép cộng 1 và trừ 1 được lấy theo modulo n. Người ta có thể nghĩ đó là thanh ghi R trong đó các bit được cập nhật đồng thời bởi R:=g(R). Cái đó được biết như là một cellular automat một chiều.

Sử dụng cái đó để tạo bit giả ngẫu nhiên, ta làm như sau:

1. Chọn x ngẫu nhiên
2. x:=g(x)
3. Lấy ra x_0. Quay lại bước 2.

Trong [Wo], bộ tạo bit giả ngẫu nhiên này đã được phân tích, kết quả của nhiều kiểm tra thống kê đã được đưa ra. Độ an toàn của nó được chứng minh chống lại kẻ thù có giới hạn một số dạng tính toán, còn khả năng của nó đánh bại kẻ thù có thời gian tính toán đa thức bất kỳ còn là giả thuyết. Tuy nhiên, tất cả những cái gì đã biết, chỉ ra rằng đây là bộ tạo thực sử mạnh.

Giả sử các bit được sinh ra bởi thuật toán từ đầu vào x được ký hiệu bởi $b_1(x)$, $b_2(x)$,...Con đường tự nhiên sử dụng cái đó để kiến thiết hàm không va chạm là chọn 2 số tự nhiên c <d và lấy hàm f_0 bằng

$$f_0(x) = b_c(x), b_{c+1}(x),..., b_d(x).$$

Có hai chỗ hổng có thể trong ý tưởng này, cần phải cẩn thận:
Thứ nhất, yêu cầu tự nhiên là hàm như vậy là tất cả các bit đầu ra phải phụ thuộc vào tất cả các bit đầu vào. Dễ thấy rằng việc thay đổi 1 bit của x thì dần dần sau nhiều lần thực hiện x:=g(x) sẽ ảnh hưởng tới toàn bộ x- nhưng hiệu ứng chỉ lan chậm trong thanh ghi: 1 bit về bên phải và khoảng 0.25 bit về bên trái cho một lần áp dụng g [Wo]. Cho nên việc chọn c rất nhỏ là nguy hiểm. Giá trị nhỏ nhất tự nhiên của c cần phải là cái đảm bảo cho tất cả các bit đầu ra phụ thuộc vào tất cả các bit đầu vào.

Vấn đề thứ hai là bản thân g không phải là một hàm không va chạm: ví dụ, $g(1^n)=g(0^n) = 0^n$. Rõ ràng, g(x) = g(y) suy ra $f_0(x) = f_0(y)$. Tổngquát hơn, nếu $g^v(x)=g^v(y)$ với v nhỏ, thì ít nhất khả năng để cho $f_0(x) = f_0(y)$ không phải là không đáng kể.

Một cách tự nhiên để tránh khỏi khó khăn đó là giới hạn f vào tập con của các chuỗi n-bit, như thế sẽ hạ thấp khả năng các cặp x, y như trên tồn tại hoặc ít nhất cũng làm cho việc tìm chúng khó. Một khả năng cụ thể là giới hạn cho các dãy có dạng x||z, với z là hằng số chuỗi được chọn ngẫu nhiên trong $\{0,1\}^r$, r < n.

Chúng ta sẽ định nghĩa ứng cử viên f sao cho nó ánh xạ chuỗi (n-r) bit x vào chuỗi

$$f(x) = f_0(x||z).$$

Bổ đề sau cung cấp bằng chứng cho phương pháp này:

Bổ đề 5.1
Nếu $g^v(x) = g^v(y) = z$, và 2v bit liên tiếp của x và y là bằng nhau thì x=y,

Chứng minh.
Giả sử j là chỉ số của vị trí ngay bên phải của 2v bit mà chúng ta biết là bằng nhau. Mỗi bit của z phụ thuộc nhiều nhất 2v+1 bit của x (hoặc y). Giả sử l được chọn sao cho z_l là hàn của các bit j,..., j+2v của x (hoặc y). Bây giờ, vì phép lấy ngược x_j sẽ lấy ngược z_l , giả thuyết của chúng tôi suy ra rằng $x_j = y_j$. Bây giờ chúng ta có thể 'trượt' các lập luận trên về phía phải một vị trí.□

Cái đó cung cấp bằng chứng tốt rằng việc chọn r tương đối lớn sẽ cho các va chạm 'tầm thường" thưa ra và khó tìm hơn.

Như một ví dụ cụ thể, giả sử chúng ta chọn n=512, r=256, c=257 và d=384. f thu được sẽ ánh xạ các chuỗi 256 bit vào các chuỗi 128 bit và hàm băm được kiến thiết từ f theo Định lý 3.1 sẽ băm các bản tin trong các khối 128 bit và tạo ra 128 bit đầu ra.

Hàm này đặc biệt thích hợp cho cài đặt phần cứng: trong các mạch VLSI, người ta có thể tính g bằng cách cập nhật tất cả các bit song song, và việc tính g sẽ mất 1 hay 2 nhịp đồng hồ, không phụ thuộc vào n. Cái đó sẽ đem lại tốc độ khoảng Mbit/sec ngay với cả các tốc độ đồng hồ xử lý trung bình. Hơn nữa, một phần cứng như thế rất rẻ và dễ xây dựng.

4.3 Dựa vào bài toán Knapsack
Mặc dù bài toán knapsack là NP-đầy đủ, và có thể rất khó trong trường hợp tồi nhất, nhưng sử dụng tính khó này trong mật mã không dễ, cái đó được chỉ ra bởi số phận của nhiều hệ mật khoá công khai dựa trên bài toán này.

Cái khó, phần lớn là ở chỗ phép mã cần phải có ngược. và knapsacks được sử dụng cần phải có một số cấu trúc built-in, trong nhiều trường hợp trở nên hữu ích đối với mã thám. Mặt khác, hàm băm không bao giờ có ngược, cho nên knapsacks

được sinh ra hoàn toàn ngẫu nhiên được sử dụng.

Con đường tự nhiên để làm việc này là chọn ngẫu nhiên các số $a_1,..., a_s$ trong khoảng 1..M, với s là độ dài tối đa của bản tin. Chúng ta có thể băm bản tin nhị phân $m_1,m_2,..., m_s$ bằng

$$f(m_1,...,m_s) = \sum_{i=1}^{s} m_i a_i$$

Như đã chỉ ra trong [GC], cái đó hoàn toàn không an toàn với s lớn, chính xác hơn khi $M \le s^{\log(s)/4}$. Để giải quyết vấn đề này, chúng ta đề xuất cố định s bằng một giá trị nhỏ vừa phải so với M, và sử dụng Định lý 3.1 để kiến thiết hàm băm thực sự. Như một ví dụ, có thể chọn s bằng 2log(M), điều đó có nghĩa là f nén các khối s bit thành các khối s/2 bit và điều kiệncủa [GC] còn lâu mới được thoả mãn.

Lựa chọn cụ thể là s=256 và $M = 2^{120}-1$. Nó cho hàm băm cuối cùng có độ dài 128 bit. Trên máy IBM P/S II Model 80, phiên bản này chạy với tốc độ 250 Kbit/s.

Để mô tả hàm chúng ta cần chỉ ra 4 KB dữ liệu. Trên máy PC có 640 K RAM, điều này là bình thường. Nhưng trong trường hợp bộ nhớ nhỏ hơn, chúng ta phải trả giá về thời gian vì bộ nhớ và tạo ra các số a giả ngẫu nhiên thay cho việc nhớ chúng.

Chỉ tiêu khác về sức mạnh của hàm là ở chỗ bài toán quyết định tổng quát là knapsack đã cho sinh ra một ánh xạ injective có là bài toán co-NP hay không? Va chạm tất nhiên là bằng chứng của tính không song ánh (bài toán quyết định là đơn giản nếu knapsack nén đầu vào, nhưng không có nghĩa là dễ tìm bằng chứng).

Chúng ta chú ý rằng ngay cả những knapsacks nới rộng đầu vào ra một chút (như thế thì không dùng được bởi [ImNa]) cũng có thể dùng để xây dựng hàm băm an toàn theo nghĩa của [NaYu], nếu giả thiết rằng chúng tạo ra các ánh xạ không va chạm. Điều này dường như là hợp lý theo nghĩa tính đầy đủ co-NP của bài toán tham gia vào và việc bài toán quyết định là không tầm thường trong trường hợp này. Cách kiến thiết và chứng minh có thể nhận được bằng cách phỏng theo kỹ thuật của [NaYu].

Tài liệu

[Da] Damgard: “Collision Free Hash Functions and Public Key Signature Schemes”, Proceedings of EuroCrypt 87, Springer.

[De] D. Denning: “Digital Signature with RSA and other Public Key Cryptosystems”, CACM, vol.27, 1984, pp.441-448

[DP] Davis and Price: “The Application of Digital Signatures Based on Public Key Crypto-Systems”, Proc. of CompCon 1980, pp. 525-530

[GC] Godlewski and Camion: ”Manipulation and Errors, Localization and Detection”, Proceedings of EuroCrypt 88, Springer.

[Gi] Gibson, A Collision Free Hash Functions and the Discrete Logarithm Problem for a Composite Modulus, Manuscript, 1/10/88, London, England.

[Gir] Girault, Hash Functions Using Modulo- Operations, Proceedings of Eurocrypt 87, Springer

GTV Girault, Toffin and Valée, Computation of Approximate L-th Roots Modulo n and Application to Cryptography, Proceedings of Crypto 88, Springer.

[ImNa] Impagliazzo and Naor: Eficient Cryptographic Schemes Provably as Secure as Subset Sum”, Proc. of FOCS 89.

[Me] Merkle, One Way hash Functions, Proc. of STOC 89.

[NaYu] Naor and Yung, Universal One-Way Hash Functions” Proc. of STOC 89.

[Wi] Winternitz, Producing a one-way Hash Function from DES, Proceesings of Crypto 83, Springer

[Wo] Wolfram, Random Sequence Generation by Cellular Automata, Adv., Appl. Math., vol 7, 123-169, 1986.

Hàm băm nhanh an toàn dựa trên mã sửa sai

Lars Knudsen and Bart Preneel

Crypto 97, p.485-498

Tóm tắt: Bài báo này xem xét các hàm băm dựa trên mã khối. Nó trình bày một tấn công mới đối với hàm nén của hàm băm 128-bit MDC-4 sử dụng DES với một độ phức tạp ít hơn nhiều so với dự kiến, nó đề xuất các kiến thiết mới cho các hàm nén nhanh và an toàn dựa trên các mã sửa sai và mã khối m-bit có khoá m-bit. Điều này dẫn đến các kiến thiết hàm băm đơn giản và thực tế dựa trên các mã khối như DES, khi mà độ dài khoá nhỏ hơn một chút so với độ dài khối; như IDEA, khi mà độ dài khoá gấp đôi độ dài khối và còn dẫn đến các hàm băm kiểu MD4. Dưới các giả thiết hợp lý về các mã khối nằm trong, chúng ta nhận được các hàm nén không va chạm. Cuối cùng, chúng tôi cung cấp các ví dụ kiến thiết hàm băm dựa trên cả DES và IDEA hiệu quả hơn nhưng đề xuất đã có và thảo luận việc áp dụng phương pháp của chúng ta đối với các hàm băm kiểu MD4.

1. Mở đầu

Các hàm băm mật mã ánh xạ một xâu có độ dài bất kỳ vào một xâu ngắn có độ dài cố định, dài 128 hoặc 160 bit. Các hàm băm được sử dụng trong các ứng dụng mật mã như chữ ký số, lược đồ bảo vệ mật khẩu, xác thực văn bản kinh điển. Cho các ứng dụng này, người ta yêu cầu rằng khó tìm được đầu vào tương ứng với giá trị băm đã cho (tấn công tìm tạo ảnh) hoặc một đầu vào thứ hai có cùng giá trị băm với giá trị đã cho (tấn công tìm tạo ảnh thứ hai). Hơn nữa, nhiều ứng dụng cũng yêu cầu rằng khó tìm được 2 đầu vào có cùng giá trị băm (tấn công va chạm). trong bài báo này chúng tôi xem xét các hàm băm thoả mãn 3 tính chất đó.

Tất cả các hàm băm quan trọng là các hàm băm lặp dựa trên hàm nén dễ tính h(.,.) từ hai dãy nhị phân có độ dài tương ứng m và l ánh xạ vào dãy nhị phân độ dài m. Văn bản M được chia thành các khối M_i có l bit, $M=(M_1,...,M_n)$. Nếu độ dài của M không là bội của l thì M được kéo dài bằng một thuật toán kéo dài không nhập nhằng. Kết quả băm Hash(IV, M) = H = H_t nhận được bởi việc tính lặp

$$H_i = h(H_{i-1},M_i) \qquad i=1,2,..,t \qquad (1)$$

với H_0= IV là giá trị khởi đầu định trước. Các tấn công va chạm, tìm tạo ảnh và tìm tạo ảnh thứ hai có thể áp dụng voà cả hàm nén và hàm băm. Phần còn lại của bài báo này sẽ không phân biệt 2 loại tấn công cuối và gọi cả hai là tấn công tìm tạo ảnh.

Có thể liên hệ độ mật của hàm Hash(.) với độ mật của h(.,.) trong một số mô hình [2, 12, 15, 17]. Để làm việc này, cần phải thêm một khối vào cuối của chuỗi đầu vào, trong đó có chứa độ dài của nó, được biết như là phép tăng cường MD (viết tắt của Merkle [15] và Damgard [2]), và ta có kết quả sau:

Định lý 1: Giả sử Hash(.) là hàm băm lặp có tăng cường MD, thế thì các tấn công tìm tạo ảnh và tấn công va chạm đối với Hash(.) (khi người tấn công có thể chọn IV một cách tự do) có độ phức tạp chính bằng đối các tấn công tương ứng trên h(.,).

Định lý 1 cho cận dưới cho độ mật của Hash(.). Phần lớn các hàm băm thực tế không có hàm nén không va chạm và không xử lý 2 đầu vào của hàm nén theo cách như trên; ngoại trừ các hàm băm dựa trên DES của Merkle [15] và các hàm của chính các tác giả bài viết này [11]. Như một ví dụ, các va chạm của hàm nén MD5 đã được trình bày ở [3, 6].

Chúng ta hãy xem xét các hàm băm dựa trên các mã khối có độ dài khối m và độ dài khoá k. Để cho tiện lợi, chúng tôi sẽ giả thiết rằng k là bội của m. Một mã khối như vậy, với mỗi khoá k, định ra một hoán vị trên tập các dãy m-bit. Ưu điểm của các hàm băm dựa trên mã khối là cực tiểu hoá công sức thiết kế và cài đặt. Hơn nữa, độ mật của mã khối, với một chừng mực nào đó, có thể chuyển sang cho hàm băm. Một điểm khác biệt là mã khối cần thoả mãn một số tính chất nào đó ngay cả khi khoá đã biết (ví dụ xem [19]). Nhược điểm chính của hướng dùng mã khối là các hàm băm đặc chủng dường như hiệu quả hơn. Tuy nhiên, các tấn công của Dobbertin [4, 6] trên các hàm băm đặc biệt như MD4 [21] và MD5[22] đã chỉ ra rằng tính hiệu quả này cần phải trả giá rất đắt, điều này làm cho việc kiến thiết hàm băm dựa trên mã khối càng trở nên hấp dẫn.

Chúng ta định nghĩa tỷ lệ băm (hash rate) của hàm băm dựa trên mã khối m-bit như là số các khối m-bit được xử lý trên một phép mã hay dịch. Độ phức tạp của tấn công là tổng số các phép toán, tức là số phép mã hay dịch cần thiết để người tấn công thành công với xác suất cao.

Các kiến thiết an toàn đầu tiên cho hàm băm dựa trên mã khối là lược đồ Matyas, Meyer và Oseas [14], nó đã được chỉ ra trong ISO/IEC 10118 [9], và cái đồng hành với nó là lược đồ Davies-Meyer. Cả hai lược đồ là các hàm băm độ dài khối duy nhất cho kết quả băm chỉ có m bit với tỷ lệ 1. Việc phân loại các lược đồ này đã được trình bày bởi Preneel trong [18]. Sử dụng tấn công ngày sinh, các va chạm cho lược đồ này có thể tìm thấy trong khoảng $2^{m/2}$ phép toán và tấn công tìm tạo ảnh tìm thấy trong 2^m phép toán. Tuy nhiên, vì phần lớn các mã khối có độ dài khốim = 64 bit, các va chạm có thể tìm thấy chỉ trong 2^{32} phép toán và cần các hàm băm có giá trị băm dài hơn. Mục đích của các hàm băm có độ dài khối đúp trước hết là để đạt được độ mật chống lại tấn công va chạm ít nhất là 2^m phép toán. Với tất cả các kiến thiết có tỷ lệ 1 của một lớp lớn, các tác giả của bài báo này đã chỉ ra rằng các va chạm (cho H) có thể tìm thấy với không nhiều hơn $2^{3m/4}$ phép toán [11]. Khi dùng DES, các kiến thiết tốt nhất được biết là MDC-2 với tỷ lệ ½ [1, 9], và MDC-4 với tỷ lệ ¼ [1]; với cả hai, tấn công va chạm được tin là cần ít nhất 2^m phép toán. Một lớp quan trọng của các kiến thiết thuộc về Merkle [15].

Trong [12], Lai và Massey đã đề xuất 2 kiến thiết cho các hàm băm dựa trên mã khối IDEA (với m=64 và k= 128): Abrest-DM và Tandem-DM có tỷ lệ ½ và khẳng định độ an toàn chống lại tấn công va chạm là 2^m phép toán.

Tuy nhiên, đối với các kiến thiết này, ngoại trừ cho những cái thuộc họ Merkle, không có chứng minh về độ an toàn; còn đối với MDC-2 và MDC-4 thì va chạm của hàm nén có thể tìm thấy nhanh hơn bởi tấn công vét cạn. Hơn nữa, với các mã khối có m=64 (ví dụ như DES hay IDEA), các kiến thiết khối đúp không cung cấp độ an toàn chấp nhận được đối với các tấn công va chạm vét cạn song song: suy ra từ [20, 23] rằng mức an toàn ít nhất $2^{75}...2^{80}$ phép mã được yêu cầu, không một đề xuất nào gần đây đưa ra cái đó. Trong [11] chúng tôi phát triển một khung mới để kiến thiết các hàm băm dựa trên các mã khối m-bit với hoá m-bit bằng cách dùng mã tuyến tính trên $GF(2^2)$. Các kiến thiết đã được chỉ ra rằng để tìm được va chạm yêu cầu ít nhất $2^{qm/2}$ phép toán (với $q \geq 2$), và để tìm tạo ảnh cần ít nhất 2^{qm} phép mã với điều kiện bộ nhớ trong phải nhiều. Với q=2, các kiến thiết là hiệu quả hơn những đề xuất đã có với cùng mức an toàn.

Trong bài báo này, trước hết chúng tôi chỉ ra rằng các va chạm cho hàm nén của MDC-4 có thể tìm thấy trong thời gian $2^{3m/4}$. Với DES, một số bit khoá cần cố định để tránh khoá yếu và tính chất bù (complementation property), cho nên độ phức tạp cho tấn công của chúng ta chỉ là 2^{41}, điều này ngày nay là hoàn toàn có thể. Vì tỷ lệ băm của MDC-4 là ¼, chỉ có 1 khối được xử lý trong 4 phép mã, chúng ta có thể chờ đợi độ mật cao cho hàm nén. Thú hai, ta tổng quát hoá phương pháp của [11]. Chúng tôi đề xuất phương án thiết kế hàm băm dựa trên các mã khối có khoá lớn hơn và làm việc trên các khối nhỏ hơn. Chính xác hơn, chúng ta xem xét các mã khối có khoá tm-bit ($t \geq 1$) bằng cách dùng các mã tuyến tính trên $GF(2^s)$, $s \geq 2$). Các kiến thiết của chúng ta đem lại các hàm băm hiệu quả hơn so với trong [11]. Sử dụng DES và IDEA, chúng tôi chỉ ra rằng có thể nhận được các tỷ lệ băm gần với 1 hoặc 2 với mức độ an toàn cao chống lạc các tấn công va chạm.

Về sau, $E_K(X)$ và $D_K(Y)$ ký hiệu phép mã/dịch cho bản rõ X có m-bit tương ứng với bản mã Y với khoá K có tm-bit. Giả thiết rằng mã khối không có điểm yếu, có nghĩa là trong các tấn công lên hàm băm dựa trên mã khối, các tấn công tắt lên mã khối không giúp gì được người tấn công.

Phần cuối của bài báo này được tổ chức như sau. Trong mục 2 chúng tôi phân tích hàm nén MDC-4. Các kiến thiết mới được trình bày tong mục 3 và trong mục 4 chúng tôi cung cấp các ví dụ về các kiến thiết có hiệu quả sử dụng DES, IDEA và các hàm tựa MD-4, chúng được ký hiệu là họ MDx.

2. Hàm nén của MDC-4

MDC-2 và MDC-4 [1] là các hàm băm dựa trên mã khối: chúng được biết như là các hàm băm Meyer-Schilling, mang tên các tác giả của bài báo đầu tiên mô tả chúng [16]. MDC-2 có ở trong Phần 2 của ISO/IEC 10118, đó là chuẩn về hàm băm dựa trên mã khối [9]. MDC-2 có thể mô tả như sau, với $h(X, Y) = E_X(Y) \oplus Y$ và $E_K(.)$ là phép mã bởi mã khối m-bit bởi khoá K:

$$T1_i = h(u(H1_{i-1}), X_i) = LT1_i || RT1_i$$
$$T2_i = h(v(H2_{i-1}), X_i) = LT2_i || RT2_i$$
$$H1_i = LT1_i \,||\, RT2_i$$
$$H2_i = LT2_i \,||\, RT1_i$$

Các biến $H1_0$ và $H2_0$ được khởi đầu bởi các giá trị IV_1 và IV_2 tương ứng, còn kết quả băm là phép nối $H1_t$ và $H2_t$. Tỷ lệ của lược đồ này bằng ½. Chuẩn ISO/IEC không chỉ ra một mã khối cụ thể nào cả; nó cũng đòi hỏi chỉ ra 2 ánh xạ u, v từ không gian mã vào không gian khoá sao cho $u(IV_1) \neq v(IV_2)$. Với DES, u và v chính là việc bỏ qua các bít kiểm tra và cố định bít thứ 2 và bit thứ 3 bằng 10 và 01 tương ứng, điều này loại trừ các tấn công dựa trên các khoá yếu và các khoá nửa yếu. Hàm nén của MDC-2 thực ra là không phải là không va chạm: với X_i cố định chọn trước bất kỳ, người ta có thể tìm được các va chạm cho cả $H1_i$ và $H2_i$ một cách độc lập cùng với tấn công ngày sinh đơn giản đòi hỏi $2^{m/2}$ phép toán.

Một lần lặp của MDC-4 [1] được định nghĩa như là phép nối của 2 bước MDC-2, với các bản rõ trong bước thứ hai chính là $H2_{i-1}$ và $H1_{i-1}$:

$$T1_i = h(u(H1_{i-1}), X_i) = LT1_i || RT1_i$$
$$T2_i = h(v(H2_{i-1}), X_i) = LT2_i || RT2_i$$
$$U1_i = LT1_i \,||\, RT2_i$$
$$U2_i = LT2_i \,||\, RT1_i$$
$$V1_i = h(u(U1_i), H2_{i-1}) = LV1_i || RV1_i$$
$$V2_i = h(v(U2_i), H1_{i-1}) = LV2_i || RV2_i$$
$$H1_i = LV1_i \,||\, RV2_i$$
$$H2_i = LV2_i \,||\, RV1_i$$

Tỷ lệ của MDC-4 bằng ¼. Rõ ràng là việc "trao đổi" của $H1_{i-1}$ và $H2_{i-1}$ trong bước thứ hai không làm tốt hơn thuật toán: sau khi đổi các nửa bên phải, $U1_i$ và $U2_i$ là đối xứng đối với $H1_{i-1}$ và $H2_{i-1}$.

Việc tìm các va chạm cho hàm nén là khó hơn so với trường hợp MDC-2. Mặt khác, các va chạm cho hàm nén MDC-2 cùng với các giá trị khác nhau của X_i với cùng giá trị của ($H1_{i-1}$, $H2_{i-2}$) cũng đem lại va chạm cho MDC-4, nhưng nhìn chung, tính chất này không đúng cho các va chạm khác cho các hàm cơ sở giống như các tựa-va chạm (pseudo-collisions), nghĩa là các va chạm mà tất cả các đầu vào của hàm nén đều thay đổi.

Sau đây, chúng tôi sẽ chỉ ra tấn công va chạm đối với hàm nén cùng với độ phức tạp nhỏ hơn tấn công vét cạn:

1. Chọn giá trị ngẫu nhiên X_i và $H2_{i-1}$ (với i=1 đó chính là giá trị khởi đầu).
2. Tính $T1_i$ cho $2^{3m/4}$ giá trị $H1_{i-1}$. Người ta mong đợi tìm được chuỗi S có m/2-bit sao cho với một tập gồm $2^{m/4}$ các giá trị (của $H1_{i-1}$) thì các bít ứng với LT_i bằng chính S (thực ra, với xác xuất 50% của các chuỗi S sẽ có $2^{m/4}$ trường hợp như vậy hoặc hơn).
3. Tính $V2_i$ cho $2^{m/4}$ đầu vào trong tập này. Xác suất để nhận được va chạm cho $V2_i$ sẽ bằng $(2^{m/4})^2/ 2^{m+1}= 2^{-m/2+1}$.
4. Nếu không tìm được cái trùng nhau, người ta chọn giá trị mới cho S; có thể tránh việc tính lại $T1_i$ nếu lưu trữ $2^{m/4}.2^{m/2} = 2^{3m/4}$ đại lượng m-bit.

Công sức trù tính để tìm được va chạm cho hàm nén của MDC-4 là $2^{3m/4}$ phép mã và cần lưu trữ $2^{3m/4}$ đại lượng m-bit. Các khoá của MDC-4 khi dùng DES chỉ gồm 54-bit [1]. Trong trường hợp này, tấn công va chạm yêu cầu khoảng 2^{41} phép mã DES và lưu trữ $2^{30.5}$ đại lượng 54-bit (khoảng 10 GB).

3. Kiến thiết bằng mã tuyến tính

Trong phần này, chúng tôi trình bày các kiến thiết mới cho các hàm băm chống được va chạm dựa trên mã khối m-bit và mã tuyến tính. Các kiến thiết này mở rộng kiểu băm đơn giản đã được tin là an toàn ra kiểu băm bội. Như là kiểu băm đơn giản, chúng ta sẽ sử dụng hàm băm Davies-Meyer:

$$h_i(M_i, H_{i-1}) = E_{M_i}(H_{i-1}) \oplus H_{i-1} \qquad (2)$$

Bất kỳ hàm nào trong số 12 hàm băm an toàn độ dài khối đơn được mô tả trong [18] đều có thể được sử dụng. Ưu điểm của việc sử dụng hàm băm Davies-Meyer là ở chỗ nó được định nghĩa cho các mã khối với kích thước khối và kích thước khoá khác nhau. Giả thuyết sau được thừa nhận trong mật mã học ngày nay:

Giả thuyết 1. Giả sử $E_K(.)$ là mã khối m-bit có khoá K dài tm-bit với số nguyên t > 0. Lấy hàm nén h là hàm Davies-Meyer (2). Việc tìm các va chạm cho h đòi hỏi khoảng $2^{m/2}$ phép mã (của khối m-bit) và việc tìm tạo ảnh cho h đòi hỏi khoảng 2^m phép mã.

Định nghĩa 2. (Davies-Meyer bội). Giả sử $E_K(.)$ là mã khối m-bit có khoá K dài tm-bit với số nguyên t > 0. Giả sử $h_1, h_2,...h_n$ là các phiên bản khác nhau của hàm Davies-Meyer, có nghĩa là $h_i(X_i, Y_i) = E_{X_i}(Y_i) \oplus Y_i$, chúng nhận được bằng cách cố định $\lceil \log_2 n \rceil$ bit khoá (hoặc bản rõ) bởi các giá trị khác nhau, trong đó X_i là các chuỗi tm-bit và Y_i là các chuỗi m-bit. Hàm nén của lược đồ Davies-Meyer bội gồm r khối đầu vào m-bit, chúng được mở rộng bởi một ánh xạ affine vào n cặp (X_i, Y_i). Đầu ra là phép nối kết quả của các hàm $h_1,..., h_n$. Đầu ra của hàm ném sẽ phụ thuộc vào tất cả r khối vào, nói một cách khác, ma trận của ánh xạ affine có hạng đầy đủ.

Các hàm con $h_i(X_i, Y_i)$ được gọi là *tích cực*, nếu trong tấn công va chạm hoặc tấn công tìm tạo ảnh các khối đầu vào tạo nên (X_i, Y_i) là khác nhau. Hai hàm con $h_i(X_i, Y_i)$ và $h_j(X_j, Y_j)$ có thể bị tấn công *độc lập*, nếu người ta có thể thay đổi một (hay nhiều) các khối đầu vào cho hàn nén, sao cho các tham số (X_i, Y_i) của h_i thay đổi, trong khi các tham số (X_j, Y_j) của h_j thì không.

Chúng ta đưa ra giả thuyết sau, khác với giả thuyết đã đưa ra trong [11] một chút.

Giả thuyết 2. Giả sử rằng va chạm hoặc tạo ảnh cho hàm nén của lược đồ Davies-Meyer bội đã tìm được, một cách đồng thời cho $h_1,...,h_n$. Giả sử N là số các hàm con tích cực và giả sử N-v là số cực đại trong số N hàm con có thể tấn công độc lập. Khi đó giả thuyết rằng việc nhận được va chạm hay tạo ảnh đòi hỏi tương ứng ít nhất $2^{vm/2}$ hoặc 2^{vm} phép mã.

Chú ý rằng trong nỗ lực tìm va chạm hoặc tạo ảnh cho lược đồ Davies-Meyer bội luôn có thể cố định một số khối đầu vào và như vậy là cố định các đầu ra.Giả sử N là số các hàm con tích cực. Cái mà Giả thuyết 2 nói là, nếu N-v trong số N hàm có thể tấn công độc lập (riêng biệt), thì sẽ không tồn tại một tấn công tốt hơn tấn công vét cạn đối với v hàm còn lại. Chú ý rằng trong độ phức tạp tổng thể của tấn công va chạm (hoặc tìm tạo ảnh) chúng ta không đưa vào độ phức tạp của tấn công đối với N-v hàm, chính điều này làm cho giả thuyết của chúng ta mạnh và có thể.

3.1 Các kiến thiết mới

Định lý sau chỉ ra xem các hàm băm dựa trên mã khối có thể kiến thiết như thế nào bằng cách dùng các mã sửa sai tuyến tính không nhị phân. Nó mở rộng định lý chính của [11].

Định lý 3. Nếu tồn tại mã [n, k, d] trên $GF(2^{t+1})$ có độ dài n, kích cỡ k và khoảng cách cực tiểu d, với $(t+1) > n$, $t \geq 1$ và $m >> \log_2 n$, thì tồn tại hàm băm song song dựa trên mã khối m-bit với khoá tm-bit, đối với nó việc tìm va chạm cho hàm nén đòi hỏi ít nhất $2^{(d-1)m/2}$ phép mã và việc tìm tạo ảnh đòi hỏi ít nhất $2^{(d-1)m}$ phép mã nếu Giả thuyết 2 là đúng. Hàm băm có bộ nhớ trong n.m bit và tỷ lệ $(t+1)k/n - 1$.

Chứng minh: Hàm nén chứa n hàm khác nhau h_i với $1 \leq i \leq n$, xem Định nghĩa 2. Đầu vào của hàm nén chứa (t+1)k khối m-bit: n biến từ H_{i-1}^1 đến H_{i-1}^n (đầu ra của n hàm từ vòng lặp trước), r khối tin từ M_i^1 đến M_i^r, với r=(t+1)k-n > 0. Sau đây, mỗi bít riêng lẻ của các khối m-bit đó được xử lý theo cùng một cách. Các bit của (t+1) khối đầu vào liên tiếp được nối lại để tạo nên k phần tử của $GF(2^{t+1})$. Mỗi một phần tử như vậy biểu diễn các đầu vào (t+1)-bit của một trong n hàm, cụ thể, một bit biểu diễn đầu vào khối rõ, còn t bit còn lại biểu diễn đầu vào khoá cho mã khối. Các bit đầu vào riêng biệt nhận được bằng cách biểu diễn các phần tử của $GF(2^{t+1})$ như là không gian vecto trên GF(2). Kiến thiết này đảm bảo rằng các điều

kiện của Giả thuyết 2 được thoả mãn cho giá trị v = d-1. Suy ra từ khoảng cách cực tiểu của mã rằng ít nhất d hàm con là chủ động trong va chạm. Cũng vậy, k là số lớn nhất các hàm con mà có thể tấn công độc lập. Nhưng do n-k ≥ d-1 theo cận Singleton [13, trang 33] nên suy ra kết quả cần thiết.□

Việc tồn tại của các kiến thiết hiệu quả cho d=3 và d=4 suy ra từ việc tồn tại các mã Hamming hoàn thiện trên $GF(2^t)$ (xem [13]) với các tham số $n=(q^s-1)/(q-1)$, k=n-s, d=3 cho luỹ thừa nguyên tố q, và từ sự tồn tại của các mã mở rộng 3 lần MDS (phân tách khoảng cách cực đại –maximum distance separable) [13, Chương 11, Định lý 10] với các tham số $n=q^s+2$, $k=q^s-1$, d=4 cho luỹ thừa nguyên tố chẵn q.

Hệ quả 4. Giả sử rằng Giả thuyết 2 đúng, tồn tại các hàm băm song song dựa trên mã khối m-bit với khoá tm-bit có tỷ lệ gần bằng t sao cho việc tìm va chạm (tìm tạo ảnh) cần ít nhất $2^m(2^{2m})$ phép toán (hoặc $2^{3m/2}(2^{3m})$ phép toán).

Chứng minh: Từ Định lý 3 suy ra rằng tồn tại các hàm băm với tỷ lệ (t+1)k/n-1. Bởi vì với mã Hamming $n/k \to 1$ với n lớn, ta suy ra kết quả.□

Kết quả này suy ra rằng khi trả giá có bộ nhớ trong lớn, sử dụng DES có thể nhận được các hàm băm tỷ lệ 1 và sử dụng IDEA có thể nhận được hàm băm tỷ lệ 2. Để so sánh MDC-2 và MDC-4 có tỷ lệ tương ứng ½ và ¼, còn Abreast-DM và Tandem-DM cho IDEA [12] có tỷ lệ ½.

Trong [11] chúng tôi đưa ra ví dụ về hàm băm dựa trên mã khối m-bit có khoá m-bit dùng mã [8, 5, 3] trên $GF(2^2)$. Mã nhận được bằng cách chặt ngắn mã Hamming [21, 18, 3]. Hàm băm có tỷ lệ ¼ và bộ nhớ trong 8.m bit. Sau đây chúng tôi chỉ ra rằng có thể làm tốt hơn kết quả này.

Ý tưởng là chia các từ m-bit thành các khối nhỏ hơn và sử dụng mã trên trường lớn hơn. Ví dụ, giả sử chúng ta có mã khối m-bit với khoá m-bit và m chẵn. Trong Định lý 3, các mã trên $GF(2^3)$ được sử dụng, khi mà 2 bit của từ mã biểu diễn bản rõ tương ứng với đầu vào khoá cho mã khối. Phương pháp khác là chia tất cả các khối m-bit thành các khối có m/2 bit và sử dụng mã trên $GF(2^4)$. Ưu điểm của phương pháp này (sẽ được minh hoạ về sau) là các cận tốt hơn tồn tại cho các mã như vậy. Cái đó dẫn tới phép tổng quát hoá sau:

Định lý 5: Giả sử m là bội của b, tức là $m = b.m_b$. Nếu tồn tại mã [n, k, d] trên $GF(2^{b(t+1)})$ có độ dài n, kích cỡ k và khoảng cách cực tiểu d, với (t+1)k > n, m>> $\log_2 n$, thế thì tồn tại hàm băm song song dựa trên mã khối m-bit với khoá tm-bit sao cho việc tìm va chạm cho hàm nén đòi hỏi ít nhất $2^{(d-1)m/2}$ phép mã nếu Giả thuyết 2 đúng. Hàm băm có bộ nhớ trong n.m bit và có tỷ lệ (t+1)k/n-1 và làm việc với các khối m_b-bit.

Chứng minh: tương tự như Định lý 3. □

Như một ví dụ, chúng ta có thể kiến thiết hàm băm dựa trên mã khối m-bit với khoá m-bit bằng cách dùng mã [8,6,3] trên GF(2^4), nó nhận được bằng cách chặt ngắn mã Hamming [17, 15, 3]. Hàm băm có tỷ lệ ½ và bộ nhớ trong 8.m bit, nhanh hơn 2 lần so với ví dụ dùng mã [8,5,3] được nói tới ở trên. Với m=64, kiến thiết này thao tác trên các từ 32-bit. Có thể mở rộng phương pháp này để kiến thiết các hàm băm bằng các mã trên GF(2^8), tức là thao tác trên các từ 16-bit, ví dụ bằng cách chặt ngắn mã Hamming [257, 255, 3] (s=2). Tuy nhiên, vì tồn tại mã [17, 15, 3] trên GF(2^4), nên kiến thiết trên GF(2^8) chỉ hiệu quả hơn với n > 17. Sử dụng mã [n,k,d] đòi hỏi n.m bít bộ nhớ trong, điều này làm cho kiến thiết ít hấp dẫn với n lớn. Hơn nữa, các mã nhận được từ việc tách các khối thành các từ nhỏ hơn đem lại các thể hiện phức tạp hơn.

Chú ý:
1. Ngoài chứng minh đơn giản về độ an toàn và tỷ lệ tương đối cao, các lược đồ còn có ưu điểm là n phép mã có thể thực hiện song song.
2. Nhược điểm của các lược đồ là việc tăng bộ nhớ trong và giá của việc cài đặt mã (đó chính là phép exclusive or)
3. Đối với tấn công tìm tạo ảnh, các giới hạn an toàn giả thiết rằng entropy của phần không biết trong đầu vào ít nhất là (d-1)m bit.

Như đối với (3), nếu D < (d-1)m bit là không biết với người tấn công, thì tấn công vét cạn tìm tạo ảnh có thể thực hiện trong khoảng 2^D phép mã.

3.2 Ánh xạ đầu ra

Các kiến thiết được trình bày trên đây có các vấn đề sau:
1. vì mỗi bit đầu ra không phụ thuộc vào tất cả các bit đầu vào của hàm nén, tương đối dễ tìm được nhiều đầu vào sao cho một số khối đầu ra của hàm nén là bằng nhau.
2. số các khối đầu ra thường lớn hơn nhiều so với mức an toàn được đề nghị.

Giải pháp là áp dụng các biến đổi đầu ra vào các đầu ra của hàm nén. Biến đổi này có thể chậm, vì nó chỉ được áp dụng 1 lần. Có nhiều kiến thiết có thể.

Trước hết chúng tôi trình bày phương pháp không ảnh hưởng đến độ an toàn chứng minh được của hàm nén. Người ta mã n khối đầu ra của hàm nén bằng mã khối với khoá cố định, sao cho tất cả các khối đầu ra của phép mã phụ thuộc vào tất cả các khối đầu vào một cách phức tạp. Sau đó, n khối được liên kết cùng với n khối đã mã được băm bằng cách sau. Ký hiệu n_{min} là giá trị nhỏ nhất của n với một giá trị d đã cho sao cho Định lý 5 đúng. Nén 2n khối thành n_{min} khối bằng kiến thiết mới với n_{min} khối song song (nếu n_{min} < n, hàm băm này sẽ có tỷ lệ nhỏ hơn

sơ với hàm ban đầu). Phương pháp này giải quyết vấn đề thứ nhất và một phần vấn đề thứ hai. Tuy nhiên, nếu đòi hỏi phép thu gọn tiếp theo cần phải nhỏ hơn n_{min} khối thì cần đến phương pháp khác.

Chúng tôi trình bày sau đây phương pháp tổng quát cho mọi giá trị n, nó có thể được sử dụng thay thế hoặc đồng thời với phương pháp thứ nhất. Trước hết, người ta kiến thiết từ mã khối m-bit một mã khối mạnh, lớn với độ dài khối n.m-bit. Mã khối này có thể chậm vì nó chỉ được áp dụng 1 lần. Sau đó, n khối (n_{min}) từ hàm nén là đầu vào của kiến thiết Davies-Meyer, còn khoá của mã khối được chọn ngẫu nhiên và cố định (và một phần của mô tả hàm băm). Với giả thuyết 1, đó là hàm băm an toàn. Đầu ra có thể cắt thành s khối bất kỳ, với $s \geq d-1$.

Chúng tôi lấy ví dụ một kiến thiết như vậy. Một phép lặp sẽ gồm 2 bước sau. Trước hết, hoán vị các khối, sao cho khối i trở thành khối i+1 và khối cuối cùng trở thành khối đầu tiên. Thứ hai, mã các khối đầu vào bằng một mã khối nhỏ theo CBC mode. Chính xác hơn, kí hiệu đầu ra của hàm nén là $H^1,...,H^n$. Giả sử K_i với i=1,.., n là n khoá được chọn ngẫu nhiên và cố định, giả sử $C^i = H^i$ với i=1,..,n. Lặp lại thủ tục sau r lần:

$C^0 = C^n$

$C^i = C^{i-1}$ với i =1,..., n

$C^i = E_{K_i}(C^i \oplus C^{i-1})$ với i=1,..,n.

(chúng ta sử dụng chính n khoá đó cho mỗi vòng lặp. Khác đi, các khoá khác nhau có thể sử dụng cho tất cả các vòng. Cũng vậy, phép hoán vị khối trong vòng lặp đầu tiên có thể bỏ qua). Sau một phép lặp, khối cuối cùng sẽ phụ thuộc vào tất cả các khối đầu vào và nói chung, khối i phụ thuộc vào i khối đầu vào. Tuy nhiên, hai vòng là không đủ để có một mã khối mạnh. Cho nên chúng tôi khuyến cáo rằng mã khối được sử dụng có ít nhất r=n vòng. Cuối cùng, kết quả của biến đổi đầu ra được định nghĩa như là phép nối các khối $H^i \oplus C^i$ với i=1,..., n. Nếu đầu vào được cắt ngắn chúng tôi khuyến cáo rằng đầu ra cuối cùng được tạo ra từ các khối có chỉ số cao nhất. Với r vòng, biến đổi đầu ra đòi hỏi r^2 phép mã của mã khối nhỏ.

Trong phần sau chúng tôi sẽ đưa ra một số ví dụ.

4. Một số ví dụ thực tế

Mục này chứa một số ví dụ của kiến thiết mới cho các tham số khác nhau của mã khối bên trong. Chúng ta sẽ sử dụng ký hiệu (m,k) cho mã khối m-bit với khoá k-bit. Trong các ví dụ sẽ đưa ra chúng tôi sử dụng mã Hamming với d=3 và mã MDS với d=4. Sự tồn tại của các mã vừa được nhắc tới có ở [13, Chương 11, Định lý 10].

4.1 Sử dụng mã khối (m, m)

Lược đồ	*Tỷ lệ*	*Va chạm h*	*Va chạm H*	*Tài liệu*
MDC-2	1/2	$2^{m/2}$	2^m	[1]
MDC-4	1/4	$2^{3m/4}$	2^m	[1]
Merkle	0.27	2^m	2^m	[15]

Bảng 1. Tỷ lệ và độ phức tạp của các đề xuất trước đây cho mã khối (m,m)

$GF(2^2)$		$GF(2^4)$		Va chạm
Code	Rate	Code	Rate	
[5,3,3]	1/5	[6,4,3]	1/4	$\geq 2^m$
[8,5,3]	1/4	[8,6,3]	1/2	$\geq 2^m$
[12,9,3]	1/2	[12,10,3]	2/3	$\geq 2^m$
[9,5,4]	1/9	[9,6,4]	1/3	$\geq 2^{3m/2}$
[16,12,4]	1/2	[16,13,4]	5/8	$\geq 2^{3m/2}$

Bảng 2. So sánh các kiến thiết dựa vào mã trên $GF(2^2)$ và $GF(2^4)$ với mã khối (m,m)

Chú ý độ phức tạp 2^m của MDC-2 và MDC-4 là các tấn công tốt nhất biết được chống lại hàm băm, trong khi chống lại lược đồ Merkle và các lược đồ của Bảng 2 thì độ phức tạp 2^m để chống lại các hàm nén và là cận dưới cho độ phức tạp để tấn công hàm băm.

Sau đây, chúng tôi chỉ ra việc thực hành kiến thiết bằng mã [9, 6, 4]. Chúng ta định nghĩa $GF(2^4)$ như là trường mở rộng $GF(2)[x]/(x^4 + x + 1)$. Có nhiều ma trận sinh cho mã tuyến tính [9, 6, 4] trên $GF(2^4)$. Chúng ta sẽ tìm ma trận sinh dẫn đến hàm nén đơn giản và hiệu quả như giải thích sau đây. Ma trận sinh có dạng sau:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & \alpha & \beta \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & \beta & \alpha \\ 0 & 0 & 0 & 1 & 0 & 0 & \alpha & 1 & \beta \\ 0 & 0 & 0 & 0 & 1 & 0 & \alpha & \beta & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & \beta & 1 & \alpha \end{bmatrix} \quad (3)$$

Ở đây, 0 và 1 là các phần tử trung hoà của phép cộng và phép nhân trong $GF(2^4)$, còn $\alpha = x$ và $\beta = x^3 + x^2 + 1$. Ý tưởng để chọn ma trận sinh như sau. Trong cài đặt của hàm nén, các phần tử của $GF(2^4)$ được biểu diễn như các phần tử của không gian vecto trên GF(2). Rõ ràng là các phép nhân với 0 và 1 là dễ thể hiện nhất. Các phân tích cụ thể hơn chỉ ra rằng các phép nhân với α và β trong ví dụ trên có thể thể hiện tương ứng bằng 1 hay 2 phép exclusive-or. Việc vét cạn để tìm ma trận

cùng với cài đặt dễ nhất rõ ràng là không thể, nhưng bằng cách giới hạn bởi việc dùng các phần tử 0, 1, α và β chúng ta nhận được lời giải gần với cái tối ưu.

Giả sử $f_i(X, Y)$ là các phiên bản khác nhau của hàm $E_X(Y)\oplus Y$, giả sử X_L và X_R ký hiệu tương ứng m/2 bít bên trái hoặc bên phải của X và || ký hiệu phép nối 2 khối m/2 bit. Tiếp theo, giả sử G^1, ..., G^9 là 9 khối đầu vào từ hàm nén trong phép lặp trên và giả sử M^1, M^2, M^3 là 3 khối tin đầu vào. Những cái đó đưa lại hàm nén sau:

$$H^1 = f_1(G^1, G^2)$$
$$H^2 = f_2(G^3, G^4)$$
$$H^3 = f_3(G^5, G^6)$$
$$H^4 = f_4(G^7, G^8)$$
$$H^5 = f_5(G^9, G^1)$$
$$H^6 = f_6(M^2, M^3)$$
$$H^7 = f_7(G^1 \oplus G^3 \oplus G^5 \oplus \left(G_R^7 \| G_L^8\right) \oplus \left(G_R^9 \| M_L^1\right) \oplus \left(M_{LR}^3 \oplus M_R^3\right),$$
$$G^2 \oplus G^4 \oplus G^6 \oplus \left(G_L^7 \oplus G_R^8 \| G_L^7\right) \oplus \left(G_L^9 \oplus M_R^1 \| G_L^9\right) \oplus \left(M_L^2 \| M_R^2 \oplus M_{LR}^3\right))$$
$$H^8 = f_8(G^1 \oplus G^7 \oplus M^2 \oplus \left(G_R^3 \| G_L^4\right) \oplus \left(G_{LR}^6 \| G_R^6\right) \oplus \left(M_{LR}^1 \oplus M_R^1\right),$$
$$G^2 \oplus G^8 \oplus M^3 \oplus \left(G_R^4 \oplus G_L^3 \| G_L^3\right) \oplus \Im^5 \oplus \left(G_L^9 \| G_R^9 \oplus M_{LR}^1\right))$$
$$H^8 = f_9(G^1 \oplus G^9 \oplus \left(G_{LR}^4 \| G_R^4\right) \oplus \left(G_R^5 \| G_L^6\right) \oplus \left(G_{LR}^8 \| G_R^8\right) \oplus \left(M_R^2 \oplus M_L^3\right),$$
$$G^2 \oplus M^1 \oplus \Im^3 \oplus \left(G_R^6 \oplus G_L^5 \| G_L^5\right) \oplus \Im^7 \oplus \left(M_L^2 \oplus M_R^3 \| M_L^2\right))$$

với $\Im^t = \left(G_L^t \| G_R^t \oplus G_L^{t+1} \oplus G_R^{t+1}\right)$ và $X_{LR} = X_L \oplus X_R$.

Như là một biến đổi đầu ra chúng tôi đề ngị trước hết băm 9 khối thành 7 khối qua hàm nén sử dụng mã [7,4,4] (n_{min}= 7 cho d=4) và sau đó băm 7 khối thành 3 khối nhờ biến đổi đầu ra có 7 vòng như mô tả ở phần trên.

4.2 Sử dụng mã khối (m, 2m)

Các hàm băm duy nhất đã biết dựa trên mã khối (m, 2m) với kết quả băm 2m-bit chính là Abreast-DM và Tandem-DM từ [12]. Các hàm nén từ cả hai hàm đó xử lý khối m-bit bằng cách dùng 2 phép mã với tỷ lệ ½. Giả sử H_i và G_i ký hiệu các giá trị băm trung gian. Thế thì Abreast-DM được định nghĩa như sau:

$$H_i = H_{i-1} \oplus E_{G_{i-1}, M_i}(H_{i-1})$$
$$G_i = G_{i-1} \oplus E_{M_i, H_{i-1}}(\overline{G}_{i-1})$$

với $\overline{G}$ là phép bù theo bit của G. Còn Tandem-DM được định nghĩa như sau:

$$W_i = E_{G_{i-1}, M_i}(H_{i-1})$$
$$H_i = W_i \oplus H_{i-1}$$
$$G_i = G_{i-1} \oplus E_{M_i, W_i}(G_{i-1})$$

Bảng 3 liệt kê các tỷ lệ và độ phức tạp của các tấn công tốt nhất đã biết đối với 2 kiến thiết. Tuy nhiên, như đã chỉ ra trên đây, tồn tại các kiến thiết hiệu quả hơn cùng với độ an toàn tốt hơn. Bảng 4 liệt kê tỷ lệ và độ phức tạp của các kiến thiết như vậy. Cũng như trước đây, có thể chia các khối m-bit thành các khối con nhỏ hơn. Ví dụ, các khối có thể chia ra làm 2 nửa và mở rộng bằng mã trên GF(2^6), ví dụ như [65, 63, 3].

Lược đồ	Tỷ lệ	Va chạm h	Va chạm H
Abrest-DM	1/2	2^m	2^m
Tandem-DM	1/2	2^m	2^m

Bảng 3. Tỷ lệ và độ phức tạp của các đề xuất
trước đó cho mã khối (m, 2m) [12]

Mã	Tỷ lệ	Va chạm
[4, 2, 3]	½	$\geq 2^m$
[6, 4, 3]	1	$\geq 2^m$
[9, 7, 3]	4/3	$\geq 2^m$
[5, 2, 4]	1/5	$\geq 2^{3m/2}$
[7, 4, 4]	5/7	$\geq 2^{3m/2}$
[10, 7, 4]	11/10	$\geq 2^{3m/2}$

Bảng 4. Tỷ lệ và độ phức tạp cho các đề xuất của chúng tôi
đối với mã khối (m, 2m) dùng mã trên GF(2^3).

4.3 Họ MDx

Cả MD4 [21] và MD5 [22] có thể được xem như là kiết thiết Davies-Meyer với mã khối m-bit bên trong nhưng có khoá 4m-bit. Từ quan điểm này, cả hai kiến thiết có tỷ lệ 4.

Tuy nhiên, các tấn công của Dobbertin [4, 6] đối với MD4 và MD5 chỉ ra rằng các hàm nén là không phải không va chạm, tấn công của ông lên MD4 mở rộng [5] chỉ ra rằng đối với MD4 ngay cả hai lần chạy độc lập của hàm nén cũng không phải là không va chạm. Tuy nhiên, dường như là các tấn công của Dobbertin không chuyển sang được hàm nén chứa 2 hay nhiều hơn các phiên bản của MD5. Chúng ta có thể áp dụng phương pháp ở đây để kiến thiết các hàm MD5 song song dựa vào các mã tuyến tính trên GF(2^5). Trong bảng 5 sẽ liệt kê các kiến thiết có thể.

Scheme	Rate	Scheme	Rate	Scheme	Rate
MD4	4	[5,3,3]	2	[5,2,4]	1
MD5	4	[10,8,3]	3	[10,7,4]	2.5

		[20, 18, 3]	3.5	[20,17,4]	3.25

Bảng 5. Tỷ lệ và độ phức tạp của các đề xuất mới cho họ MDx sử dụng mã trên GF(2^5)

Vì các giả thuyết cho các kiến thiết của chúng ta, tức là các thành phần cơ bản là an toàn, là không đúng cho MD4 và MD5, chúng ta không thể chỉ ra chính xác các cận cho độ phức tạp của các tấn công va chạm đối với các hàm nén. Tuy nhiên, chúng ta giả thuyết rằng đối với các kiến thiết dựa trên MD5 và các mã có khoảng cách cực tiểu bằng 4 thì tấn công va chạm là không thể. Tấn công đòi hỏi va chạm đồng thời cho ít nhất 3 phiên bản khác nhau với các đầu vào độc lập.

5. Kết luận

Chúng tôi đã trình bày việc tìm va chạm cho hàm nén của MDC-4 cần 2^{41} phép tính nếu sử dụng DES. Điều này tạo nên một mối hoài nghi về độ an toàn của MDC-4. Chúng tôi cũng trình bày phương pháp mới để kiến thiết hàm băm dựa trên mã khối như DES nhưng nhanh hơn và an toàn hơn so với các đề xuất trước đây. Cũng vậy, phương pháp của chúng tôi mở rộng ra cho các mã khối như IDEA, nó có độ dài khoá và độ dài khối khác nhau. Với một dung lượng lớn bộ nhớ trong, các kiến thiết dùng IDEA tồn tại với tỷ lệ gần với 2, cái đó nhanh hơn 4 lần so với đề xuất hiện nay. Cuối cùng, chúng tôi bàn luận về việc áp dụng các kết quả của chúng ta vào họ MDx. Chúng tôi chỉ ra rằng các kiến thiết dùng MD5, sẽ nhanh gần như MD5, nhưng (giả thuyết) sẽ an toàn hơn nhiều so với MD5.

Tài liệu

1. B. O. Brachtl, D. Coppersmith, M.M. Hyden, S.M. Matyas, C.H. Meyer, J. Oseas, S. Pilpel, M. Schilling, "*Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function*, " U.S. Patent Number 4.908.861, March 13, 1990
2. I.B. Damgard, "A design principle for hash functions," *Advances in Cryptology, Proc. Crypto'89*, LNCS 435, G. Brassard, Ed., Springer-Verlag, 1990, pp.416-427.
3. B. den Boer, A. Bosselaers, "*Collisions for the compression function of MD5*," Advances in Cryptology, Proc. Eurocrypt'93, LNCS 765, T. Helleseth, Ed., Springer-Verlag, 1994, pp.293-304.
4. H. Dobbertin, "Cryptanalysis of MD4," *Fast Software Encryption*, LNCS 1039, D. Gollman, Ed., Springer-Verlag, 1996, pp. 53-69.
5. H. Dobbertin, "Extended MD4 Compression is not Collision-free," Unbublished, October 1995.
6. H. Dobbertion, "Cryptanalysis of MD5 compression," *Presented at rump session of Eurocrypt'96*, May 1996.
7. FIPS 46, "Data Encryption Standard," Federal Information Processing Standard (FIPS), Publication 46, National Bureau of Standard, U.S. Department of Commerce, Washington D.C., January 1997.

8. W. Hohl, X. Lai, T. Meier, C. Waldvogel, "Security of iterated hash functions based on block ciphers," *Advances in Cryptology, Proc. Crypto'93*, LNCS 773, D. Stinson, Ed., Springer-Verlag, 1994, pp.379-390.
9. ISO/IEC 10118, "*Information technology-Security techniques-Hash-functions. Part 1: General and Part 2: Hash-functions using an n-bit block cipher algorithm*," 1994.
10. L.R. Knudsen, X. Lai, "*New attacks on all double block lenth hash functions of hash rate 1, including the parallel-DM*," Advances in Cryptology, Proc. Eurocrypt'94, LNCS 959, A. De Santis, Ed., Springer-Verlag, 1995, pp.410-418.
11. L.R. Knudsen, B. Preneel, "Hash functions based on block ciphers and quaternary codes," *Advances in Cryptology, Proc. Asiacrypt'96*, LNCS 1163, K. Kim, T. Matsumoto, Eds., Springer-Verlag, 1996, pp.77-90.
12. X. Lai, "*On the Design and Security of Block Ciphers,*" ETH Series in Information Processing, Vol. 1, J.L. massey, Ed., Hartung-Gorre Verlag, Konstanz, 1992.
13. F.J. MacWilliams, N.J.A. Sloane, "*The Theory of Error-Correcting Codes,*" North-Holland Publishing Company, Amsterdam, 1978.
14. S.M. Matyas, C.H. Meyer, J. Oseas, "*Generating strong one-way functions with cryptographic algorithm*," IBM Techn. Disclosure Bull., Vol. 27, No. 10A, 1985, pp. 5658-5659.
15. R. Merkle, "One way hash functions and DES," *Advances in Cryptology, Proc. Crypto'89*, LNCS 435, G. Brassard, Ed., Springer-Verlag, 1990, pp.428-446.
16. C.H. Meyer, M.Schilling, "Secure program load with Manipulation Detection Code," *Proc. Securicom 1988*, pp.111-130.
17. M. Naor, M. Yung, "Universal one-way hash functions and their cryptographic applications," *Proc. 21st ACM Symposium on the Theory of Computing*, ACM, 1989, pp.387-394.
18. B. Preneel, R. Govaerts, J. Vandewalle, "Hash functions based on block ciphers: a synthesic approach," *Advances in Cryptology, Proc. Crypto'93,* LNCS 773, D. Stinson, Ed., Springer-Verlag, 1994, pp.368-378.
19. V. Rijmen, B. Preneel, "Improved characteristics for differential cryptanalysis of hash functions based on block ciphers," *Fast Spftware Encryption, LNCS 1008*, B. Preneel, Ed., Springer-Verlag, 1995, pp.242-248.
20. J.-J. Quisquater, J.-P. Delescaille, "How easy is collision search? Application to DES," *Advances in Cryptology, Proc. Eurocrypt'89, LNCS 43*4, J.-J. Quisquater, J. Vandewalle, Eds., Springer-Verlag, 1990, pp.429-434.
21. R.L. Rivest, "The MD4 message digest algorithm," Advances in Cryptology, Pro. Crypto'90, LNCS 537, S. Vanstone, Ed., Springer-Verlag, 1991, pp.303-311.
22. R. L. Rivest, "The MD5 message-digest algorithm," *Request for Comments (RFC) 1321*, Internet Activities Board, Internet Privacy Task Force, April 1992.

23. P.C an Oorschot, M.J. Wierner, “Parallel collision search with application to hash functions and discrete logarithms,” *Proc. 2nd ACM Conference on Computer and Communications Security*, ACM, 1994, pp.210-218.

Độ mật của hàm băm lặp dựa trên mã khối

Walter Hohl, Xuejia Lai, Thomas Meier, Christian Waldvogel
Crypto 93, pp. 379-390

Tóm tắt: Các hàm băm mật mã có được bằng cách lặp các hàm vòng (round function), các hàm vòng được kiến thiết từ mã khối và trường hợp độ dài của giá trị băm bằng 2 lần độ dài khối m của mã khối bên trong được xem xét. Độ an toàn tính toán của các hàm băm như vậy chống lại 2 tấn công cụ thể, là tấn công đích có khởi đầu tự do và tấn công va chạm có khởi đầu tự do được khảo sát; hai tấn công này là khác so với tấn công đích và tấn công va chạm "thông thường" ở chỗ giá trị khởi đầu của phép lặp không được chỉ ra. Lý do là từ việc các hàm băm lặp an toàn tính toán chống lại hai tấn công này suy chúng an toàn tính toán chống lại các tấn công đích và tấn công va chạm thông thường. Cho một lớp hàm băm lặp 2m-bit tổng quát, một cận trên chặt hơn so với những gì đã được công bố trong tài liệu về độ phức tạp của tấn công đích có khởi đầu tự do và tấn công va chạm có khởi đầu tự do được đưa ra. Một đề xuất hàm băm lặp 2m-bit đạt được các cận trên này được đưa ra; đề xuất mới này được chỉ ra là an toàn hơn về mặt tính toán chống lại tấn công đích có khởi đầu tự do và tấn công va chạm có khởi đầu tự do hơn so với một số lược đồ đã được đề xuất rơi vào lớp tổng quát này. Cũng chỉ ra rằng đề xuất của chúng tôi là tốt hơn đề xuất hiện tại cho chuẩn ISO theo nghĩa cả hai lược đồ đạt được các cận trên này, nhưng chỉ một phép mã cần thiết trong đề xuất của chúng tôi để băm một khối tin m-bit so với hai phép mã trong đề xuất cho chuẩn ISO. Cuối cùng, hai tấn công mới đối với hàm băm-khối-đúp LOKI (LOKI Double-Block-Hash) đã được trình bày với độ phức tạp thấp hơn so với những gì đã biết.

1. Mở đầu

Hàm băm là một ánh xạ dễ tính từ tập tất cả các dãy nhị phân với độ dài tối thiểu nào đó hoặc lớn hơn vào tập các dãy nhị phân có độ dài cố định nào đó. Trong mật mã, các hàm băm được sử dụng để cung cấp tính toàn vẹn dữ liệu và để tạo ra các chữ ký số ngắn.

Một phương pháp quen biết để nhận được hàm băm là dùng phép lặp. Giả sử H_0 ký hiệu giá trị khởi đầu m-bit ($m > 0$) và giả sử M ký hiệu bản tin nhị phân có độ dài là bội dương của k, tức là $M = (M_1, M_2,...,M_n)$ với n dương nào đó còn M_i là các khối m-bit. (Chú ý rằng người ta kéo dài độ dài bản tin thành bội của k bằng cách áp dụng các kỹ thuật padding xác định, xem [ISO 91, Merkle 90]). Khi đó, chúng ta viết *hash*(.,.) để ký hiệu hàm băm lặp m-bit, khi cho H_0 và M nó tính giá trị băm $hash(H_0,M)=H_n$ theo đẳng thức đệ qui sau:

$$H_i = \text{round}(H_{i-1}, M_i) \quad i=1, 2,..., n \qquad (1)$$

với round(.,.) là hàm dễ tính từ 2 đầu vào m và k bit, còn đầu ra m-bit. Hàm round(.,.) được gọi là hàm vòng m-bit.

Giả sử H_0 và $\hat{H}_0$ là 2 giá trị khởi điểm m-bit và giả sử M = (M_1, ..., M_n) và $M = \left(\hat{M}_1,...,\hat{M}_{\hat{n}}\right)$ là 2 bản tin nhị phân với M_i ($1 \le i \le n$) và $\hat{M}_i$, $(1 \le i \le \hat{n})$ là các khối k-bit. Với hàm băm lặp hash(.,.) chúng ta có thể phân biệt 5 dạng tấn công sau (xem [Lai 92]):

- **tấn công đích**: Cho H_0 và M, tìm $\hat{M}$ sao cho $\hat{M} \neq M$ nhưng $hash(H_0,\hat{M}) = hash(H_0,M)$
- **tấn công đích có khởi đầu tự do**: Cho H_0 và M, tìm $\hat{H}_0$ và $\hat{M}$ sao cho $(\hat{H}_0,\hat{M}) \neq (H_0,M)$ nhưng $hash(\hat{H}_0,\hat{M}) = hash(H_0,M)$
- **tấn công va chạm:** Cho H_0, tìm M và $\hat{M}$ sao cho $\hat{M} \neq M$ nhưng $hash(H_0,\hat{M}) = hash(H_0,M)$
- **tấn công va chạm có khởi đầu nửa tự do**: Tìm H_0, M và $\hat{M}$ sao cho $\hat{M} \neq M$ nhưng $hash(H_0,\hat{M}) = hash(H_0,M)$
- **tấn công va chạm có khởi đầu tự do**: Tìm H_0, M, $\hat{H}_0$ và $\hat{M}$ sao cho $(\hat{H}_0,\hat{M}) \neq (H_0,M)$ nhưng $hash(\hat{H}_0,\hat{M}) = hash(H_0,M)$.

- Khi các bản tin M và $\hat{M}$ chỉ chứa một khối, tức là $n = \hat{n} = 1$, chúng ta có hash(H_0,M)=round(H_0,M), từ đó suy ra rằng mỗi một trong số các tấn công trên được đưa về một tấn công cùng loại đối với hàm vòng m-bit. Ví dụ, tấn công đích sẽ như sau: cho H_0 và M, tìm $\hat{M}$ sao cho $\hat{M} \neq M$ nhưng $round(H_0,\hat{M}) = round(H_0,M)$.

Chúng ta sẽ xem xét các hàm băm lặp dựa trên các mã khối (m,k), khi mà mã khối (m,k), với mỗi khoá k-bit, là một ánh xạ có ngược từ tập tất cả các bản rõ m-bit lên tập tất cả các bản mã m-bit. Khi cho mã khối (m,k), chúng ta viết $E_Z(X)$ để ký hiệu phép mã m-bit rõ X với khoá Z k-bit, và $D_Z(Y)$ ký hiệu phép giải mã bản mã Y m-bit với khoá Z k-bit. Trong lập luận của chúng ta, luôn giả thiết rằng mã khối (m,k) không có điểm yếu. Chúng ta định nghĩa tỷ lệ (rate) của hàm băm lặp (hay tương đương, của hàm vòng) như là số các khối m-bit được xử lý theo mỗi phép mã hoặc giải mã.

Khi cho hàm vòng m-bit dựa trên mã khối (m,k), chúng ta định nghĩa độ phức tạp của tấn công như là tổmg số phép mã hoặc giải mã của mã khối (m,k) cần thiết cho việc tấn công, tức là nếu phép tấn công cần 2^s phép mã hay giải mã thì ta nói nó có độ phức tạp 2^s. Bởi vì phép tấn công trên hàm vòng m-bit kéo theo phép tấn công cùng kiểu trên hàm băm lặp m-bit với cùng độ phức tạp, nên việc thiết kế các hàm vòng an toàn tính toán là điều kiện cần (nhưng không đủ) để thiết kế các hàm băm lặp an toàn tính toán. Hơn nữa, dưới các điều kiện nào đó (xem [Merkle 90, Damgard 90, Naor 89, Lai 92]), hàm vòng an toàn tính toán kéo theo hàm băm lặp an toàn tính toán. Chúng ta sẽ tập trung sự chú ý vào việc thiết kế các hàm vòng an toàn tính toán.

Trong phần 2 chúng ta sẽ xem xét một lớp tổng quát các hàm băm lặp 2m-bit với tỷ lệ 1 dựa trên mã khối (m,m). Một số lược đồ được đề xuất trước đây [Preneel 89, Quisquater 89, Brown 90] được chỉ ra là ở trong lớp này. Cho lớp này, chúng ta nhận được các cận trên về độ phức tạp của các tấn công đích có khởi đầu tự do và tấn công va chạm cơ khởi đầu tự do, bằng cách mô tả các tấn công tốt hơn tấn công vét cạn. Trong phần 3, chúng tôi đề xuất một hàm băm lặp 2m-bit, đối với nó sẽ chứng minh rằng với những giả thiết chấp nhận được, nó đạt được các cận trên. Phần 4 chứa một tấn công va chạm có khởi đầu tự do mới sử dụng 2 phép mã đối với lược đồ LOKI DBH [Brown 90] và tấn công va chạm có khởi đầu nửa tự do đòi hỏi $2^{m/2}$ phép mã. Trong phần 5 chúng ta sẽ khảo sát lớp các hàm băm lặp 2m-bit với tỷ lệ 1/2. Cũng chỉ ra rằng các cận trên đạt được trong phần 2 cũng đúng cho lớp các hàm băm có tỷ lệ 1/2. Điều đó chỉ ra rằng cả đề xuất của chúng tôi, cả lược đồ Meyer-Schilling [Meyer 88] (nó hiện đang được xem xét như chuẩn ISO [ISO 91]) đạt được cùng một độ an toàn tính toán chống lại các tấn công có khởi đầu tự do, tuy nhiên, đề xuất của chúng tôi là hiệu quả hơn theo nghĩa chỉ một phép mã được yêu cầu để băm một khối tin m-bit trong khi 2 phép mã cần trong lược đồ Meyer-Schilling.

2. Các hàm vòng 2m-bit với tỷ lệ 1

Trong mục này, chúng ta xem xét các hàm vòng 2m-bit với tỷ lệ 1 dựa trên các mã khối (m, m), tức là các mã khối với khối rõ-mã m-bit và khoá m-bit. Giả sử các giá trị băm 2m-bit H_i được viết như là việc ghép nối (được ký hiệu bằng ký tự :) hai vecto m-bit H_i^1 và H_i^2 sao cho $H_i = H_i^1 : H_i^2$, tương tự, giả sử khối tin 2m-bit M_i được viết như là $M_i = M_i^1 : M_i^2$ với M_i^1 và M_i^2 là 2 vecto m-bit. Sử dụng ký hiệu mới này, chúng ta có thể viết lại (1) như sau:

$$H_i^1 : H_i^2 = round(H_{i-1}^1 : H_{i-1}^2, M_i^1 : M_i^2) \qquad (2)$$

Một trong những đề xuất một hàm băm lặp 2m-bit dựa trên hàm vòng 2m-bit với tỷ lệ 1 được định nghĩa trong (2) là như sau:

Lược đồ LOKI Double Block Hash (DBH): Với hàm băm lặp 2m-bit được đề xuất trong [Brown 90], hàm vòng 2m-bit được cho bởi

$$\begin{cases} H_i^1 = E_{H_{i-1}^1 \oplus M_i^1}\left(H_{i-1}^1 \oplus M_i^2\right) \oplus H_{i-1}^1 \oplus H_{i-1}^2 \oplus M_i^2 \\ H_i^2 = E_{H_{i-1}^2 \oplus M_i^2}\left(H_{i-1}^1 \oplus M_i^1 \oplus H_i^1\right) \oplus H_{i-1}^1 \oplus H_{i-1}^2 \oplus M_i^1 \end{cases} \qquad (3)$$

với $\oplus$ ký hiệu phép cộng từng bit theo modulo 2.

Các đề xuất tương tự khác là lược đồ Preneel-Bosselaers-Govaerts-Vandewalle (PBGV) được đưa ra trong [Preneel 89] và lược đồ Quisquater-Gilraut (QG) được đưa ra trong [Quisquater 89]. Mục đích của các lược đồ này là để nhận được các hàm vòng 2m-bit an toàn bằng cách sửa đổi hàm vòng m-bit đã được coi là an toàn được đưa ra bởi Davies và Meyer [Davies 85, Matyas 85, Winternitz 84]. Tuy

nhiên, một số công trình gần đây (xem [Quisquater 89, Miyaguchi 91, Lai 92, Preneel 93]) và các tấn công đối với lược đồ LOKI-DBH được trình bày trong bài báo này sẽ chỉ ra rằng các đề xuất cho hàm vòng 2m-bit như vậy về thực chất là yếu hơn so với hàm vòng m-bit bên trong đối với các tấn công có khởi đầu tự do. Để đạt được một lời giải có hệ thống cho bài toán này, trước hết chúng ta xem xét dạng tổng quát sau của các hàm vòng 2m-bit.

Dạng tổng quát của hàm vòng 2m-bit có tỷ lệ 1:

$$\begin{cases} H_i^1 = E_A(B) \oplus C \\ H_i^2 = E_R(S) \oplus T \end{cases} \qquad (4)$$

với A, B và C là các tổ hợp tuyến tính nhị phân của các vecto m-bit $H_{i-1}^1, H_{i-1}^2, M_i^1$ và M_i^2, còn R, S và T là các tổ hợp nào đó (không nhất thiết phải tuyến tính nhị phân) của các vecto $H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2$ và H_i^1. Chúng ta có thể viết A, B và C ở dạng ma trận như sau:

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \end{bmatrix} \begin{bmatrix} H_{i-1}^1 \\ H_{i-1}^2 \\ M_i^1 \\ M_i^2 \end{bmatrix} \qquad (5)$$

với các giá trị nhị phân a_i, b_i và c_i nào đó ($1 \le i \le 4$).

Chúng ta có thể dễ dàng thấy rằng với lược đồ LOKI-DBH ta có

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} H_{i-1}^1 \\ H_{i-1}^2 \\ M_i^1 \\ M_i^2 \end{bmatrix} \text{ và } \begin{bmatrix} R \\ S \\ T \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} H_{i-1}^1 \\ H_{i-1}^2 \\ M_i^1 \\ M_i^2 \\ H_i^1 \end{bmatrix}$$

Các lược đồ PBGV và QG cũng có thể được biểu diễn một cách tương tự.

Bây giờ chúng tôi chỉ ra các cận trên về độ phức tạp của các tấn công đích có khởi đầu tự do và tấn công va chạm có khởi đầu tự do trên các hàm băm lặp 2m-bit với hàm vòng có dạng (4).

Khẳng định 1: Với hàm băm lặp 2m-bit có tỷ lệ 1 cùng với hàm vòng 2m-bit có dạng (4), độ phức tạp của tấn công đích có khởi đầu tự do có cận trên là 2^m phép mã, còn tấn công va chạm với khởi đầu tự do có độ phức tạp bị chặn trên khoảng $2^{m/2}$

Chứng minh: Vì có tất cả 2^{4m} giá trị có thể cho vecto 4m-bit

$(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$ và 2^{3m} giá trị có thể cho vecto 3m-bit (A,B,C), cho nên với mỗi giá trị (A,B,C) sẽ có 2^m giá trị $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$ thoả mãn (5). Bài toán tìm 2^m giá trị đó là không đáng kể về tính toán vì từ (5) chúng ta thấy rằng bài toán này tương đương với việc giải hệ 3 phương trình tuyến tính với 4 ẩn số.

Tấn công đích có khởi đầu tự do: Với giá trị đã cho của $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$, chúng ta cần tìm một giá trị như thế nữa sao cho có cùng giá trị (H_i^1, H_i^2) như ở (4). Chúng ta xử lý như sau: Với giá trị đã cho của $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$ chúng ta tính giá trị của (A, B ,C) như (5) và sử dụng lập luận ở trên để tìm ra 2^m giá trị khác của $(\hat{H}_{i-1}^1, \hat{H}_{i-1}^2, \hat{M}_i^1, \hat{M}_i^2)$ sao cho dẫn đến cùng giá trị của (A,B,C). Sau đó ta tính giá trị của H_i^2 cho giá trị $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$ đã cho và tính $\hat{H}_i^2$ cho 2^m giá trị $(\hat{H}_{i-1}^1, \hat{H}_{i-1}^2, \hat{M}_i^1, \hat{M}_i^2)$ vừa nói tới ở trên. Do có 2^m giá trị có thể của $\hat{H}_i^2$ nên với xác xuất 0.63 sẽ tìm được một bộ trong số 2^m giá trị của $(\hat{H}_{i-1}^1, \hat{H}_{i-1}^2, \hat{M}_i^1, \hat{M}_i^2)$ sao cho nó cho $\hat{H}_i^2$ trùng với H_i^2 ứng với $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$ đã cho. Phép tấn công như vậy cần 2^m phép mã, điều nay đem lại cận trên khoảng 2^m cho độ phức tạp của tấn công đích có khởi đầu tự do đối với hàm băm lặp 2m-bit.

Tấn công va chạm có khởi đầu tự do: Chúng ta sẽ tìm hai giá trị khác nhau của $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$ dẫn đến cùng giá trị (H_i^1, H_i^2) theo như (4). Ta xử lý như sau: trước hết ta sinh ra $2^{m/2}$ giá trị cho $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$ sao cho có cùng giá trị H_i^1. Vì có 2^m giá trị có thể cho khối m-bit H_i^2 cho nên theo "lập luận ngày sinh" thông thường với xác suất 0.63 sẽ tìm được 2 giá trị $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$ cho cùng giá trị H_i^2. Phép tấn công này cần $2^{m/2}$ phép mã, nó cho cận trên của độ phức tạp đối với tấn công va chạm có khởi đầu tự do trên hàm băm lặp 2m-bit.

Chú ý. Tư tưởng chính của tấn công trong Khẳng định 1 là tấn công hai đẳng thức trong (1) một cách riêng biệt. Nếu ai đó có thể tìm được nhiều giá trị $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$ dẫn đến cùng một giá trị (A,B,C), thì tấn công đối với hàm vòng 2m-bit dạng (4) dẫn đến tấn công một hàm vòng m-bit. Cho nên, các tấn công tương tự như những tấn công đã được mô tả trong chứng minh của Khẳng định 1 cũng sẽ làm việc, ngay cả khi ánh xạ từ $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$ vào (A,B,C) trong (4) không phải là tổ hợp tuyến tính nhị phân. Cho nên, Khẳng định 1 vẫn đúng nếu dễ tìm được 2^m giá trị khác nhau của $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$ cho cùng một giá trị của (A,B,C).

3. Đề xuất cho hàm băm 2m-bit với tỷ lệ 1

Trong mục này, chúng tôi đề xuất hàm băm lặp 2m-bit mới với hàm vòng 2m-bit dạng (4). Chúng ta sẽ chứng minh rằng đề xuất mới này là tối ưu chống lại các tấn công đích có khởi điểm tự do và tấn công va chạm có khởi đầu tự do.

Trước khi đưa ra đề xuất mới, chúng tôi mô tả hàm băm lặp m-bit đã được đề xuất một cách độc lập bởi Davies và Meyer, xem [Davies 85, Matyas 85, Winternitz 84].

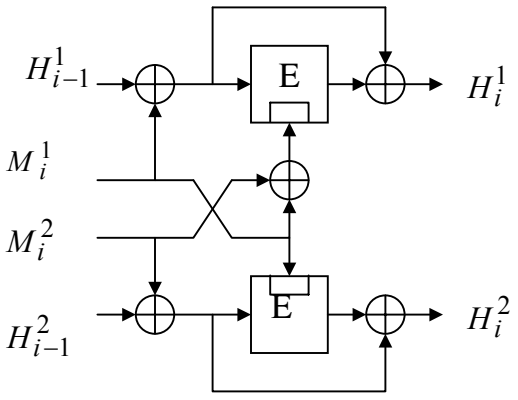
Lược đồ Davies-Meyer (DM): Lược đồ này chứa hàm băm lặp m-bit được định nghĩa như trong (1) với hàm vòng m-bit dựa trên mã khối (m, k). Với mục đích của mình, chúng tôi sẽ giả thiết rằng k=m, tức là độ dài rõ-mã và độ dài khoá bằng nhau. Giả sử H_{i-1} và M_i ký hiệu 2 khối m-bit, đầu ra m-bit H_i của hàm vòng DM đối với cặp đầu vào (H_{i-1}, M_i) được cho bởi

$$H_i = E_{M_i}(H_{i-1}) \oplus H_{i-1} \qquad (6)$$

Lược đồ DM nói chúng được xem là an toàn, tức là các tấn công đích có khởi đầu tự do và tấn công va chạm có khởi đầu tự do đối với (6) cần tương ứng khoảng 2^m và $2^{m/2}$ phép mã. Đề xuất của chúng tôi dựa vào lược đồ DM.

Lược đồ Davies-Meyer song song (Parallel-DM): Với hàm băm lặp 2m-bit được định nghĩa như ở (1), Chúng tôi đề nghị hàm vòng 2m-bit với tỷ lệ 1 dựa trên mã khối (m,m) như sau:

$$\begin{cases} H_i^1 = E_{M_i^1 \oplus M_i^2}(H_{i-1}^1 \oplus M_i^1) \oplus H_{i-1}^1 \oplus M_i^1 \\ H_i^2 = E_{M_i^1}(H_{i-1}^2 \oplus M_i^1) \oplus H_{i-1}^2 \oplus M_i^2 \end{cases} \qquad (7)$$



Hình 1. Hàm vòng 2m-bit của lược đồ đề xuất Parallel-DM
(các hộp nhỏ biểu diễn đầu vào khoá của mã khối

Để tránh các tấn công tầm thường dựa trên sự việc là bản tin giả mạo có thể có độ dài khác với bản tin ban đầu, chúng tôi định nghĩa củng cố sau đối với hàm băm

lặp, chúng được đề nghị một cách độc lập bởi Merkle và Damgard, xem [Merkle 90, Damgard 90].

Củng cố Merkle-Damgard (MD): Với hàm băm lặp được định nghĩa trong (1), củng cố MD chính là việc định ra khối cuối cùng M_n của bản tin nhị phân được băm $M=(M_1,...,M_n)$ chính là độ dài của M (ở dạng nhị phân).

Khẳng định 2: Giả thiết rằng lược đồ DM là an toàn và hàm băm lặp 2m-bit được sử dụng cùng với củng cố MD, từ đó suy ra rằng lược đồ Parallel-DM là tối ưu chống lại tấn công đích có khởi đầu tự do và tấn công va chạm có khởi đầu tự do.

Chứng minh: Bằng cách áp dụng biến đổi có ngược

$$(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2) \to (h_{i-1}^1, h_{i-1}^2, m_i^1, m_i^2)$$

sao cho $\begin{matrix} h_{i-1}^1 = H_{i-1}^1 \oplus M_i^1 & h_{i-1}^2 = H_{i-1}^2 \oplus M_i^2 \\ m_i^1 = M_i^1 & m_i^2 = M_i^1 \oplus M_i^2 \end{matrix}$

đối với hàm vòng Parallel-DM được định nghĩa trong (7), chúng ta nhận được hàm vòng mới 2m-bit

$$\begin{cases} h_i^1 = E_{m_i^2}(h_{i-1}^1) \oplus h_{i-1}^1 \\ h_i^2 = E_{m_i^1}(h_{i-1}^2) \oplus h_{i-1}^2 \end{cases} \qquad (8)$$

.....

Chú ý: Từ (7) làm sao nhận được (8) ???

4. Các tấn công mới đối với lược đồ LOKI-DBH

Chúng ta mô tả tấn công va chạm có khởi đầu tự do mới đối với lược đồ LOKI-DBH, phép tấn công chỉ yêu cầu 2 phép mã và tấn công va chạm nửa khởi đầu tự do, nó sử dụng $2^{m/2}$ phép mã. Các tấn công này có thể áp dụng vào lược đồ LOKI-DBH với một mã khối (m,m) bất kỳ nằm bên trong. Độ phức tạp tấn công thấp như vậy đối với lược đồ LOKI-DBH chưa được công bố trong các tài liệu. Hơn nữa, độ phức tạp thấp của tấn công va chạm có khởi đầu tự do mới này chỉ ra rằng lược đồ LOKI-DBH là không tối ưu chống lại tấn công va chạm khởi đầu tự do.

Bằng cách áp dụng ánh xạ có ngược $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2) \to (h_{i-1}^1, h_{i-1}^2, m_i^1, m_i^2)$ với

$$\begin{matrix} h_{i-1}^1 = H_{i-1}^1, & h_{i-1}^2 = H_{i-1}^2 \\ m_i^1 = M_i^1 \oplus H_{i-1}^1, & m_i^2 = M_i^2 \oplus H_{i-1}^2 \end{matrix} \qquad (9)$$

vào hàm vòng LOKI-DBH được định nghĩa trong (3), chúng ta nhận được hàm vòng mới 2m-bit như sau:

$$\begin{cases} h_i^1 = E_{m_i^1}(h_{i-1}^1 \oplus h_{i-1}^2 \oplus m_i^2) \oplus h_{i-1}^1 \oplus m_i^2 \\ h_i^2 = E_{m_i^2}(m_i^1 \oplus h_i^1) \oplus h_{i-1}^2 \oplus m_i^1 \end{cases} \quad (10)$$

Free-start collision attack: Chúng ta sẽ tìm hai giá trị khác nhau $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$ mà cùng dẫn đến một giá trị (H_i^1, H_i^2) theo như (3). Ta xử lý như sau:

Bước 1: Chọn ngẫu nhiên giá trị x m-bit và hai giá trị khác nhau m-bit là y và $\hat{y}$.

Bước 2: Tính z và $\hat{z}$ như sau:

$$z = E_y(x \oplus y) \oplus x \oplus y \quad (11)$$

$$\hat{z} = E_{\hat{y}}(x \oplus \hat{y}) \oplus x \oplus \hat{y} \quad (12)$$

chúng ta nhận được 2 giá trị khác nhau (z, x⊕ z, y, y) và $(\hat{z}, x \oplus \hat{z}, \hat{y}, \hat{y})$ cho $(h_{i-1}^1, h_{i-1}^2, m_i^1, m_i^2)$.

Bước 3: Bằng cách áp dụng ánh xạ ngược của (9), chúng ta có hai giá trị khác nhau (z, x⊕z, y⊕z, x⊕y⊕z) và $(\hat{z}, x \oplus \hat{z}, \hat{y} \oplus \hat{z}, x \oplus \hat{y} \oplus \hat{z})$ cho $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$.

Chú ý rằng khi thay (z, x⊕ z, y, y) cho $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$ trong (3) sẽ cho

$$\begin{cases} H_i^1 = E_y(x \oplus y) \oplus y \oplus z \\ H_i^2 = E_y(y \oplus H_i^1) \oplus x \oplus y \oplus z \end{cases}$$

sử dụng biểu thức của z được định nghĩa trong (11) cho

$$\begin{cases} H_i^1 = x \\ H_i^2 = E_y(y \oplus H_i^1) \oplus E_y(x \oplus y) \end{cases}$$

Thay H_i^1 bởi x trong vế phải của đẳng thức thứ hai ta có $(H_i^1, H_i^2) = (x,0)$. Cũng bằng cách như vậy, có thể chỉ ra rằng bằng cách thay $(\hat{z}, x \oplus \hat{z}, \hat{y} \oplus \hat{z}, x \oplus \hat{y} \oplus \hat{z})$ cho $(H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2)$ trong (3) cũng dẫn đến $(H_i^1, H_i^2) = (x,0)$, điều đó chứng minh tính đúng đắn phép tấn công của chúng ta. Bởi vì phép tấn công vào hàm vòng suy ra phép tấn công cùng loại về hàm băm lặp với cùng độ phức tạp, chúng ta kết luận rằng tấn công trên suy ra tấn công va chạm có khởi đầu tự do trên hàm băm lặp LOKI-DBH yêu cầu 2 phép mã.

Tấn công va chạm có khởi đầu nửa tự do: Chúng tôi sẽ tìm giá trị của (H_{i-1}^1, H_{i-1}^2) và hai giá trị khác nhau của (M_i^1, M_i^2) dẫn đến cùng một giá trị (H_i^1, H_i^2) theo như (3). Chúng ta xử lý như sau:

Bước 1: Chúng ta chọn ngẫu nhiên giá trị x có m-bit

Bước 2: Giả sử z và $\hat{z}$ được định nghĩa bởi (11) và (12) một cách tương ứng. Cho x, chúng ta chọn ngẫu nhiên cặp $(y, \hat{y})$ gồm 2 giá trị khác nhau m-bit cho đến khi tìm được cặp giá trị dẫn đến sự trùng của z và $\hat{z}$, tức là $z = \hat{z}$. Bằng "nghịch lý ngày

sinh", điều đó cần khoảng $2^{m/2}$ phép mã và có xác xuất thành công là 0.63 để tìm được một cặp $(y, \hat{y})$ như vậy.

Bước 3: Bằng cách áp dụng ánh xạ ngược của (9), chúng ta nhận được giá trị (z, x⊕z) của (H_{i-1}^1, H_{i-1}^2) và hai giá trị khác nhau (y⊕z, x⊕y⊕z) và $(\hat{y} \oplus z, x \oplus \hat{y} \oplus z)$ cho (M_i^1, M_i^2).

Bằng cách áp dụng việc thay tương tự cũng như đối với tấn công va chạm có khởi đầu tự do, chúng ta có thể dễ dàng chứng minh được tính đúng đắn của tấn công này. Chúng ta kết luận rằng chính tấn công vừa mô tả kéo theo tấn công va chạm khởi đầu nửa tự do trên hàm băm lặp LOKI-DBH 2m-bit bằng cách dùng khoảng $2^{m/2}$ phép mã.

5. Các hàm vòng 2m-bit với tỷ lệ 1/2

Trong phần này, chúng ta xem xét các hàm vòng 2m-bit với tỷ lệ 1/2 dựa trên mã khối (m,m). Giả sử các giá trị băm 2m-bit H_i có thể viết như là phép nối (ký hiệu bằng :) của hai vecto m-bit H_i^1 và H_i^2 sao cho $H_i = H_i^1 : H_i^2$, chúng ta có thể viết (1) như là

$$H_i^1 : H_i^2 = round(H_{i-1}^1 : H_{i-1}^2, M_i) \qquad (13)$$

với M_i là khối tin m-bit.

Dạng tổng quát của hàm vòng 2m-bit với tỷ lệ 1/2

$$\begin{cases} H_i^1 = E_A(B) \oplus C \\ H_i^2 = E_R(S) \oplus T \end{cases} \qquad (14)$$

với A, B và C là các tổ hợp tuyến tính nhị phân của các vecto m-bit H_{i-1}^1, H_{i-1}^2 và M_i, còn R, T và S là các tổ hợp nào đó (không nhất thiết phải tuyến tính) của các vecto $H_{i-1}^1, H_{i-1}^2, H_i^1$ và M_i. Chúng ta có thể viết A, B và C theo kiểu ma trận như sau:

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \prod \begin{bmatrix} H_{i-1}^1 \\ H_{i-1}^2 \\ M_i \end{bmatrix} \qquad (15)$$

với Π ký hiệu ma trận 3x3 có các thành phần là 0 hoặc 1.

Từ phần 2 chúng ta biết rằng độ phức tạp của hàm băm lặp 2m-bit với tỷ lệ 1 có hàm vòng dạng (4) bị chặn trên bởi 2^m đối với tấn công đích có khởi đầu tự do và bởi $2^{m/2}$ đối với tấn công va chạm có khởi đầu tự do. Bây giờ chúng ta sẽ chỉ ra rằng cũng những cận trên ấy vẫn đúng cho hàm băm lặp 2m-bit với tỷ lệ 1/2 có hàm vòng thuộc dạng (14).

Khẳng định 3: Với hàm băm lặp có tỷ lệ 1/2 mà hàm vòng 2m-bit có dạng (14) thì độ phức tạp của tấn công đích có khởi đầu tự do được chặn trên bởi 2^m và độ phức tạp của tấn công va chạm có khởi đầu tự do được chặn trên bởi $2^{m/2}$.

Chứng minh:

Tấn công đích có khởi đầu tự do: Cho giá trị $(H_{i-1}^1, H_{i-1}^2, M_i)$, tìm giá trị khác $(\hat{H}_{i-1}^1, \hat{H}_{i-1}^2, \hat{M}_i)$ sao cho dẫn đến cùng giá trị (H_i^1, H_i^2) theo như (14). Khi ma trận Π được định nghĩa trong (15) là không suy biến, giả sử D là giá trị của H_i^1 đối với bộ giá trị đã cho $(H_{i-1}^1, H_{i-1}^2, M_i)$. Khi đó chúng ta có thể tạo ra 2^m giá trị khác nhau của $(\hat{H}_{i-1}^1, \hat{H}_{i-1}^2, \hat{M}_i)$ sao cho dẫn đến cùng một giá trị D bằng cách đầu tiên tính $C = D \oplus E_A(B)$ cho 2^m giá trị chọn ngẫu nhiên của (A,B) và sau đó, với mỗi giá trị của (A,B,C) ta tính $(\hat{H}_{i-1}^1, \hat{H}_{i-1}^2, \hat{M}_i)$ theo cách sau:

$$\begin{bmatrix} \hat{H}_{i-1}^1 \\ \hat{H}_{i-1}^2 \\ \hat{M}_i \end{bmatrix} = \prod{}^{-1} \begin{bmatrix} A \\ B \\ C \end{bmatrix}$$

với Π^{-1} là ma trận đảo của ma trận không suy biến Π. Khi ma trận Π là suy biến, thì với giá trị (A,B,C) nhận được từ giá trị đã cho của $(H_{i-1}^1, H_{i-1}^2, M_i)$, tồn tại ít nhất 2^m giá trị khác nhau của $(\hat{H}_{i-1}^1, \hat{H}_{i-1}^2, \hat{M}_i)$ dẫn tới cùng một giá trị (A,B,C), tức là cùng giá trị H_i^1. Với giá trị đã cho và với 2^m giá trị vừa mới sinh ra của $(\hat{H}_{i-1}^1, \hat{H}_{i-1}^2, \hat{M}_i)$, chúng ta tính giá trị của H_i^2 theo như (14). Bởi vì có 2^m giá trị có thể của H_i^2, khi chúng ta tính H_i^2 cho 2^m giá trị $(\hat{H}_{i-1}^1, \hat{H}_{i-1}^2, \hat{M}_i)$ thì với xác suất 0.63 tìm được một giá trị của $(\hat{H}_{i-1}^1, \hat{H}_{i-1}^2, \hat{M}_i)$ sao cho dẫn đến cùng một giá trị của H_i^2 như đối với giá trị đã cho $(H_{i-1}^1, H_{i-1}^2, M_i)$. Phép tấn công này cần khoảng 2^m phép tính.

Tấn công va chạm có khởi đầu tự do: chúng ta tìm 2 giá trị khác nhau của $(H_{i-1}^1, H_{i-1}^2, M_i)$ sao cho dẫn đến cùng một giá trị của (H_i^1, H_i^2) theo (14). Tấn công này là tương tự với tấn công đích có khởi đầu tự do, ngoại trừ rằng chúng ta chỉ tạo ra $2^{m/2}$ giá trị của $(\hat{H}_{i-1}^1, \hat{H}_{i-1}^2, \hat{M}_i)$ dẫn đến cùng một giá trị H_i^1. Từ nghịch lý ngày sinh suy ra rằng với xác suất 0.63, khi chọn ngẫu nhiên $2^{m/2}$ gía trị của $(\hat{H}_{i-1}^1, \hat{H}_{i-1}^2, \hat{M}_i)$ sẽ tìm được 2 giá trị có cùng giá trị H_i^2.

Ví dụ Meyer-Schilling (có sửa đổi)

Trong [Meyer 88], Meyer và Schilling đã đề xuất một hàm băm 2m-bit dựa trên mã khối DES (m=64, k=56), sau đó nó được gọi là MDC-2 trong [Matyas 91] và hiện nay đang được xem xét như là chuẩn ISO [ISO 91]. Lược đồ Meyer-Schilling, sau một số thay đổi nhỏ, đó là, sử dụng mã khối (m,m) thay cho DES, không xét phép "cut-and-paste" phụ thêm và các phép tráo đổi, có thể viết như sau:

$$H_i^1 = E_{H_{i-1}^1}(M_i) \oplus M_i$$

$$H_i^2 = E_{H_{i-1}^2}(M_i) \oplus M_i$$

hoặc dưới dạng tổng quát (14) như sau

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} H_{i-1}^1 \\ H_{i-1}^2 \\ M_i \end{bmatrix} \text{ và } \begin{bmatrix} R \\ S \\ T \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} H_{i-1}^1 \\ H_{i-1}^2 \\ M_i \end{bmatrix}$$

Cho nên, các cận trên của Khẳng định 3 cũng đúng cho lược đồ Meyer-Schilling. Bằng cách áp dụng phương pháp tương tự như trong chứng minh của Khẳng định 2, chúng ta có thể chỉ ra rằng lược đồ Meyer-Schilling thực chất đạt được cận trên nếu mã khối ở trong không có điểm yếu.

6. Kết luận

Trong bài báo này, chúng tôi đưa ra các cận trên cho độ phức tạp của tấn công đích có khởi đầu tự do và tấn công va chạm có khởi đầu tự do đối với lớp rộng lớn các hàm băm lặp 2m-bit dựa trên mã khối (m, m). Mặc dù các tấn công đích có khởi đầu tự do và tấn công va chạm có khởi đầu tự do không phải là các tấn công thực sự (vì giá trị khởi đầu của hàm băm lặp luôn cố định), độ phức tạp của chúng là cận dưới cho các độ phức tạp của các tấn công đích và tấn công va chạm thực sự.

Chúng tôi cũng đề xuất hàm băm lặp 2m-bit với tỷ lệ 1, với giả thiết rằng nếu sơ đồ DM là an toàn và cách làm mạnh MD được sử dụng, thì chứng minh được rằng nó là tối ưu chống lại các tấn công đích có khởi đầu tự do và tấn công va chạm có khởi đầu tự do. Hơn nữa, tấn công va chạm có khởi đầu tự do mới của chúng tôi đới với sơ đồ LOKI-DBH chỉ ra rằng sơ đồ này là không tối ưu chống lại tấn công va chạm có khởi đầu tự do. Cuối cùng, mặc dù hàm băm lặp 2m-bit Meyer-Schilling cũng là tối ưu đối với 2 tấn công ấy, nhưng nó chỉ đạt tỷ lệ 1/2, trong khi đề xuất của chúng tôi đạt tỷ lệ 1, có nghĩa là 2 phép mã cần trong lược đồ Meyer-Schilling để băm một khối tin m-bit trong khi lược đồ của chúng tôi chỉ cần 1.

Vì lược đồ m-bit Davies-Meyer dường như là an toàn, cho nên có một câu hỏi mở [Preneel 93, Lai 92] là liệu có thể sửa đổi lược đồ DM để kiến thiết hàm băm lặp 2m-bit sao cho an toàn hơn so với lược đồ m-bit DM ban đầu. Bài toán này được giải một phần nhờ các kết quả của các Bổ đề 1 và 3, chúng đã chỉ ra rằng, khi cho một hàm vòng 2m-bit thực hiện băm ít nhất một khối tin m-bit bằng cách dùng 2

lần mã khối m-bit với khoá m-bit thì luôn tồn tại các tấn công có khởi đầu tự do (đích và va chạm) tốt hơn các tấn công vét cạn. Cho nên, bằng cách dùng 2 lần mã khối (m, m) cùng với một số phép toán đơn giản thì ***không thể*** nhận được hàm vòng 2m-bit với tỷ lệ 1/2 hoặc lớn hơn sao cho an toàn hơn hàm vòng m-bit DM trong việc chống lại các tấn công có khởi đầu tự do. Cho nên, để nhận được một hàm vòng 2m-bit an toàn, người ta cần phải thực hiện mã khối (m,m) ít nhất 3 lần trong một vòng, hoặc phải sử dụng 2 lần mã khối m-bit với độ dài khoá lớn hơn m. Thực tế, đã có các hàm vòng 2m-bit an toàn được kiến thiết bằng cách sử dụng 2 lần ma khối m-bit với khoá 2m-bit [Lai 92].

Tài liệu tham khảo

Brown 90 L. Brown, J. Pieprzyk and J. Seberry, "*LOKI-A Cryptographic Primitive for Authentication and Secrecy Application*", Advances in Cryptology-AUSCRYPT'90 Proceedings, pp.229-236, Springer-Verlag, 1990

Damgård 90 I.B. Damgård, "*A Design Principle for Hash Functions*", Advances in Cryptology- CRYPTO'89 Proceedings, pp. 416-427, Springer-Verlag, 1990

Davies 85 R.W. Davies and W.L. Price, "*Digital Signature-an Update*", Proc. International Conference on Computer Communications, Sydney, Oct.1984, Elsevier, North Holland, pp. 843-847, 1985.

ISO 91 ISO/IEC CD 10118, *Information technology-Security techniques-Hash functions*, I.S.O., 1991.

Lai 92 X. Lai and J. L. Massey, "*Hash Functions Based on Block Ciphers*", Advances in Cryptology-EUROCRYPT'92 Proceedings, pp. 55-70, LNCS 658, Springer-Verlag, 1993

Matyas 85 S. M. Matyas, C.H. Meyer and J. Oseas, "*Generating Strong One-Way Functions with Cryptographic Algorithm*", IBM Technical Disclosure Bulletin, Vol.27, No. 10A, pp. 5658-5659, March 1985.

Matyas 91 S.M. Matyas, "*Key Processing with Control Vectors*", Journal of Cryptology, Vol.3, No.2, pp.113-136, 1991.

Merkle 90 R. C. Merkle, "*One-Way Hash Functions and DES*", Advances in Cryptology-CRYPTO'89 Proceedings, pp.428-446, Springer-Verlag, 1990.

Meyer 88 C.H.Meyer and M.Schilling, "*Secure Program Code with Modification Detection Code*", Proceedings of SECURICOM 88, pp.111-130, SEDEP.8, Rue de la Michodies, 75002, Paris, France.

Miyaguchi 91 S. Miyaguchi, K. Ohta and M. Iwata, "*Confirmation that Some Hash Functions Are Not Collision Free*", Advances in Cryptology-EUROCRYPT'90 Proceedings, LNCS 473, pp.326-343, Springer-Verlag, Berlin, 1990.

Naor 89 M. Naor and M. Yung, "*Universal One-Way hash Functions and*

Their Cryptographic Applications", Proc. 21 Annual ACM Symposium on Theory of Computing, Seattle, Washington, May 15-17, 1989, pp. 33-43.

Preneel 89 B. Preneel, A. Bosselaers, R. Govaerts and J. Vandewalle, "*Collision-Free Hash functions Based on Blockcipher Algorithm*", Proceedings of 1989 International Carnahan Conference on Security Technology, pp. 203-210, 1989.

Preneel 93 B. Preneel, *Analysis and Design of Cryptographic Hash Functions*, Ph.D thesis, Katholieke Universiteit Leuven, Belgium, January 1993.

Quisquater 89 J.J.Quisquater and M. Girault, "*2n-bit Hash Functions Using n-bit Symmetric Block Cipher Algorithms*", Abstracts of EUROCRYPT'89

Winternitz 84 R.S. Winternitz, "*Producing One-Way Hash Function from DES*", Advances in Cryptology-CRYPTO'83 Proceedings, pp.203-207, Plenum Press, New York, 1984.

Các bài có liên quan

1. Xuejia Lai, Lars R. Knudsen, Attacks on Double Block Length Hash Functions, Proceedings of The Algorithm Workshop, Cambridge, U.K, Dec., 1993. (đã có, 10 trang)
2. L. R. Knudsen, X. Lai, B. Preneel, Attacks on Fast Double Length Hash Functions, Journal of Cryptology, sau 1996 (chưa có)
3. L. Knudsen, X. Lai, "New Attacks on All Double-Block Length Hash Function of Hash Rate 1, Including the Parallel-DM", EUROCRYPT 94, p. 410-408

PHÂN PHỐI KHOÁ VÀ THOẢ THUẬN

1. TRUYỀN TIN BẢO MẬT TRÊN MẠNG

Trên các mạng công cộng không an toàn hai bên nào đó muốn liên lạc an toàn với nhau thì phải lập mã thông tin trong phiên liên lạc. Muốn vậy họ phải có khoá bí mật trước để liên lạc an toàn với nhau. Quá trình trao đổi khoá bí mật phải xảy ra trước đó bằng các kênh an toàn.

Người ta phải giải quyết vấn đề này nhờ vào quá trình phân phối khoá và quá trình thoả thuận khoá trước khi liên lạc.

1.1 Tổ chức truyền tin bảo mật trên mạng

Chúng ta đi vào xem xét chi tiết của công việc tổ chức truyền tin bảo mật trên mạng.

Phân phối khoá: là cơ chế liên lạc bảo mật mà một bên chọn khoá bí mật và sau đó truyền khoá này đến cho bên kia.

Việc truyền khoá mật dĩ nhiên phải xảy ra bằng một kênh an toàn. Sau khi truyền xong khoá bí mật thì hai bên liên quan sẽ có thể liên lạc an toàn với nhau trên kênh không an toàn.

Thoả thuận khoá: là giao thức mà hai hay nhiều bên cùng nhau thiết lập khoá bí mật bằng liên lạc trên kênh công khai.

Đây là khái niệm mới chỉ dùng kênh công cộng mà có thể thiết lập khoá mật giữa các bên liên lạc. Trong sơ đồ thoả thuận khoá thì khoá chung là giá trị của hàm số mà đầu vào được cả hai bên cung cấp. Môi trường liên lạc của chúng ta thường là n người sử dụng dùng mạng liên lạc không an toàn để liên lạc an toàn với nhau. Người ta cần có n(n-1)/2 cặp khoá chung bí mật trước khi liên lạc với nhau. Đây là một công việc tốn kém.

Mục tiêu của phân phối khoá và thoả thuận khoá là cuối giao thức đó hai bên liên quan đều có cùng một khoá K và không một bên nào khác biết giá trị của khoá K. Nếu không có một trung tâm uỷ nhiệm thì để cung cấp khoá cho n người dùng trung tâm này phải truyền đi C_n^2 khoá một cách an toàn. Điều này là rất tốn kém và phức tạp nếu mạng có nhiều người sử dụng.

Người ta nghĩ ra các phương thức làm giảm tối thiểu số khoá bí mật cần phân phối trên mạng. Đó là kiểu phân phối khoá Blom hay phân phối khoá trực tuyến bởi trung tâm uỷ nhiệm Kerberos hay giao thức thoả thuận khoá Diffie-Hellman.

1.2 Sơ đồ tiền phân phối khoá Blom và Diffie-Hellman

1.2.1 Sơ đồ phân phối khoá Blom

Sơ đồ này làm giảm tối thiểu việc truyền và lưu trữ khoá bằng cách tính trực tiếp khoá giữa hai bên liên lạc. Người ta vẫn quen gọi nó là sơ đồ tiền phân phối khoá Blom. Sơ đồ này hoạt động như sau:

1.Trung tâm uỷ nhiệm chọn một số nguyên tố p và chọn cho mỗi người dùng U một số $r_u \in Z_p$, các r_u khác nhau. Số p và các số r_u được công bố công khai.

2.Trung tâm chọn ba số ngẫu nhiên a, b, c $\in Z_p$ không cần phải khác nhau và tạo ra đa thức $f(x,y) = a + b(x + y) + cxy \mod p$

3.Đối với mỗi người dùng U, trung tâm uỷ nhiệm tính đa thức: $g_u(x) = f(x,r_u)$ và truyền $g_u(x)$ cho U trên kênh an toàn. Nhận xét rằng $g_u(x)$ là đa thức tuyến tính theo x nên nó có thể được viết là: $g_u(x) = a_u + b_u x$, ở đó $a_u = a + br_u \mod p$ và $b_u = b + cr_u \mod p$

4.Nếu U và V muốn liên lạc thì họ dùng khoá chung

$K_{U,V} = K_{V,U} = f(r_u, r_v) = a + b(r_u + r_v) + cr_u r_v \mod p$

Trong đó U tính $K_{U,V}$ như là: $f(r_u, r_v) = g_U(r_v)$ và V tính $K_{U,V}$ như là: $f(r_U,r_V) = g_V(r_U)$

Ví dụ : giả sử có 3 người sử dụng là U, V và W, p = 17, còn các phần tử công khai của họ là r_u =12, r_v=7, r_w=1. Giả thiết rằng trung tâm chọn a = 8, b = 7 và c = 2, khi đó đa thức f có dạng như sau:

$f(x,y) = 8 + 7(x + y) + 2xy$

Các đa thức g như sau:

$g_u(x) = 7 + 14x$

$g_v(x) = 6 + 4x$

$g_w(x) = 15 + 9x$

Như vậy ba khoá sẽ là :

$K_{U,V} = 3$

$K_{U,W} = 4$

$K_{V,W} = 10$

U tính $K_{U,V}$ như sau: $g_u(r_v) = 7 + 14 \text{ x } 7 \bmod 17 = 3$
V sẽ tính $K_{U,V}$ như sau: $g_v r_u = 6 + 4 \text{ x } 12 \bmod 17 = 3$

Chỉ một mình người dùng U có thông tin a_u, b_u do đó chỉ có anh ta mới tính được khoá chung với người dùng V là:

$K_{U,V} = g_U(r_v) = a_u + b_u r_v \bmod p.$

Do p lớn nên không thể duyệt tất cả các cặp (a_u ,b_u) để tìm ra cặp đúng và nếu người dùng khác U,V cũng không thể biết được $K_{U,V}$ nào là đúng.

Sơ đồ Blom bảo đảm sự bí mật hoàn toàn của khoá $K_{U,V}$ đối với bất kỳ một người thứ ba W nào, theo nghĩa rằng với những thông tin mã W có được (W biết a_w,b_w) thì bất kỳ giá trị $l \in Z_p$ nào cũng có thể nhận là $K_{U,V}$.

Ta có định lý sau: Sơ đồ Blom là an toàn không điều kiện trước bất kỳ người sử dụng cá biệt nào.

1.2.2 Sơ đồ phân phối khoá Diffie-Hellman

Sơ đồ này là an toàn về mặt tính toán nếu bài toán tính logarit rời rạc là không thể giải được. Sơ đồ hoạt động như sau:

1.Số nguyên tố p và phần tử nguyên thuỷ $\alpha \in Z_p^*$ được công bố công khai. Ngoài ra chứng chỉ có chứa tham số công khai của U và V cũng được công bố.

2. V dùng giá trị b_U công khai từ chứng chỉ của U cùng với giá trị bí mật của riêng mình a_V để tính: $K_{U,V} = \alpha^{a_u a_v} \bmod p = b_U{}^{a_v} \bmod p$

3. U dùng giá trị công khai b_V từ chứng chỉ của V cùng với giá trị bí mật của riêng a_u để tính : $K_{U,V} = \alpha^{a_u a_v} \bmod p = b_v{}^{a_v} = \bmod p$

Vì b_u, b_V công khai. Muốn có khoá $K_{U,V}$ người mã thám chỉ biết b_u, b_V do đó phải thử các a_v, a_u để tìm ra giá trị đúng. Do p lớn nên việc duyệt các khả năng của a_v, a_u là không thể được và sẽ không tính được $K_{U,V}$ trừ khi giải được bài toán lôgarit rời rạc (g là phần tử sinh của trường Z_p^*):

-Cho phần tử $b_u \in Z_p$. Hãy tìm số tự nhiên bé nhất a_u sao cho $b_u = g^{a_u} \bmod p$.

-Cho phần tử $b_v \in Z_p$. Hãy tìm số tự nhiên bé nhất a_v sao cho $b_v = g^{a_v} \bmod p$.

Do trung tâm uỷ nhiệm đã xác nhận các chứng chỉ của người sử dụng nên ngăn cản được một cách hiệu quả người dùng W muốn sửa đổi thông

tin về chứng chỉ của bất kỳ ai khác. Do có xác nhận các chứng chỉ dùng chữ ký số nên việc giả mạo chứng chỉ là không thể được (Tương đương với việc giả mạo chữ ký số).Vấn đề còn lại chỉ là tấn công bị động của W tức là tìm cách tính $K_{U,V}$ từ $\alpha^{a_u} \bmod p$ và $\alpha^{a_v} \bmod p$. Đây là bài toán Diffie-Hellman. Do đó sơ đồ Diffie-Hellman là an toàn chống lại các tấn công thụ động nếu bài toán Diffie-Hellman là không giải được. Chúng ta thấy rằng các sơ đồ trên phân phối khoá không chỉ tiết kiệm truyền và lưu trữ tin mà chúng còn chủ động ngăn chặn các tấn công chủ động của các bên không liên quan.

1.3 Hệ phân phối khoá X.509 và Kerberos

Đây là hai loại hình phân phối khoá điển hình có và không có trung tâm uỷ nhiệm.

1.3.1 Sơ đồ phân phối khoá X.509

Sơ đồ phân phối khoá X.509 có chức năng hoạt động trực tuyến không có trung tâm uỷ nhiệm, xác thực lẫn nhau và cung cấp khoá phiên liên lạc dùng mật mã khoá công khai.

Khoá phiên không cố định và luôn được tạo ra mới mỗi lần hai bên nào đó có nhu cầu liên lạc sau khi đã hoàn thành giao thức X.509. Chúng ta mô tả hoạt động của giao thức phân phối khoá như sau:

$$1.\ A \rightarrow B : A, S_A\{ts_{AB}, nrv_{ab}, B, E_B\{key_{AB}\}\}$$
$$2.\ B \rightarrow A : S_B\{ts_{BA}, nrv_{BA}, A, nrv_{AB}, E_A\{key_{BA}\}\}$$
$$3.\ A \rightarrow B : S_A\{B, nrv_{BA}\}$$

Trong đó các ký hiệu được biểu diễn như sau:

-A, B là hai bên liên lạc liên quan

-$E_x\{\}$ lập mã một dãy số liệu dùng khoá công khai của bên X

-$S_x\{\}$ là dãy số liệu và chữ ký số của chúng dùng khoá bí mật của bên X.

-ts_{xy} là tem thời gian hiện hành sinh bởi bên X để giúp bên Y phát hiện các thông báo được dùng lại.

-nrv_{xy} là giá trị không lặp lại được gửi bởi bên X để giúp bên Y phát hiện các thông báo dùng lại.

-key_{xy} là khoá bí mật sinh bởi bên X được dùng để bảo vệ các liên lạc sau đó giữa X và Y.

X.509 ba pha thực chất là sự kết hợp của thách đố/trả lời dùng để xác thực lẫn nhau. Các thách đố đó ở đây chính là nrv_{AB}, nrv_{BA} . Ngoài ra tem thời gian chỉ là phần phụ trợ thêm.

Các khoá được bảo vệ bằng các hệ mật khoá công khai của người nhận. Tính nguyên vẹn của các luồng thông tin qua lại và tính "tươi mới" của các giá trị *nrv* được bảo vệ bởi các sơ đồ chữ ký số.

Các hệ mật khoá công khai và các sơ đồ chữ ký số đều được chọn an toàn chống lại các tấn công của các bên không liên quan.

Tuy nhiên vấn đề tính nguyên vẹn của cả quá trình xác thực phải được đảm bảo bằng độ an toàn của giao thức X.509. So với Kerberos, sơ đồ X.509 không đòi hỏi trung tâm uỷ nhiệm trực tuyến. Đối với sơ đồ X.509 ba pha thì thời gian là không cần thiết nữa.

Tuy nhiên nếu khoá xác thực giữa các bên mà bị lộ thì tất cả các phiên liên lạc có liên quan đến khoá này đều bị lộ theo.

1.3.2 Sơ đồ phân phối khoá Kerberos

Là sơ đồ phân phối khoá trực tuyến với trung tâm uỷ nhiệm, xác thực lẫn nhau và cung cấp khoá phiên liên lạc dùng mật mã khoá đối xứng. Chúng ta mô tả hoạt động của Kerberos như sau:

1. Người dùng U yêu cầu trung tâm uỷ nhiệm TA cung cấp khoá phiên để liên lạc với V.

2. Trung tâm uỷ nhiệm TA chọn khoá ngẫu nhiên của phiên K, tem thời gian T và khoảng tồn tại L (K có giá trị trongkhoảng thời gian từ T đến T + L)

3. Trung tâm uỷ nhiệm tính: $m_1 = e_{K_U}(K, ID(V), T, L)$, $m_2 = e_{K_V}(K, ID(U), T, L)$ và gửi m_1, m_2 cho U.

4. U dùng hàm dịch d_{K_U} để tính K, T, L và ID(V) từ m_1. Sau đó U lại tính $m_3 = e_K(ID(U),T)$ và gửi cho V các thông báo m_3, m_2.

5. V dùng hàm dịch d_{K_V} để tính K,T, L và ID(U) từ m_2. V lại tính T, ID(V) từ m_3 dùng d_K. Anh ta kiểm tra 2 giá trị của T và 2 giá trị của ID(U) xem có giống nhau không. Nếu vậy thì V tính tiếp: $m_4 = e_K(T + 1)$.

6. U dịch m_4 dùng d_K và kiểm tra xem kết quả có phải là T + 1 không.

Các ký hiệu trong sơ đồ là như sau:

-Mỗi người dùng U chia sẻ khoá bí mật DES là K_U với trung tâm uỷ nhiệm TA.

-ID(U) là thông tin định danh công khai của người dùng U.

-Khoá K là khoá phiên ngẫu nhiên do TA sinh ra mỗi khi có nhu cầu tới. TA cũng ghi lại tem thời gian T và khoảng tồn tại L mà khoá K sẽ hợp lệ trong khoảng thời gian đó, tức là khoá K sẽ được coi là hợp lệ từ thời điểm T đến thời điểm T + L.

-Các thông báo m_1, m_2 cung cấp bí mật trong truyền tin của khoá phiên K. Trong khi đó m_3, m_4 cung cấp sự đảm bảo về khoá tức là cho U, V tin rằng họ có cùng một khoá K.

-Các tem thời gian T, và khoảng tồn tại L nhằm ngăn chặn các tấn công chủ động hoặc dùng lại các thông báo.

Xem xét độ an toàn của sơ đồ Kerberos giống như X.509 về vấn đề độ mật mật mã nhưng độ mật giao thức của nó sẽ phức tạp hơn X.509. Một hạn chế cơ bản của Kerberos là cần phải đồng bộ thời gian của các người dùng cùng với trung tâm uỷ nhiệm.

2. TRAO ĐỔI KHOÁ

Đôi khi dùng trung tâm uỷ nhiệm trực tuyến để cung cấp khoá phiên cũng không được ưa chuộng. Người ta phải tìm đến với giao thức thoả thuận khoá để trao đổi các khoá bí mật.

2.1 Giao thức trao đổi khoá Diffie-Hellman

Là giao thức thoả thuận khoá đầu tiên được biết đến và rất quen thuộc. Hoạt động của giao thức là như sau: p là số nguyên tố, α là phần tử nguyên thuỷ của p và các giá trị này được công bố công khai.

1. U chọn a_U ngẫu nhiên, $0 \le a_U \le p - 1$.
2. U tính $\alpha^{a_U} \bmod p$ và gửi nó cho V.
3. V chọn a_V ngẫu nhiên, $0 \le a_V \le p - 2$.
4. V tính $\alpha^{a_V} \bmod p$ và gửi nó cho U.
5. U tính $K = (\alpha^{a_v})^{a_u} \bmod p$ và V tính $K = (\alpha^{a_u})^{a_v} \bmod p$.

Cuối cùng cả U và V có khoá chung K. Khác với sơ đồ này, mỗi lần khoá K lại sinh ra mới một lần do a_U, a_V luôn được sinh ra mới mỗi khi có phiên liên lạc.

Một yếu điểm căn bản trong sơ đồ này là nó không chống lại được tấn công kẻ ở giữa. kẻ ở giữa W đóng vai trò V đối với U và đóng vai U đối với V và kết quả là cả U và V đều bị lừa rằng mình đang liên lạc với nhau thực ra họ đã liên lạc với W.

Do không có xác thực nên cả U và V đều không biết chắc được khoá K tạo ra có đúng là khoá K giữa U và V hay không. U không dám chắc $\alpha^{a_u} \bmod p$ đến được từ V và không bị thay đổi. V không dám chắc $\alpha^{a_v} \bmod p$ đến được từ U mà không bị thay đổi. Rất có thể là U có $K' = (\varepsilon^{a'_v})^{a'_u} \bmod p$ và V có $K'' = (\varepsilon^{a'_u})^{a'_v} \bmod p$. Để chống lại kiểu tấn công nguy hiểm này người ta đã cải tiến giao thức thành giao thức thoả thuận khoá có xác thực.

2.2 Sơ đồ phân phối khoá có xác thực Diffie-Hellman

Đây là sơ đồ cải tiến của giao thức trao đổi khoá Diffie-Hellman. Giao thức giả thiết số nguyên tố p và phần tử nguyên thuỷ $\alpha \in Z_p$ được công bố công khai. Mỗi người dùng U có một sơ đồ chữ ký với thuật toán kiểm tra Ver_U và thuật toán ký Sig_U. Trung tâm uỷ nhiệm TA cũng có sơ đồ chữ ký với thuật toán kiểm tra Ver_{TA}. Mỗi người dùng U có một chứng chỉ: $C(U) = (ID(U), Ver_U, Sig_{TA}((ID(U),Ver_U))$, ở đó ID(U) là thông tin định danh của U.

Sơ đồ hoạt động như sau:

1. U chọn số ngẫu nhiên a_U, $0 \le a_U \le p - 2$.
2. U tính $\alpha^{a_u} \bmod p$ và gửi cho V.
3. V chọn một số ngẫu nhiên a_V, $0 \le a_V \le p - 2$.
4. V tính $\alpha^{a_v} \bmod p$ và tính tiếp $K = (\alpha^{a_u})^{a_v} \bmod p$.

   Và $y_V = Sig_V(\alpha^{a_v}, \alpha^{a_u})$.
5. V gửi $(C(V), \alpha^{a_v}, y_V)$ cho U.
6. U tính: $K = (\alpha^{a_v})^{a_u} \bmod p$ và kiểm tra y_V dùng Ver_V và kiểm tra C(V) dùng Ver_{TA}.
7. U tính $y_U = Sig_U(\alpha^{a_u}, \alpha^{a_v})$ và gửi $(C(U),y_U)$ cho V
8. V kiểm tra y_U dùng Ver_U và kiểm tra C(U) dùng Ver_{TA}.

Do không có các thuật toán ký của U và V nên kẻ tấn công ở giữa W không thể làm được gì cả.

Tuy nhiên, sơ đồ khoá này chưa đảm bảo được khoá chung đã đúng hay chưa. Muốn đạt được vậy, người ta có thể sửa lại:

$y_V = e_K(Sig_V(\alpha^{a_v}, \alpha^{a_u}))$ trong bước 4

và $y_U = e_K(Sig_U(\alpha^{a_u}, \alpha^{a_v}))$ trong bước 6.

Có thể có nhiều sơ đồ cải biên khác để chống lại tấn công kẻ xen vào ở giữa.

Xác thực và trao đổi khóa có xác thực[1]

WHITFIELD DIFFIE
Sun Mucrosystems, 2550 Garcia Ave., Mountain View, CA 94043

PAUL C. VAN OORSCHOT AND MICHAEL J. WIERNER
Bell-Northen Reseach, P.O. Box 3511 Station C, Ottawa, Ontario KIY 4H7
Canada

Giới thiệu bởi S.A. Vanstone
Nhận 22-11-1991, sửa lại 6-3-1992.

Tóm tắt: Chúng ta bàn về các thủ tục tương tác 2 bên nhằm trao đổi khóa có xác thực, chủ yếu nhằm vào các thủ tục có sử dụng kỹ thuật phi đối xứng. Một thủ tục đơn giản, hiệu quả được gọi là thủ tục trạm-tới trạm (station-to-stattion hay STS) được giới thiệu, phân tích kỹ lưỡng, và so sánh với các thủ tục hiện có. Định nghĩa thế nào là thủ tục an toàn cũng được xem xét, các đặc trưng mong muốn của thủ tục an toàn cũng được bàn tới.

1. Mở đầu

Mục đích của thủ tục xác thực là cung cấp cho các bên tham gia liên lạc một sự tin tưởng rằng họ biết định danh đúng của phía bên kia. Trong trao đổi khóa có xác thực, mục đích thêm mà 2 bên nhận được là chia sẻ một khóa chung mà chỉ có họ biết. Khóa bí mật đó có thể được dùng sau đó để đảm bảo bí mật, toàn vẹn dữ liệu, hay cả hai. Trong bài báo này, chúng ta bàn về độ an toàn của các thủ tục xác thực dựa trên mật mã khóa công khai, có thể hay không kèm theo việc trao đổi khóa. Chúng ta giới hạn mối quan tâm của mình tới việc xác thực 2 bên, thay cho các thủ tục nhiều bên hay một bên. Chúng ta giả thiết rằng các kỹ thuật mật mã nằm ở phía dưới là không bị tổn thương, và giới hạn mối quan tâm của mình chỉ ở phần thủ tục. Kẻ địch (người tấn công, người xâm nhập, đối phương) có thể xem tất cả các thông tin được trao đổi, có thể xóa, thay đổi, chèn hay có thể đổi hướng thông tin, có thể khởi đầu cuộc trao đổi với bất kỳ bên nào, và có thể sử dụng lại thông tin từ những cuộc liên lạc trước đó.

Nhiều công trình đã được hoàn thành trong những năm gần đây về các lược đồ nhận thực (identification) và xác thực (authentication) sử dụng kỹ thuật phi đối xứng. Các lược đồ dựa trên nhận dạng (identity-based) được Shamir [29] đưa ra, dựa vào sự có mặt của một nhà chức trách trung tâm tin cậy giữ thông tin bí mật, từ thông tin bí mật này các thông tin bí mật khác được sinh ra và phân phối đến từng cá thể khi các cá thể này tham gia vào hệ thống. Günther [15] đề nghị thủ tục dựa trên nhận diện đảm bảo việc thiết lập khóa bí mật, sử dụng ý tưởng của lược

[1] Designs, Codes and Cryptography, 2, 107-125 (1992), Kluwer Academic Publishers

đồ trao đổi khóa Diffie-Hellman [9] và lược đồ chữ ký số ElGamal [11]. Việc xác thực là gián tiếp và không cung cấp bí mật chuyển tiếp hoàn hảo (perfect forward secrecy) (xem Mục 4), mặc dù tính chất này có thể đạt được khi trả giá thêm một lần trao đổi nữa các đại lượng lũy thừa Diffie-Hellman. Okamoto và Tanaka [22] đề nghị một thủ tục thiết lập khóa có xác thực dựa trên nhận dạng bằng cách trao đổi lũy thừa khóa và RSA. Họ đưa ra phiên bản cung cấp việc xác thực gián tiếp hay trực tiếp, mặc dù việc xác thực trực tiếp, như đã trình bày, có dùng đến tem thời gian (Mục 4), và một số trường trong khi trao đổi là không cần thiết hoặc dư thừa. Các thủ tục nhận diện tương tác do Fiat và Shamir [12] đề xuất cung cấp bằng chứng nhận diện đã sử dụng các ý tưởng liên quan tới tri thức-không, và có một số thủ tục hiệu quả hơn đã được đề xuất liền sau đó, bao gồm Guillou và Quisquater [14] và Schnorr [28]. Những thủ tục nhận thực này khác các thủ tục trao đổi khóa có xác thực ở chỗ các thủ tục nhận thực không cung cấp khóa để sử dụng trong các liên lạc về sau (ví dụ cho toàn vẹn dữ liệu hay bí mật dữ liệu).

Như đã chỉ ra bởi nhiều tài liệu, [5], [7], [13], [19], [20], việc thiết kế thủ tục mật mã nói chung, thủ tục xác thực nói riêng là rất hay mắc lỗi. Trong các tài liệu có rất nhiều thủ tục được tìm thấy có chứa lỗi về độ an toàn mức độ từ nhẹ tới rất nặng. Hơn nữa, bên cạnh độ an toàn, thực tế cho thấy rằng nhiều thủ tục đã được công bố chứa những cái dư thừa hoặc là không hiệu quả theo quan điểm số lần trao đổi thông tin cần làm, số các phép toán mật mã đòi hỏi (yêu cầu công suất tính toán cao), hoặc số lượng hay kiểu của các trường được yêu cầu trong các thông tin được trao đổi. Những điều này thúc đẩy việc tìm kiếm các thủ tục xác thực đơn giản, yêu cầu tối thiểu số lần trao đổi, số ít các trường trong mỗi thông điệp hay thẻ, số ít các phép toán mật mã. Những suy nghĩ đó đã thúc đẩy công trình này trên các thủ tục dựa vào khóa công khai. Những suy nghĩ tương tự đã thúc đẩy Bird và các tác giả khác [5] trong công trình của họ về các thủ tục xác thực dựa trên mật mã đối xứng, chúng đã tập trung sự chú ý của chúng tôi đến ý tưởng trùng hợp của các vận hành thủ tục (xem Mục 3). Bài báo của chúng tôi mở rộng định nghĩa của thủ tục an toàn sang trường hợp thủ tục dựa trên khóa công khai có thể kèm thêm việc trao đổi khóa.

Chúng ta quan tâm tới cả việc xác thực và cả việc trao đổi khóa. Nên chấp nhận xem xét 2 chủ đề này cùng với nhau chứ không tách biệt [2]. Các thủ tục cung cấp việc xác thực nhưng không trao đổi khóa sẽ bị tổn thương bởi kẻ địch đợi cho đến khi việc xác thực đã hoàn thành rồi mới chiếm một đầu của kênh liên lạc. Tấn công này vẫn có tác dụng do việc trao đổi khóa độc lập với việc xác thực. Việc trao đổi khóa cần liên kết với việc xác thực so cho các bên tin rằng khóa được trao đổi (có thể được dùng cho việc bảo mật hay xác thực và như vậy giữ cho *tính xác thực sống động*) thực chất là cái được chia sẻ với đối tác được xác thực, chứ không phải với một kẻ giả mạo. Với các lý do này, cần phải luôn nhớ tới việc trao đổi khóa trong khi thiết kế và phân tích các thủ tục xác thực.

Trong phần còn lại của bài báo, trước hết chúng tôi cung cấp một số kiến thức nền tảng dùng để tấn công các thủ tục, đó là lý do để thúc đẩy và lý giải những gì được đưa ra sau đó. Sau đó chúng tôi đưa ra định nghĩa của thủ tục an

toàn, bàn về các đặc trưng mà chúng tôi cho rằng là cần có trong thủ tục xác thực. Chúng tôi giới thiệu một thủ tục được gọi là trạm-tới-trạm (station-to-station), kiểm tra nó chi tiết, bình luận các đặc tính của nó.Một số thủ tục có liên quan cũng được bàn đến, và các thủ tục được đề nghị là có liên quan đến chúng. Chúng tôi kết thúc cùng với việc tóm tắt những nguyên tắc mà chúng tôi cảm thấy là quan trọng trong việc thiết kế các thủ tục xác thực.

2. Khái niệm và những vấn đề đặt ra

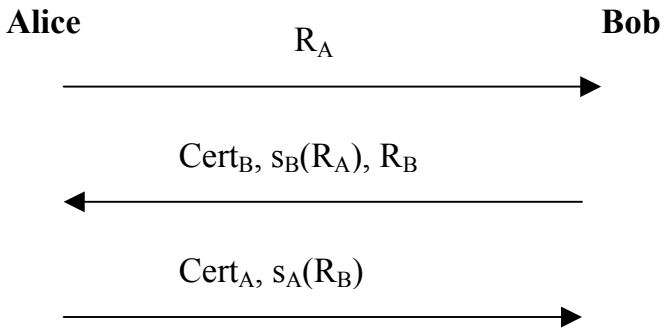
Trước khi bàn về các thủ tục một cách chi tiết, trước hết chúng ta định nghĩa một số khái niệm. Vì lý do lịch sử, chúng tôi gọi 2 bên tham gia là Alice và Bob.

{.}	Dấu ngoặc nhọn ký hiệu cho hàm băm. {x,y} là kết quả khi hàm băm được áp dụng cho x kết nối với y.
s_A	Khóa bí mật của Alice dùng để ký. $s_A(x)$ là chữ ký của Alice lên x. $s_A\{x\}$ là chữ ký của Alice lên giá trị băm của x.
p_A	Khóa công khai của Alice dùng trong lược đồ ký. Nếu lược đồ ký là hệ mật khóa công khai, thì chúng ta ký hiện $p_A\{x\}$ và $p_A(x)$ là hàm mã bởi khóa công khai của Alice lên giá trị băm của x hay lên chính x.
$Cert_A$	Chứng chỉ của Alice, chứa tên của Alice (và các thông tin khác có thể), khóa công khai của Alice, chữ ký của nhà chức trách tin cậy lên thông tin này. $Cert_A$ = (Alice, p_A, ..., s_T{Alice, p_A,...}). $Cert_A$ kết nối tên của Alice với khóa công khai p_A. Nếu Alice gửi chứng thực này của cô ta cho Bob và cung cấp bằng chứng rằng cô ấy biết khóa công khai s_A tương ứng với p_A, thì cô ta cung cấp được bằng chứng cho Bob rằng cô ấy chính là Alice.
$E_K(.)$	Mã hóa sử dụng hệ mật khóa đối xứng bằng khóa K,

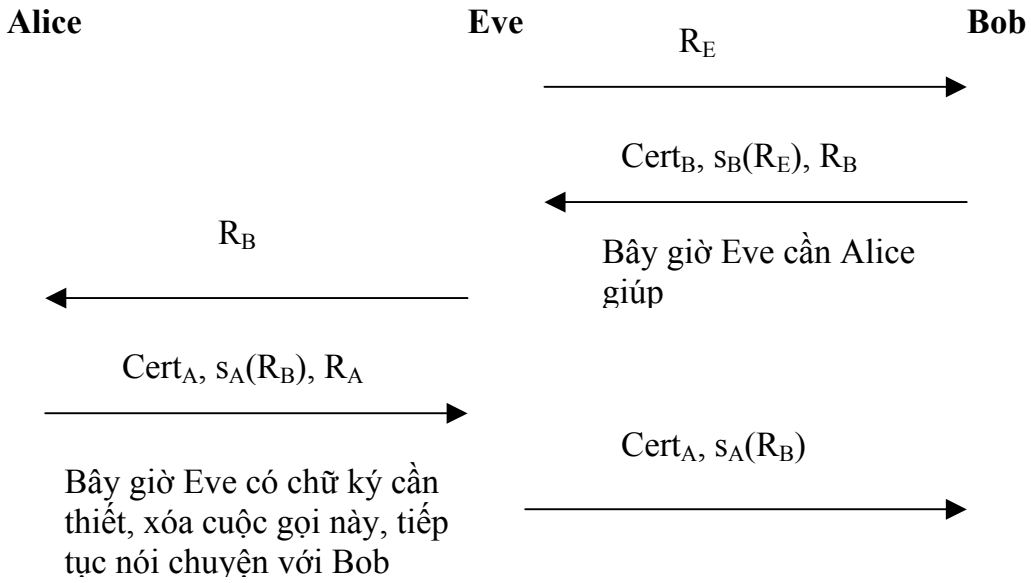
Để minh họa tấn công vào thủ tục và những gì đặt ra tiếp theo, hãy xem xét một thủ tục thách thức-trả lời đơn giản (nhưng có lỗi) khi Alice và Bob ký vào những số thách thức ngẫu nhiên của nhau.

Thủ tục thách thức-trả lời đơn giản không an toàn:

Khởi đầu, Alice gửi số ngẫu nhiên R_A cho Bob. Bob gửi trả lại chứng chỉ của anh ta, chữ ký của anh ta lên R_A, số ngẫu nhiên R_B. Alice sử dụng khóa công khai của Bob trong $Cert_B$ để kiểm tra chữ ký của Bob và sau đó phúc đáp bằng chứng chỉ của Alice và chữ ký của Alice lên R_B. Cuối cùng, Bob kiểm tra chữ ký của Alice.



Kẻ thù địch Eve có thể đóng giả Alice trong phiên liên lạc với Bob bằng cách chuyển những thách thức của Bob tới cho Alice.



Eve bắt đầu bằng việc khởi đầu thủ tục với Bob. Khi Bob gửi thách thức cho Eve, Eve khởi đầu thủ tục với Alice và làm cho Alice ký vào số ngẫu nhiên của Bob. Vấn đề chính ở đây là những người bị thách thức không có ảnh hưởng đến những gì mà anh ta ký vào (Bob ký vào R_A nhưng không sinh R_A) (Như một qui tắc chung, sẽ tốt hơn nếu cả hai bên có một ảnh hưởng nào đó đối với các đại lượng được ký.) Người thách thức có thể lạm dụng thủ tục này để nhận được chữ ký lên bất kỳ đại lượng nào mà anh ta chọn.

Bây giờ chúng ta chuyển sự chú ý của mình vào các thủ tục an toàn.

3. Định nghĩa của thủ tục an toàn

Một xuất hiện cụ thể của thủ tục xác thực được coi như là một *vận hành (run)*. Trước khi trình bày định nghĩa của thủ tục an toàn, trước hết chúng ta xem xét các tính chất của cái mà chúng ta coi là một vận hành thành công (successful run). Trong một vận hành thành công, hai đối tác liên lạc, Alice và Bob, trao đổi một số thông điệp, khi kết thúc thì họ có được niềm tin vào nhận dạng của nhau và hơn nữa, có thể là họ chia sẻ một khóa chung mà chỉ có họ biết. Với mỗi lần vận hành được thực hiện xong, mỗi bên hoặc là *chấp nhận* hoặc là *bác bỏ* nhận dạng của bên kia, và có thể làm thêm việc trao đổi khóa. Trong mỗi lần vận hành thành công, việc vận hành được thực hiện xong và cả hai bên đều chấp nhận.

Tính chất 1 của vận hành thành công: Cả Alice và Bob *chấp nhận* nhận dạng của phía bên kia. Nếu việc xác thực bao gồm cả trao đổi khóa thì cả hai cũng *chấp nhận* khóa đã được trao đổi.

Tính chất thứ hai mà vận hành thành công có liên quan là nhật ký của việc vận hành thủ tục (giả thiết các bên tham gia có ghi lại việc trao đổi). Để tiếp tục, chúng ta cần các định nghĩa để ý đến việc *trùng nhau (match)* khi áp dụng vào các nhật ký của việc vận hành (khác một chút so với định nghĩa được đưa ra bởi Bird [5]).

Việc trùng các thông điệp: Chúng ta nói một thông điệp từ một nhật ký trùng với một thông điệp từ một nhật ký khác nếu một nhật ký ghi thông điệp như là tin đến, còn một nhật ký ghi thông điệp như là tin đi, và tất cả các trường có liên quan đến việc xác thực là như nhau trong cả hai nhật ký.

Việc định tính chất có liên quan đến xác thực cần phải cho phép các thông điệp có thể trùng nhau ngay cả khi chúng không trùng nhau từng bit. Qui cách ở đây là nếu thông điệp có chứa các trường không được ký, tức là về mặt mật mã không liên quan đến việc xác thực, thì sự khác biệt chỉ ở các trường này không làm ảnh hưởng việc làm cho thông điệp đáp ứng được định nghĩa của việc trùng.

Việc trùng nhật ký vận hành: Chúng ta nói rằng hai nhật ký của việc vận hành là trùng nhau nếu các thông điệp của chúng có thể phân chia thành tập của các thông điệp trùng (mỗi tập chứa 1 thông điệp từ mỗi nhật ký), các thông điệp xuất pháp từ một bên tham gia xuất hiện với cùng một thứ tự trong cả hai nhật ký, các thông điệp xuất phát từ bên tham gia còn lại cũng vậy. Để cho đơn giản, chúng ta không xem xét các thủ tục mà trong đó các thông điệp không đến đích theo thứ tự mà chúng đã được gửi đi.

Chú ý rằng các thông điệp xuất phát từ các bên tham gia khác nhau không nhất thiết phải có cùng thứ tự đối với nhau. Điều này cho phép trường hợp khi có thông điệp đi ngang qua. Trong trường hợp này, mỗi bên tham gia sẽ ghi nhật ký thông điệp của mình như là được gửi đi trước khi nhận các thông điệp đi qua.

Tính chất 2 của vận hành thành công: Nếu Alice và Bob cùng ghi lại quá trình trao đổi, thì nhật ký vận hành của họ trùng nhau.

Bây giờ chúng ta phân biệt giữa vận hành thành công và vận hành an toàn. Để xem xét một vận hành thành công theo bất kỳ một định nghĩa hợp lý nào, vận hành phải được xem là an toàn theo nghĩa trực giác. Mặt khác, có thể một vận hành là không thành công ngay cả khi không xảy ra sự vi phạm về độ an toàn (ví dụ, cả hai bên cùng bác bỏ vì một lý do nào đó). Cũng có thể là kẻ thù địch giữ chậm lại thông điệp hợp pháp của vận hành một cách vô thời hạn. Giả sử rằng trong một lần vận hành cụ thể, Alice chấp nhận nhận dạng của Bob, gửi thông điệp cuối cùng trong thủ tục tới Bob, nhưng kẻ thù địch huỷ bỏ mất thông điệp này. Giả thiết rằng Bob cần phải nhận được thông điệp này trước khi nhận được nhận dạng của Alice, như vậy Bob sẽ không chấp nhận nhận dạng của Alice. Một cách trực giá, mặc dù cuộc vận hành này không thành công, nhưng tại đây không có sự vi phạm về độ an toàn; tại thời điểm Alice nhận được nhận dạng của Bob (trước khi cô ta gửi thông điệp cuối cùng), nhật ký vận hành của Bob trùng với nhật ký của Alice. Theo mục đích của chúng ta, tấn công như *từ chối dịch vụ* không được xem xét là sự vi phạm độ an toàn; vấn đề này thường được giải quyết bằng biện pháp an toàn vật lý hay các kỹ thuật khác.

Bây giờ đã đến lúc chúng ta định nghĩa xem cái gì là sự vận hành của một thủ tục xác thực đa phương (đối xứng hay phi đối xứng) không an toàn.

ĐỊNH NGHĨA 1: Một vận hành cụ thể của thủ tục được gọi là vận hành *không an toàn* nếu bất kỳ một bên tham gia nào trong vận hành, ví dụ như Alice, thực hiện thủ tục một cách trung thực, chấp nhận nhận dạng của phía bên kia, và một trong các điều kiện sau xảy ra:

- Tại thời điểm Alice chấp nhận nhận dạng của phía bên kia (trước khi cô ấy gửi hay nhận thông điệp tiếp theo), một phần hay toàn bộ nhật ký của phía bên kia không trùng với nhật ký của Alice.
- Khóa trao đổi được chấp nhận bởi Alice bị một một người khác biết, đây không phải là người mà Alice đã chấp nhận nhận dạng. (Điều kiện này không áp dụng cho việc xác thực không có trao đổi khóa).

Chú ý rằng theo điều kiện này thì thủ tục trao đổi khóa kinh điển đòi hỏi có một đối tượng tin cậy thứ ba [18] sẽ là không an toàn.

Rõ ràng là nhật ký của Alice (cần phải trùng với nhật ký của phía bên kia theo định nghĩa trên) chính là nhật ký mà cô ấy có tại thời điểm mà cô ấy nhận đủ thông tin để thực hiện một tính toán bất kỳ cần thiết để đạt được trạng thái chấp nhận; các thông điệp gửi hay nhận sau đó sẽ không liên quan đến.

Mục đích của kẻ thù địch là làm cho việc vận hành trở nên không an toàn. Mục đích của người thiết kế thủ tục là làm cho công việc của kẻ địch trở nên không thể (hay là không thể về mặt tính toán) trong mọi thể hiện của thủ tục (trong

mọi lần vận hành). Ngược lại với Định nghĩa 1, chúng ta có định nghĩa của thủ tục xác thực đa phương an toàn (đối xứng hay phi đối xứng):

ĐỊNH NGHĨA 2: *Thủ tục an toàn* là thủ tục mà đối với chúng các điều kiện sau là đúng trong mọi trường hợp, khi mà một bên, ví dụ Alice, thực hiện thủ tục một cách trung thực và chấp nhận nhận dạng của phía bên kia:

- Tại thời điểm Alice chấp nhận nhận dạng của phía bên kia (trước khi cô ta gửi hay nhận thông điệp tiếp theo), một phần hay toàn bộ nhật ký vận hành của phía bên kia trùng với nhật ký vận hành của Alice.
- Đối với khóa trao đổi nếu được chấp nhận bởi Alice thì không thể về mặt tính toán một ai đó có thể tìm ra, ngoại trừ chính Alice và có thể là đối tác mà Alice đã chấp nhận nhận diện. (Điều kiện này không có đối với việc xác thực không có trao đổi khóa).

Chỉ bằng chính mình, các định nghĩa trên không giúp đỡ gì trong việc quyết định xem một thủ tục đã cho có an toàn hay không, bởi vì nó không đưa đến một quá trình kiến thiết cho phép hoặc kiểm chứng hoặc chỉ ra điểm yếu của thủ tục. Dù sao, định nghĩa này có thể áp dụng trực tiếp để quyết định xem một tấn công tiềm năng đã có có là một tấn công thực hay không. Ví dụ, trong việc xác thực có trao đổi khóa, giả sử kẻ thù địch chỉ tóm bắt các thông điệp của Alice và Bob sau đó lại thả cho chúng đi tiếp mà không thay đổi gì. Một cách trực quan, kẻ thù địch không làm tổn hại hệ thống trong trường hợp này; các bên tham gia chấp nhận nhận dạng của phía bên kia, có nhật ký vận hành trùng nhau và có khóa chia sẻ khóa bí mật chung. Chú ý rằng theo Định nghĩa 1, một vận hành như vậy là không an toàn. Trong các trường hợp khác, tấn công được đề nghị có thể trở nên hoàn toàn phức tạp, và có thể không rõ ràng rằng tấn công chẳng có gì khác ngoài việc cho đi qua các thông điệp. Định nghĩa 1 có thể được sử dụng để phân biệt việc cho đi qua như vậy không là tấn công. Trong Mục 5, định nghĩa này phục vụ tốt trong việc nhận dạng các tấn công thực sự; đặc biệt, điều kiện thứ hai, tưởng chừng như tầm thường, là cần thiết.

Trong khi các kỹ thuật phân tích hình thức có thể sử dụng thành công để khám phá các điểm yếu trong một số thủ tục xác thực (xem Mục 6), chứng minh tính đúng đắn là khó hơn nhiều, và phụ thuộc nhiều vào việc mô hình hóa đúng mục đích và các giả thuyết. Một kỹ thuật khác có thể dùng để khám phá điểm yếu là việc vét cạn đối với tấn công chèn (interleaving attack) [5]. Thật đáng tiếc, vì không có được chứng minh tuyệt đối về tính đúng đắn, niềm tin vào thủ tục chỉ có được khi các chuyên gia liên tục tiến hành phân tích nó và thất bại trong việc tìm ra điểm yếu.

4. Các đặc trưng mà thủ tục cần có

Bên cạnh tính an toàn, sau đây là các đặt tính khác cần có cho một thủ tục.

Độ an toàn chuyển tiếp hoàn thiện (Perfect Forward Secrecy-PFS). Thủ tục trao đổi khóa có xác thực cung cấp độ an toàn chuyển tiếp hoàn thiện nếu việc để lộ tài liệu khóa bí mật thời hạn dài (long-term secret key material) không làm tổn hại đến độ an toàn của những khóa được trao đổi trong những lần vận hành trước. Tính chất độ an toàn chuyển tiếp hoàn thiện không áp dụng cho việc xác thực không có trao đổi khóa.

Xác thực trực tiếp. Trong một số thủ tục trao đổi khóa, việc xác thực không được thực hiện cho đến khi cả hai bên chứng minh tri thức về khóa chung bằng cách sử dụng nó trong liên lạc sau đó. Thủ tục như vậy gọi là *không trực tiếp (indirect)*. Nếu việc xác thực được thiết lập ngay sau khi kết thúc việc vận hành thủ tục thì thủ tục được gọi là *trực tiếp*. Một thủ tục gián tiếp có thể được sửa đổi để trở thành trực tiếp bằng cách thêm việc trao đổi một thông điệp đã biết hay thông điệp có độ dư được mã bằng khóa đã trao đổi. Với xác thực không có trao đổi khóa, thủ tục gián tiếp không cung cấp tính bảo mật vì không có bên nào có thể chấp nhận nhận dạng của bên kia.

Không có tem thời gian. Trong khi tem thời gian là quen thuộc cho mục đích quản lý và tài liệu hóa, trong thực hành cần phải không dựa vào việc sử dụng nó để đảm bảo độ mật của thủ tục xác thực. Những trở ngại, cảnh báo và việc chống lại tem thời gian được trình bày ở [3], [5], [13]. Để cho tiện, sau đây chúng tôi tóm tắt một số vấn đề chính.

Để sử dụng tem thời gian cho xác thực, tất cả các bên tham gia đều phải duy trì đồng hồ của mình được đồng bộ thường xuyên một cách an toàn với nguồn thời gian tin cậy. Giữa những lần đồng bộ với nguồn thời gian tin cậy, thời gian ở từng trạm có thể khác nhau. Hai bên, Alice và Bob, cần cho phép một cửa sổ thời gian cho tem thời gian để bù lại những chênh lệch đồng hồ cục bộ và việc thông điệp đi trên mạng cũng tốn thời gian. Alice sẽ chấp nhận tem thời gian từ Bob nếu nó nằm trong cửa sổ xung quanh thời gian tại đồng hồ cục bộ của Alice và thêm nữa là Bob phải chưa sử dụng giá trị thời gian này trước đây. Alice có thể lưu trữ tất cả giá trị thời gian được sử dụng bởi tất cả các bên mà nằm trong cửa sổ hiện nay của cô ta (điều này có thể là không thực tế trong một số môi trường truyền thông) hoặc Alice có thể lưu trữ thời điểm cuối cùng được sử dụng bởi mỗi bên và kiên quyết đòi hỏi tăng giá trị thời gian từ mỗi bên tham gia. Tuy nhiên, trong trường hợp giá trị thời gian tăng thực sự, nếu Bob sử dụng thời điểm t ở tương lai xa vì một lý do nào đó (sự chênh lệch đồng hồ lớn hoặc đồng bộ sai với nguồn thời gian tin cậy), thì Bob không thể liên lạc với Alice cho đến khi thời điểm t nằm trong cửa sổ của Alice. Để tránh điều này, Alice cần phải lưu trữ thời điểm t và không cập nhật nhật ký của mình đối với giá trị thời gian cuối cùng được sử dụng bởi Bob. Điều này có thể dẫn đến việc phải chọn số lượng lớn các dữ liệu được lưu trữ, hy sinh khả năng truyền thông hoặc hy sinh độ an toàn. Liên quan đến khả năng truyền thông, nếu đồng hồ cục bộ của 2 bên rất không đồng bộ với nhau thì

hai bên không thể liên lạc. Điều này làm cho những ai lo lắng tới khả năng truyền thông muốn cửa sổ thời gian rộng, như vậy làm tăng yêu cầu lưu trữ. Trong khi tem thời gian được coi là tiện lợi từ quan điểm lý thuyết, nó làm nảy sinh nhiều vấn đề thực tế. Những thủ tục dựa trên các thách thức ngẫu nhiên không gặp phải những khó khăn ấy.

Gần đây, các phân tích hình thức đã được sử dụng trong việc kiểm tra các thủ tục xác thực [7], [13]. Bắt đầu bằng danh sách các niềm tin hình thức khởi đầu, mục tiêu là bằng một cách logic dẫn đến mục đích của thủ tục đã được phát biểu bằng cách áp dụng danh sách các bước của thủ tục. Một trong những giả thiết cơ sở mà phân tích như vậy thường dựa vào là các bên tham gia có khả năng kiểm tra tính được làm tươi (freshness) của tem thời gian. Thực ra, một trong những kết quả chính của công trình do Gaarder và Snekkenes thực hiện cho thấy có sự gắn bó chặt chẽ của yêu cầu độ an toàn với việc đồng hồ phải được tin cậy trong một thủ tục nào đó. Điều này có nghĩa là trong thực tế, độ an toàn của thủ tục dựa vào tem thời gian phụ thuộc mạnh vào việc thi hành đúng đắn các đồng hồ được đồng bộ và bảo vệ an toàn. Không may, mặc dù đã có nhiều tài liệu bàn về các thủ tục dựa vào tem thời gian (ví dụ [8], [16]), khi áp dụng thực tế các thủ tục này cho thấy, việc đảm bảo an ninh cho các đồng hồ dễ bị vi phạm và hơn nữa, giá thành chi cho việc thi hành đúng đắn cũng đáng kể.

5. Thủ tục trạm-tới-trạm

Bây giờ, chúng tôi giới thiệu một thủ tục trao đổi khóa xác thực hiệu quả, đơn giản có tên gọi là trạm-tới-trạm (STS). Thủ tục STS được phát triển theo thời gian, phiên bản trước đây của thủ tục được mô tả trong hội nghị quốc tế về chuyển mạch năm 1987 [21]. Chúng ta tin vào tính an toàn của nó theo như Định nghĩa 2 và có một số tính chất mong muốn khác. Trong phần còn lại của mục này, chúng ta mô tả thủ tục, bàn về tính chất của nó, mổ xẻ những chi tiết tế nhị bằng cách chỉ ra những thay đổi làm cho nó bị tổn thương.

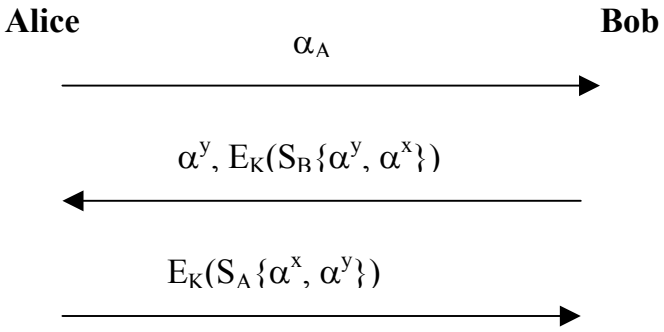
5.1 Thủ tục STS cơ sở

Thủ tục STS chứa việc thiết lập khóa Diffie-Hellman [9], kèm theo việc trao đổi chữ ký xác thực. Trong phiên bản cơ sở của thủ tục, chúng ta giả thiết rằng các tham số được sử dụng để thiết lập khóa (tức là các đặc tả của nhóm cyclic đặc biệt và phần tử nguyên thủy tương ứng α) là cố định và mọi người đều biết. Trong khi chúng ta nói tới phép toán Difie-Hellman như là phép lũy thừa, suy ra rằng nhóm cơ sở là nhóm nhân, nhưng mô tả cũng được áp dụng tốt cho nhóm cộng (tức là nhóm các điểm của đường cong elliptic trên trường hữu hạn). Chúng ta cũng giả thiết trong mục này rằng Alice biết khóa công khai đích thực của Bob và ngược lại; giả thiết này được bỏ trong mục sau.

Thủ tục được bắt đầu bởi một bên, ví dụ, Alice, tạo ra một số ngẫu nhiên x và gửi lũy thừa α^x sang bên kia cho Bob. Bob tạo số ngẫu nhiên y và sử dụng lũy thừa của Alice để tạo ra khóa trao đổi $K=\alpha^{xy}$. Bob phúc đáp cùng với lũy thừa α^y

và một thẻ chứa bản mã của chữ ký của anh ta lên các lũy thừa bởi khóa K và thuật toán mã đối xứng thích hợp E (tức là $E_K(s_B\{\alpha^y , \alpha^x \})$). Alice tính khóa K, giải mã thẻ bởi K, kiểm tra chữ ký của Bob lên các lũy thừa bởi khóa công khai của Bob. Alice gửi cho Bob bản mã của chữ ký của cô ta lên các lũy thừa, tức là $E_K(s_A\{\alpha^x , \alpha^y \})$. Cuối cùng, tương tự, Bob kiểm tra chữ ký của đã được mã hóa của Alice bằng cách dùng khóa K và khóa công khai của Alice. Độ an toàn của trao đổi khóa lũy thừa dựa vào tính không phá được của bài toán logarit rời rạc [24].

Thủ tục STS cơ sở



Có thể tạo ra một phiên bản đối xứng hơn cho thủ tục này, trong đó các bên trao đổi các lũy thừa trước và sau đó trao đổi bản mã của các chữ ký lên các thông điệp riêng biệt. Trong trường hợp này, cả Alice và Bob không cần biết ai bắt đầu cuộc gọi. Điều này là mong muốn, giống như tình huống trong thực tế (cả trên đường điện thoại cũng như truyền dữ liệu X.25) trong đó ở một mức độ cài đặt nào đấy, không rõ là bên nào bắt đầu trước. Điều này giải thích tại sao trong khuôn dạng chữ ký của mỗi bên thì lũy thừa của chính người đó được đưa lên trước. Nếu các lũy thừa có cùng thứ tự trong cả hai chữ ký thì Alice và Bob cần tìm cách thỏa thuận xem lũy thừa của ai được đưa lên trước (ví dụ như dựa vào quy định là xem bên nào bắt đầu cuộc gọi).

Bây giờ, hãy xem xét độ đảm bảo mà thủ tục STS cung cấp cho mỗi bên. Từ quan điểm của Bob, như là kết quả của việc trao đổi khóa Diffie-Hellman, anh ta chia sẻ khóa chung với một người (có thể không phải là Alice). Giả thiết của chúng ta trong mục này là Bob biết khóa công khai của Alice (điều này đạt được trong mục sau thông qua việc sử dụng chứng thực). Vì Alice có ký vào các lũy thừa đặc biệt tương ứng với lần vận hành này, một trong số đó do Bob tự tạo ra chỉ cho lần vận hành ấy, nên chữ ký của Alice được gắn chặt với lần vận hành này. Bằng cách mã chữ ký của Alice bởi khóa K, Alice trình diễn cho Bob rằng Alice chính là người đã tạo ra x. Điều này làm cho Bob tin rằng người mà Bob trao đổi khóa với chính là Alice. Alice cũng có niềm tin tương tự đối với Bob.

Thủ tục STS có các tính chất mong muốn như đã bàn ở mục 4. Thay cho việc sử dụng tem thời gian, các thách thức đã được sử dụng. Bởi vì các bên tham

gia trình diễn kiến thức về khóa được trao đổi bằng cách mã chữ ký của họ nên việc xác thực là trực tiếp. Hơn nữa, thủ tục STS còn cung cấp tính an toàn chuyển tiếp hoàn thiện. Tư liệu khóa bí mật dài hạn duy nhất được lưu trữ bởi người sử dụng là khóa bí mật của họ dùng để sinh chữ ký số. Nếu khóa bí mật bị lộ, thì độ an toàn của những khóa trao đổi từ những vận hành trước đó không bị ảnh hưởng bởi vì trao đổi khóa Diffie-Hellman được sử dụng; trao đổi khóa Diffie-Hellman không có tư liệu khóa thời hạn dài. Ngoài ra, thủ tục STS còn 2 tính chất mong muốn nữa. Thứ nhất là kỹ thuật khóa công khai được sử dụng để quản trị khóa đơn giản hơn và an toàn hơn so với khả năng dùng mật mã cổ điển. Nếu các bên tham gia tự sinh ra khóa bí mật của mình, các khóa này không bao giờ được để lộ (với bất cứ ai, kể cả với đối tác mà mình tin cậy), thậm chí ngay trong quá trình khởi tạo. Thứ hai là không cần thiết việc các đối tác liên lạc phải liên hệ với thiết bị trung tâm trên cơ sở cuộc gọi. Nếu các chứng thực được sử dụng để phân phối khóa công khai (xem Mục 5.2), một khi ai đó có chứng thực của mình và khóa công khai của người có thẩm quyền được tin cậy, thì anh ta có thể trao đổi khóa với và xác thực các đối tác khác mà không cần tham khảo thiết bị trung tâm. Thủ tục này xuất hiện đã gây được tiếng vang về tính cân đối, tao nhã, đơn giản, an toàn và không có các phần tử thừa hay không cần thiết.

Để minh họa tính cần thiết cho các đại lượng của thủ tục STS, bây giờ chúng ta trình diễn xem thủ tục yếu đi như thế nào khi những thay đổi sau được làm: bỏ việc mã chữ ký số, chỉ ký vào lũy thừa của riêng mình, chỉ ký vào lũy thừa của đối tác, tháo bỏ việc xác thực khỏi trao đổi khóa.

Bỏ việc mã chữ ký số

Xem xét thủ tục STS có sửa đổi, chữ ký số lên các lũy thừa không được mã bằng khóa trao đổi K nữa. Bởi vì các lũy thừa là các thông tin công khai, bất kỳ một ai đó cũng có thể ký nó. Giả sử rằng trong thông điệp cuối cùng của thủ tục, Eve thay chữ ký Alice lên các lũy thừa bằng chữ ký của mình. (Nếu các bên tham gia trao đổi khóa công khai có chứng thực, Eve có thể thay chứng thực của Alice bằng chứng thực của mình.) Đây không phải là một tấn công nguy hiểm vì thực ra Eve không biết khóa được trao đổi. Tuy nhiên, nếu Bob làm việc tại ngân hàng, Eve có thể rút tiền ở tài khoản của Alice. Thật thú vị, mặc dù Bob bị nhầm nhưng Alice lại có thể là bên bị hại.

Một cách không hình thức, chúng ta đã chỉ ra tại sao vận hành trên là không an toàn, bây giờ ta áp dụng Định nghĩa 1. Bob thực hiện thủ tục một cách chân thực và chấp nhận nhận dạng của Eve, nhưng khóa trao đổi được biết bởi một người khác, Alice. Theo Định nghĩa 1, vận hành như vậy là không an toàn. Bởi vì có vận hành không an toàn nên thủ tục có sửa đổi là không an toàn.

Chỉ ký vào lũy thừa của riêng mình

Xem xét một phương án của thủ tục STS khi mỗi bên tham gia chỉ ký vào lũy thừa của mình (có nghĩa là bản mã chữ ký của Alice là $E_K(s_A\{\alpha^x\})$ và của Bob là $E_K(s_B\{\alpha^y\})$. Chúng ta biết rằng trong trường hợp này không có một tấn công tổng

quát, nhưng khi lược đồ chữ ký số là RSA [26] thì tấn công áp dụng được, *hàm băm là hàm đồng nhất* và trao đổi khóa Diffie-Hellman được thực hiện trên trường GF(p). Trong trường hợp này, Eve có thể giả mạo Alice khi vận hành thủ tục với Bob bằng cách dùng x = 0 như là lũy thừa trong khi trao đổi khóa. Hàm mũ của Eve là $\alpha^0 = 1$, và khóa trao đổi là $K = \alpha^{xy} = 1$. Eve cần bản mã chữ ký sau:

$$\begin{aligned} E_K(s_A\{\alpha^x\}) &= E_1(s_A\{1\}) \\ &= E_1(s_A(1)) \quad \text{vì hàm băm là hàm đồng nhất} \\ &= E_1(1) \quad \text{vì chữ ký trong RSA là hàm mũ và } 1^z = 1 \text{ với mọi z} \end{aligned}$$

Eve có thể tính được $E_1(1)$ nên có thể giả được Alice. Mặc dù tấn công này chỉ có trong trường hợp đặc biệt, nó minh họa một vấn đề lớn hơn khi chỉ ký vào một lũy thừa của mình: nếu Eve có thể nhận được đại lượng mà cô ta có thể kiếm được hoặc tính được logarit rời rạc, và có thể kiếm được hoặc tính được chữ ký của Alice lên đại lượng đó, thì Eve có thể sử dụng (sử dụng lại) đại lượng này như một lũy thừa để đóng giả Alice. Bằng cách đưa vào lũy thừa thứ hai trong dữ liệu được ký, kẻ tấn công buộc phải giải một phương án khác của bài toán cần giải trong thời gian thực mỗi khi có ý định đóng giả.

Chỉ ký vào lũy thừa của đối tác

Hãy xem xét phương án của thủ tục STS trong đó mỗi bên tham gia chỉ ký vào lũy thừa của người kia (có nghĩa là bản mã chữ ký của Alice là $E_K(s_A\{\alpha^y\})$ và của Bob là $E_K(s_B\{\alpha^x\})$. Một lần nữa, chúng ta biết rằng không có một tấn công tổng thể được áp dụng cho trường hợp này, nhưng có một số điều cần quan tâm.

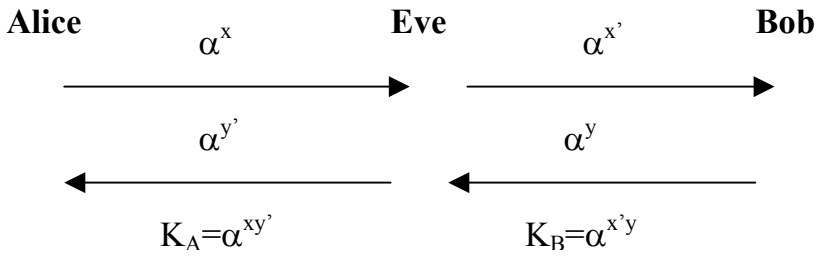
Về nguyên tắc, sẽ là không cẩn thận nếu ký vào đoạn text bất kỳ được cung cấp bởi kẻ thù địch tiềm tàng. Trong trường hợp cụ thể này, để cho kẻ thù địch khôi phục được chữ ký, anh ta cần phải biết khóa K. Để tính K, kẻ thù địch cần biết logarit rời rạc của đại lượng được ký. Trong khi kẻ thù địch nhìn chung là không biết logarit rời rạc của một đại lượng cố định cụ thể nào đó mà anh ta muốn ký, việc sinh ra các đại lượng như vậy bằng cách chọn trước các logarit là tầm thường, do đó điều rắc rối xuất hiện khi cho phép kẻ thù địch được tự do đòi hỏi chữ ký lên bất kỳ một đại lượng nào mà logarit của nó đã biết.

Trong mục 5.3, sẽ chỉ ra rằng thủ tục STS có thể được rút gọn đến thủ tục chỉ xác thực bằng cách thay các lũy thừa bằng các số ngẫu nhiên và bỏ việc mã các chữ ký. Nếu các bên chỉ ký vào lũy thừa của phía bên kia, thì thủ tục chỉ xác thực sẽ là mục tiêu cho tấn công đối với cặp thách thức-trả lời đơn giản như đã chỉ ra ở mục 2. Tương tự, nếu chỉ ký lên lũy thừa của chính mình cũng không đưa đến một thủ tục mà có thể được rút gọn đến phương án chỉ xác thực an toàn.

Chú ý rằng khi ký cả hai lũy thừa thì công sức bỏ ra thêm chỉ là việc tính thêm giá trị băm, nói chung nó là tương đối nhỏ. Không có thêm các phép toán bao gồm lược đồ chữ ký, các thao tác mã hóa đối xứng hay truyền dữ liệu được đưa vào bởi việc ký cả hai lũy thừa, đúng hơn là chỉ thêm có một.

Tháo bỏ việc xác thực khỏi trao đổi khóa

Nếu thủ tục STS được thay đổi sao cho việc xác thực được bỏ khỏi việc trao đổi khóa bằng cách bắt các bên tham gia ký một đại lượng nào đó độc lập với các lũy thừa, thủ tục thu được sẽ là mục tiêu cho tấn công cổ điển người xâm nhập ở giữa (xem [27]) đối với trao đổi khóa Diffie-Hellman.



Eve thay lũy thừa của mình vào chỗ lũy thừa của Alice và Bob. Kết quả là Alice và Bob tính ra hai khóa khác nhau, nhưng cả 2 khóa này Eve đều tính được. Eve dùng chung khóa K_A với Alive và dùng chung khóa K_B với Bob. Trong giai đoạn xác thực khi vận hành, Eve có thể chuyển thông điệp được mã của Alice đến Bob và ngược lại bằng cách giải mã bằng một khóa và mã lại bằng khóa khác. Sau khi xác thực, Eve tự do chặn bắt thụ động hoặc chèn thông điệp riêng của mình. Theo Định nghĩa 1, thủ tục được thay đổi là không an toàn bởi vì trong khi Alice thực hiện thủ tục một cách trung thành và chấp nhận nhận dạng của Bob, thì khóa trao đổi được dùng chung bởi một bên khác là Eve. Vấn đề tương tự cũng có từ cách nhìn của Bob.

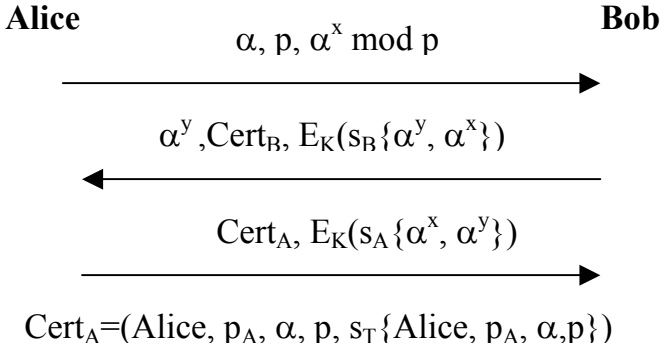
5.2 Thủ tục STS trong thực tế

Bây giờ chúng ta mô tả việc sử dụng thủ tục STS trong thực tế, cho trường hợp đặc biệt khi việc trao đổi khóa được thực hiện trong nhóm nhân của trường hữu hạn. Để cho rõ ràng, chúng ta tập trung vào trường nguyên tố GF(p). Hai tham số được yêu cầu cho trao đổi khóa Diffie-Hellman là: phần tử nguyên thủy α trong GF(p) và số nguyên tố thích hợp p. Số nguyên tố p cần phải chọn sao cho loại bỏ được các tấn công kiểu Pohlig-Hellman [25]. Trong ánh sáng của những kết quả mới đây về bài toán logarit rời rạc ([24] cho các trường nguyên tố; [23] cho các trường có đặc trưng 2), nên cẩn thận sử dụng trường khác nhau cho mỗi người sử dụng (có nghĩa là GF(p), với p khác nhau, được chọn bởi chính người dùng). Tấn công tốt nhất đối với trao đổi khóa Diffie-Hellman trên trường hữu hạn là kỹ thuật tính chỉ số bao gồm một khối lượng tính toán trước khổng lồ đưa đến một cơ sở dữ liệu cho từng trường. Cơ sở dữ liệu sau đó cho phép tính riêng từng logarit trong trường đó tương đối nhanh. Nếu một trường duy nhất được sử dụng trên toàn mạng, một cơ sở dữ liệu duy nhất cho phép tấn công tất cả các cuộc trao đổi khóa-điều này càng khích lệ cố gắng xây dựng cơ sở dữ liệu.

Để hỗ trợ việc phân phối khóa công khai của người sử dụng và các tham số Diffie-Hellman theo từng người sử dụng, có thể sử dụng các chứng thực. Thêm

vào các đại lượng này, chứng thực nên chứa tên người sử dụng và chữ ký của người có thẩm quyền được tin cậy lên các dữ liệu này. Nguyên nhân đưa cặp (α, p) vào chứng thực được giải thích phía sau. Thủ tục STS sẽ hoạt động như sau. Để tránh làm công thức phức tạp, việc tính mod p được bỏ qua.

Thủ tục STS trong thực tế



Sự khác biệt là như sau. Alice gửi các tham số Diffie-Hellman cùng trong thông điệp thứ nhất; Bob sử dụng chúng thay cho các tham số cố định của toàn mạng. Sau khi nhận được thông điệp thứ ba, Bob kiểm tra các tham số Diffie-Hellman được gửi trong thông điệp thứ nhất khớp với những cái thực ra có trong chứng thực của Alice. Trong thông điệp thứ hai, Bob gửi cho Alice chứng thực của Bob, từ đó Alice có thể trích ra khóa công khai đích thực của Bob; Alice kiểm tra tính xác thực bằng cách kiểm tra chữ ký của người có thẩm quyền tin cậy lên chứng thực của Bob. Tương tự, trong thông điệp thứ ba, Alice gửi cho Bob chứng thực của Alice, cho phép Bob trích ra khóa công khai được xác thực của Alice, sau đó tương tự kiểm tra chữ ký của của người có thẩm quyền tin cậy lên chứng thực của Alice. Chú ý rằng Bob không cần đến chứng thực của Alice cho đến bước thứ 3, và trong thực tế không muốn nhận được nó sớm hơn, bởi vì điều này đòi hỏi phải có chỗ lưu trữ để ghi lại chứng thực cho đến khi nó được dùng đến. Nguyên nhân tiếp theo để Alice làm chậm việc chuyển chứng thực của mình cho đến thông điệp thứ ba là điều này cho phép cả Alice và Bob khả năng mã chứng thực của họ bằng khóa trao đổi. Mặc dù, về mặt lý thuyết, các chứng thực là thông tin công khai, nhưng mong muốn trong một số ứng dụng cần ngăn cản việc xem chúng bởi người chặn bắt, từ đó ngăn cản người chặn bắt thụ động nghiên cứu về nhận dạng của Alice và Bob.

Chú ý rằng kiến thức về khóa công khai của đối tác là không yêu cầu để kiến thiết và gửi hay nhận và xử lý thông điệp đầu tiên. Nếu khóa công khai được yêu cầu tại giai đoạn này thì việc đưa vào các chứng thực sẽ kéo theo thông điệp thêm trước đó để làm cho các chứng thực là có thể sớm hơn.

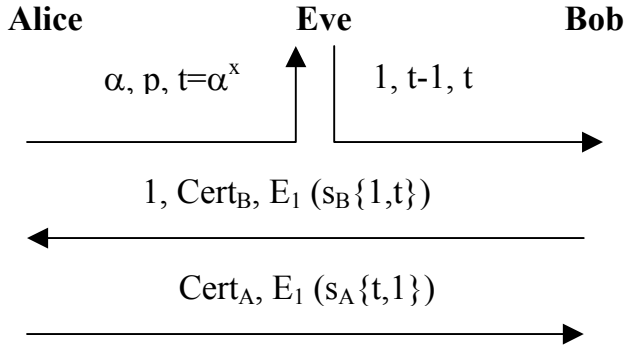
Như đã bàn luận ở mục 5.1, trong một số trường hợp có thể mong muốn

rằng cả hai bên đều gửi thông điệp khởi đầu cùng một lúc. Trong trường hợp này, một số phương pháp có thể được sử dụng để thiết lập một trong các bên là thành viên trội (dominant party) (có nghĩa là, đối tác có cặp (α,p) được sử dụng). Thành viên không trội tiếp tục thủ tục với thông điệp thứ hai. Một ví dụ đơn giản là chọn đối tác có p lớn hơn là đối tác trội.

Bây giờ chúng ta chỉ ra rằng thủ tục sẽ là yếu nếu các tham số Diffie-Hellman không được đưa vào chứng thực.

Bỏ các tham số Difie-Hellman trong chứng thực

Không có các tham số Diffie-Hellman trong chứng thực, kẻ thù địch, Eve, có thể tự do thay đổi α và p trong thông điệp đầu tiên của Alice. Giả sử lũy thừa của Alice là $t = (\alpha^x \bmod p)$. Giả thiết rằng Eve thay đổi α bằng 1 và p bằng t-1 (xem sơ đồ dưới đây). Thế thì lũy thừa của Bob là $1^y \bmod (t-1)=1$, và Bob tính khóa trao đổi sẽ là $t^y \bmod(t-1)=1$. Alice tính khóa trao đổi là $1^x \bmod p = 1$. Bởi vì Eve không thay đổi các lũy thừa được trao đổi và Alice cũng như Bob cùng tính ra một khóa chung, nên Alice và Bob sẽ chấp nhận chữ ký đã được mã hóa của nhau.



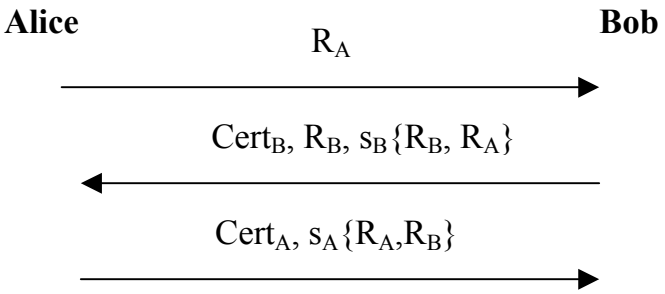
Eve biết khóa trao đổi và sau khi xác thực, cô ta được tự do chặn bắt và chèn thông điệp của riêng mình. Chú ý rằng Alice và Bob chấp nhận nhận dạng của nhau, nhưng nhật ký vận hành của họ không trùng nhau, và khóa trao đổi được biết bởi bên thứ 3, thủ tục trên là không an toàn theo định nghĩa của chúng ta cũng như theo trực quan.

Trong khi có thể thấy rằng phép thế được mô tả ở trên là tầm thường và dễ phát hiện bởi việc kiểm tra đặc biệt, tiềm năng bị tổn thương vẫn còn. Các tấn công tinh vi hơn hay các tấn công cải trang liên quan là có thể, bao gồm khả năng sử dụng các số nguyên tố yếu theo Pohlig-Hellman. Mối quan tâm nền tảng là để cho tin được vào tính không phá được của bài toán Diffie-Hellman, cần phải tin chắc rằng các tham số Diffie-Hellman thích hợp được sử dụng trong thực tế.

5.3 Phiên bản chỉ xác thực của thủ tục STS

Có thể biến thủ tục STS thành một thủ tục chỉ có xác thực bằng cách thay các lũy thừa với các số ngẫu nhiên và bỏ phép mã trên chữ ký số.

Thủ tục STS chỉ có xác thực



Thủ tục được đơn giản hóa này về thực chất chính là thủ tục xác thực 3 lần tương tác được đề xuất gần đây bởi ISO [1]. Cái này sẽ được bàn tiếp ở mục sau.

6. Bàn luận về các thủ tục khác

Tấn công người xâm nhập đứng giữa đối với trao đổi khóa Diffie-Hellman không xác thực về thực chất là nhại lại ý tưởng của bài toán "kiện tướng cờ qua thư" đã quen biết[2], các tấn công khác đối với các thủ tục xác thực là rất nhiều và được trình bày trong các tài liệu khác. Burrows, Abadi và Needham đã phân tích 8 thủ tục và tìm ra 6 trong chúng có dư thừa, 4 trong chúng có điểm yếu [7, bảng 1], bao gồm cả dư thừa và điểm yếu trong các thủ tục xác thực đa phương CCITT X.509 [30]. Để thấy được mối quan tâm đối với nhiều thủ tục được đề xuất gần đây, chúng tôi bàn luận ngắn gọn về 4 thủ tục đã được phân tích bởi Burrows: Kerberos, một trong các thủ tục X.509. Chúng tôi cũng bàn về thủ tục ISO có liên quan.

Thủ tục Kerberos. Thủ tục Kerberos quen thuộc [18], dựa trên hệ mã đối xứng có một số đặc tính làm cho nó không thích hợp trong nhiều ứng dụng. Những cái đó bao gồm việc sử dụng tem thời gian (đã được giải thích ở trên), yêu cầu về máy chủ xác thực trực tuyến, độ dư thừa trong thủ tục. Những vấn đề này và một số vấn đề khác nữa đã được Bellovin và Merritt bàn đến [3].

Thủ tục xác thực X.509 ba lần tương tác. Khuyến nghị CCITT X.509 [30] là thủ tục xác thực được chuẩn hóa quốc tế và được nhiều người biết đến dựa trên mật mã khóa công khai. Các thủ tục X.509 một hay hai lần tương tác cần có tem thời gian, còn tem thời gian là dư thừa trong thủ tục 3 lần tương tác; đặc tả cho phép trường tem thời gian có thể bằng không đối với thủ tục 3 lần tương tác (làm cho

[2] Người mới chơi cờ trong 2 ván cờ chơi đồng thời với 2 kiện tướng cờ, chới quan trắng ở ván này và quân đen ở ván kia, anh ta lấy nước đi của đối thủ trong mỗi ván và sử dụng chúng ở ván kia, việc này đảm bảo cho anh ta thắng cả hai hoặc thắng 1và thua 1, như vậy hạng thứ về cờ của anh ta sẽ được tăng lên tuy rằng chơi không đẹp.

thủ tục 3 lần tương tác thực tế hơn, mặc dù tốt hơn hết là không có trường nào được dành chỗ cho tem thời gian). Một số vấn đề về thủ tục này sẽ được tóm tắt bây giờ. Thông điệp cuối cùng của thủ tục này là chữ ký của Alice lên cả thách thức của Bob và nhận dạng của Bob : $s_A\{R_B, Bob\}$[3]. Điều này cho phép Bob nhận được chữ ký của Alice lên đại lượng mà Bob nắm quyền kiểm soát. Điều này là không nên, mặc dù không rõ ràng rằng sử dụng nó như thế nào cho những tấn công trực tiếp. Vấn đề thứ hai là việc đề xuất sử dụng trường dữ liệu mã không bắt buộc trong thủ tục để để hoàn thành việc trao đổi khóa; việc này không đảm bảo tính an toàn chuyển tiếp hoàn thiện[4]. Tiếp theo nữa là với việc sử dụng trường này, chưa có đảm bảo gì để người gửi dữ liệu đã mã thực sự biết dữ liệu mã, và thực tế là kẻ thù địch có thể dán dữ liệu được mã của người khác vào đó [7], [13]. Vấn đề thứ 3 [17] là giới hạn đối với hệ chữ ký số được sử dụng cần có khả năng vừa mã dữ liệu, vừa ký dữ liệu, điều đó loại trừ nhiều lược đồ chữ ký dự tuyển, trong đó có cả NIST DSA [10].

Thủ tục 3 lần tương tác ISO. Như đã nói tới ở Mục 5.3, phiên bản chỉ xác thực của thủ tục STS chính là thủ tục 3 lần tương tác được ISO đề nghị gần đây [1]. Những điểm khác biệt là thủ tục ISO cho phép những bản sao dư thừa của các số ngẫu nhiên, những trường có thể lựa chọn cho nhận dạng của người nhận thông điệp có định trước, các trường lựa chọn cho đoạn văn bản bất kỳ. Do hạn chế của các thủ tục chỉ xác thực như đã bàn ở trên, trong phần lớn các ứng dụng người ta mong chờ rằng chức năng thiết lập khóa của thủ tục ISO (được cung cấp bởi các trường lựa chọn cho đoạn văn bản bất kỳ cả ở trong và ở ngoài phần đã ký của mỗi thông điệp) sẽ được khai thác. Nhắc lại những điều chú ý đã được nhắc tới trong X.509, việc sử dụng những trường này nên cẩn thận; hơn nữa, chú ý rằng việc sử dụng chúng để vận chuyển các khóa phiên đã được mã hóa không đảm bảo tính an toàn chuyển tiếp hoàn thiện.

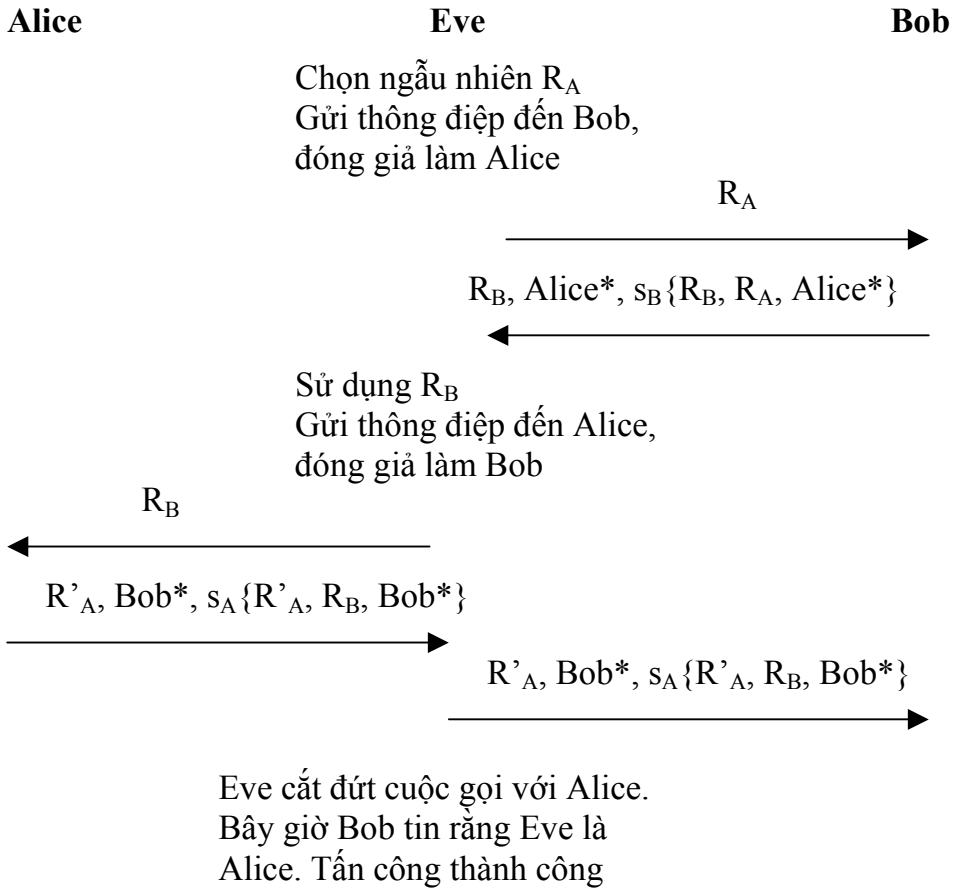
Tấn công một thủ tục xác thực nào đó. Để tăng thêm tài liệu nói về các tấn công đối với các thủ tục đặc biệt, để nhấn mạnh việc các lỗ hổng có thể dễ được đưa vào và dễ bị bỏ qua khi xem xét như thế nào, bây giờ chúng ta hãy xem xét một biến dạng sau (có sai sót) của thủ tục xác thực ISO. Thực ra, biến dạng này chính là phiên bản ban đầu của thủ tục. Tức là, Alice được cho phép sử dụng số ngẫu nhiên mới R'_A thay cho R_A trong thông điệp thứ ba; R'_A được gửi cùng với trường văn bản rõ thêm trong thông điệp thứ ba. Trong thủ tục có biến đổi này, kẻ thù địch Eve có thể xác thực cô ta như Alice đối với đối tác không nghi ngờ Bob như sau (xem sơ đồ dưới đây). Eve gọi Bob, đóng giả làm Alice, gửi thách thức tới

[3] Trong phiên bản trước đây của X.509, thông tin cuối cùng chỉ là $s_A\{R_B\}$; từ đó đến nay khuyến nghị đã thay đổi về hình thức.

[4] Chú ý rằng việc dùng RSA [26] như trước đây để nhằm trao đổi khóa một cách tương tự không đảm bảo bí mật chuyển tiếp hoàn hảo.

Bob; Eve phúc đáp lại phản thách thức (counter-challenge) của Bob bằng cách gọi Alice và làm cho Alice phúc đáp đúng thách thức; sau đó Eve ngắt cuộc gọi với Alice và chuyển phúc đáp đúng qua cho Bob, như vậy là hoàn thành việc xác thực từ quan điểm của Bob. Chú ý rằng tấn công này thành công ngay cả nhận dạng của người nhận được định trước của mỗi thông điệp được gắn kèm ngay bên trong phần ký của mỗi thẻ xác thực, đây là lựa chọn dùng được trong định nghĩa hình thức của thủ tục ISO có liên quan. Để nhấn mạnh điều này, những nhận dạng chủ yếu đã được đưa vào và được ký hiệu cùng với dấu * trong tấn công được chi tiết hóa dưới đây. Để cho đơn giản, các chứng thực không được chỉ ra.

Liên quan đến các tấn công khác đã được viết trong các tài liệu, chúng ta chú ý rằng Bird và các tác giả khác ([5], mục 4) đã chi tiết hóa tấn công đối với một thủ tục cụ thể. Đây là trường hợp đặc biệt của lớp tổng quát các tấn công phản chiếu trong đó người thách thức bị mắc mẹo trong việc cung cấp câu trả lời cho chính câu hỏi của anh ta [19].



7. Các kết luận

Sau đây là một số nguyên tắc chung, cần cẩn thận tuân thủ khi thiết kế các thủ tục xác thực. Trong khi phần lớn các nguyên tắc này đã được chú ý đến ở trên, để cho tiện chúng tôi tập hợp chúng lại ở đây.

1. ***Việc xác thực và trao đổi khóa phải gắn kết với nhau.*** Nếu việc xác thực và trao đổi khóa là độc lập, kẻ tấn công có thể cho phép 2 bên thực hiện việc xác thực một cách tự do, sau đó đóng vai một bên nào đó trong việc trao đổi khóa. Điều này cho phép kẻ tấn công có thể đóng giả một bên hợp pháp sau khi xác thực và việc trao đổi khóa được thực hiện.

2. ***Tính phi đối xứng trong thủ tục là cần thiết.*** Tính đối xứng trong thủ tục cần được dùng một cách thận trọng, có 2 lý do, thứ nhất là khả năng có thể của các tấn công phản chiếu, thứ hai là các tấn công trong đó các phúc đáp từ một bên tham gia có thể được sử dụng lại trong thủ tục. Như một ví dụ minh họa rõ nét, phúc đáp xác thực của 2 bên cần phải không giống nhau.

3. ***Các thông điệp trong một lần vận hành thủ tục cụ thể cần có liên kết logic hay kết nối theo một cách nào đó để ngăn cản việc sử dụng lại các thông diệp có trước hoặc việc đưa vào các thông điệp từ một vận hành song song.*** Mục đích ở đây là tránh tấn công lặp lại và tấn công chèn ngang. Các thông điệp nên được liên kết đến khung thời gian hiện hành (có nghĩa là thông qua việc hợp nhất của các số ngẫu nhiên vừa được sinh ra). Tấn công đặc biệt được chi tiết hóa trong mục 6 là có thể do sự thiếu vắng của móc xích như vậy của các thông điệp; tương tự, tấn công của người đứng giữa được bàn đến bởi Gengio [4] là có thể trong các thủ tục nhưng sẽ thất bại khi đối chọi với nguyên tắc này.

4. ***Các bên tham gia thực hiện thao tác mật mã (như chữ ký số) nên hợp nhất vào dữ liệu được thao tác một lượng hợp lý dữ liệu mà anh ta chọn ngẫu nhiên.*** Nói một cách khác, thủ tục không yêu cầu các bên thực hiện các thao tác mật mã với đầu vào có thể bị hoàn toàn kiểm soát bởi kẻ thù địch. Nguyên tắc "thêm muối của riêng mình" này nhằm ngăn chặn kẻ thù địch muốn nhận được câu trả lời đối với những câu hỏi đặc biệt mà anh ta không thể tự trả lời được. Điều này cũng chống lại tấn công bản mã lựa chọn ([6, trang 27]). Liên quan đến nguyên tắc này, chúng tôi nêu ra nguyên tắc sau được trích đoạn từ Moore [20, mục III]:

5. ***Chữ ký đích thực nên nhận được từ biến đổi của thông điệp trong không gian thông điệp là tập thưa trong miền giá trị của hàm chữ ký.*** Ví dụ, yêu cầu độ dư, hay một sự tính trước nào khác, trong dữ liệu được đem ký, có thể làm trở ngại các tấn công mà kẻ thù địch cố gắng giả chữ ký mới bằng cách kết hợp các chữ ký đích thực nhận được từ trước. Với thủ tục STS, hàm băm được chọn khi băm các lũy thừa cần sinh ra kết quả nhỏ hơn kích thước tối đa của dữ liệu vào cho quá trình ký, như vậy sẽ cho phép thêm độ dư được thêm vào kết quả băm trước khi ký.

Thủ tục trạm-tới -trạm được đề xuất thỏa mãn các nguyên tắc trên, đồng thời những tính chất mong đợi được nêu ở mục 4 (độ an toàn chuyển tiếp hoàn thiện, xác thực trực tiếp, không đòi hỏi tem thời gian). Tính tương thích của nó với các thủ tục xác thực ISO đang nổi lên, và khả năng của nó đảm bảo việc thiết lập khóa trong khuôn khổ, thêm vào tính hấp dẫn của nó. Hơn nữa, thủ tục trạm-tới trạm sử dụng số ít nhất các thông điệp cần thiết cho việc xác thực hỏi-đáp hai phía dựa vào số ngẫu nhiên (bằng 3), và chỉ yêu cầu một lần sinh chữ ký, một lần kiểm tra chữ ký, hai phép mã hóa đối với từng bên (cùng với việc kiểm tra chữ ký thêm nếu chứng thực được sử dụng trên cơ sở dựa vào vận hành để kết nối nhận diện của người sử dụng và khóa công khai).

Lược đồ chữ ký thích hợp có thể được sử dụng trong thủ tục STS, bao gồm Thuật toán chữ ký số (Digital Signature Algorithm) được đề xuất gần đây bởi NIST [10]. Với lý do tính hiệu quả thực tế, lược đồ chữ ký thay thế hiển nhiên là RSA [26]. Tương tự, thuật toán mã dữ liệu đối xứng thích hợp bất kỳ có thể được sử dụng. Trong một số ứng dụng có thể cần phải tránh việc sử dụng thuật toán mã. Một phương pháp được xem xét để tránh việc sử dụng thuật toán mã hóa E_K như sau: thay chữ ký được mã hóa bởi chữ ký cộng với mã xác thực thông điệp (message authentication code-MAC) trên chữ ký; có nghĩa là thay $E_K(s)$, với $s=s_B\{\alpha^y, \alpha^x\}$ (như ở trong mục 5.1), bởi $(s, M_K(s))$, với M_K là MAC có khóa K. Phía bên nhận cần phải kiểm tra cả chữ ký và MAC trên chữ ký. Trong khi cho phép tránh được yêu cầu cần có khả năng mã/dịch (cả hai thủ tục Kerberos và X.509 đều đòi hỏi), nhược điểm của phương án này là cần truyền thêm dữ liệu.

Lời cảm ơn

Các tác giả xin cảm ơn tới tập thể đồng nghiệp vì sự hỗ trợ và các ý tưởng liên quan đến các thủ tục được xem xét, và trong các thảo luận đặc biệt với Carlise Adams và Warwick Ford.

Tài liệu tham khảo

1. Information Technology- Security Techniques. *Entity Authentication Mechanisms- Part 3: Entity Authentication Using a Public- Key Algorithm* (CD 9798-3), Nov. 1991 (ISO/IEC JTCI/SC27 Commitee Draft #4).
2. Bauspiess, F. and Knobloch, H.-J. 1990. How to keep authenticity alive in a computer network. Advance in Cryptology-Eurocrypt 89, (J.J. Quisquater and J. Vandewalle, eds.) *Lecture Notes in Computer Science* 434:38-46, Berlin/New York: Springer Verlag.
3. Bellovin, S.M. and Merritt, M.1990. Limitations of the Kerberos authentication system. *ACM Computer Communication Review* 20 (5): 175-183.
4. Bengio, S., Brassard, G., Desmedt, Y.G., Coutier, C., Quisquater, J.-J. 1991, *Secure implementation of identification system*. J. Cryptology 4 (3): 175-183.

5. Bird, R., Gopal, I., Herzberg, A., Janson, P., Kutten, S., Molva, R., and Yung, M. Forthcoming. Systematic design of two-party authentication protocols. *Advances in Cryptology-Crypto'91*, Berlin/New York: Springer-Verlag.
6. Brassard, G. 1988. Modern Cryptology, Lecture Notes in Computer Science 325. Berlin/New York: Springer- Verlag.
7. Burrows, M., Abadi, M., and Needham, R. 1990. A logic of authentication. *ACM Transactions on Computer Systems* 8 (1):18-36.
8. Denning, D.E. and Sacco, G.M. 1981. Timestamps in key distribution protocols. *Comm. ACM* 24 (8): 533-536.
9. Diffie, W. and Hellman, M.E. 1976. New directions in cryptography. *IEEE Trans. Info. Theory* IT-22 (6):644-654.
10. (proposed U.S. FIPS) Digital Signature Standard (DSS), announced in *Federal Register*, vol 56, no. 169 (Aug. 30, 1991), 42980-42982.
11. ElGamal, T. 1988. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Info. Theory* IT-31 (4): 469-472.
12. Fiat, A. and Shamir, A. 1987. How to prove yourself: pravtice solutions to identification and signature problems. *Advances in Cryptology-Crypto 86* (A. Odlyzko, ec.), *Lecture Notes in Computer Science* 263: 194-196, Berlin/New York: Springer-Verlag.
13. Gaarder, K. and Snekkenes, E. 1991. Applying a formal analysis technique to CCITT X.509 strong two-way authentication protocol. *J. Cryptology* 3 (2):81-98.
14. Guillou, L.C.and Quisquater, J.-J. 1988. A practical zero-knowledge protocol fitted to security microprocessing minimizing both transmission and memory. *Advances in Cryptology- Eurocrypt'88,* (C.G. Günther, (ed.), Lecture Notes in Computer Science 330: 123-128, Berlin/New York: Springer-Verlag.
15. Günther, C.G. 1990. An identity-based key-exchange protocol. *Advances in Cryptology-Eurocrypt 89*, (J.-J. Quisquater and J. Vandewalle, eds.), *Lecture Notes in Computer Science* 434:29-37, Berlin/New York: Springer-Verlag.
16. Haber, S. and Stornetta, W.S. 1991. How to time-stamp a digital document. *J. Cryptology* 3 (2):99-111.
17. I'Anson, C. and Michell, C. 1990. Security defects in CCITT Recommendation X.509- The Directory Authentication Framework. *Computer Communication Review* 20 (2):30-34.
18. Kohl, J. and Neuman, B.C. 1991. The Kerberos network authentication service. MIT Priject Athena Version 5.
19. Mitchell, C. 1989. Limitations of challenge-response entity authentication. *Electronic Letters* 25 (17): 195-196.
20. Moore, J.H. 1988. Protocol failures in cryptosystems. *Proc. of the IEEE* 76 (5):594-602.
21. O'Higgins, B., Diffie, W., Strawczynski, L. and de Hoog, R. 1987. Encryption and ISDN- A Natural fit. In *Proc. 1987 International Switching Symposium*, Pheonix Arizona, pp. A1141-7.

22. Okamoto, E. and Tanaka, K. 1989. Key distribution system based on identification information. IEEE J. Selected Areas in Comm. 7 (4): 481-485.
23. Odlyzko, A.M. 1985. Discrete logarithm in finite fields and their cryptographic significance. *Advances in Cryptology-Eurocrypt 84*, (T. Beth, N. Cot and I. Ingemarsson, eds.), *Lecture Notes in Computer Science* 209:224-314, Berlin/New York: Springer-Verlag.
24. LaMacchia, B.A. and Odlyzko, A.M. 1991. Computation of discrete logarithms in prime fields. *Designs, Codes and Cryptography* I (1): 47-62.
25. Pohlig, S.C. and Hellman, M. 1978.An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory* IT-24: 106-110.
26. Rivest, R.L. Shamir, A. and Adlemman, L. 1978. A method for obtaining digital signatures and public-key cryptosystems. Comm. ACM 21: 120-126.
27. Rivest, R.L. and Shamir, A. 1984. How to expose an eavesdropper. *Comm. ACM 27* (4): 393-395.
28. Schnorr, C.P. 1990, 1991. Efficient signature generation by smart cards, J. Cryptology 4 (3): 161-174; see also: Efficient identification and signature for smart cards. *Advances in Cryptology-Crypto 89*, (G. Brassard, ed.), *Lecture Notes in Computer Science* 435:239-251, Berlin/New York: Springer-Verlag.
29. Shamir, A. 1985. Identity-based cryptosystems and signature schemes, *Advances in Cryptology-Crypto 84*, (G.R. Blakley and D. Chaum, eds.), *Lecture Notes in Computer Science* 196: 47-53, Berlin/New York: Springer-Verlag.
30. CCITT Blue Book Recommendation X.509. The Directory-Authentication Framework. 1988. Geneva. March 1988; amended by resolution of Defect 9594/016 (1Q 1991). Also ISO 9594-8.

Cập nhật thông tin về hàm băm SHA-1

Trong những ngày cuối tháng 2 năm 2005, cộng đồng mật mã đã xôn xao vì thông tin phá được SHS-1 bởi 3 người Trung Quốc. Chúng tôi xin giới thiệu về thông báo này và ý kiến bình luận của Bruce Schneier, cũng như giới thiệu 2 bài báo về thành tựu mới đối với việc tấn công hàm băm trong thời gian gần đây. Hiện chúng tôi đang theo dõi bài báo đầy đủ về vấn đề nay.

Mã thám SHA-1

Bruce Schneier

Ngày thứ 3, tôi đã đưa lên trang web (www.schneier.com/blog/archives/2005/02/sha1_broken.html) thông tin về một kết quả mã thám mới (theory.csail.mit.edu/~yiqun/shanote.pdf), đây là tấn công đầu tiên nhanh hơn tấn công vét cạn chống lại SHA-1. Tôi đã viết về SHA, và sự cận thiết phải thay nó từ tháng 9 năm 2004 (www.schneier.com/essay/074-html). Trừ những chi tiết của tấn công mới, mọi cái tôi đã nói vẫn đúng cả. Tôi sẽ trích dẫn từ bài báo này, có thêm tư liệu ở những nơi thích hợp:

> Các hàm băm một chiều là các kiến thiết mật mã được sử dụng trong nhiều ứng dụng. Chúng được sử dụng cùng với các thuật toán khoá công khai cho cả mã mật và chữ ký số. Chúng được sử dụng trong kiểm tra tính toàn vẹn. Chúng được sử dụng trong xác thực. Chúng có nhiều kiểu loại ứng dụng trong rất nhiều giao thức khác nhau. Nhiều hơn nhiều so với các thuật toán mật mã, các hàm băm một chiều là con ngựa thồ của mật mã hiện đại.
>
> Vào năm 1990, Ron Rivest đã sáng tạo ra hàm băm MD4. Vào năm 1992, ông đã cải tiến MD4 và phát triển một hàm băm khác: MD5. Vào năm 1993, NSA đã công bố một hàm băm rất giống với MD5, nó được gọi là SHA. Sau đó, vào năm 1995, khi trích dẫn điểm yếu mới được phát hiện mà nó chối từ chau chuốt lại, NSA đã làm những thay đổi đối với SHA. Thuật toán mới được gọi là SHA-1. Ngày nay, hàm băm thông dụng nhất là SHA-1, cùng với MD5 hãy còn được sử dụng trong các ứng dụng cũ.
>
> Các hàm băm một chiều được giả thiết là có 2 tính chất. Thứ nhất, chúng là một chiều. Điều này có nghĩa là dễ lấy một văn bản và tính giá trị băm, nhưng không thể lấy giá trị băm rồi tạo lại văn bản xuất phát (tôi hiểu "không thể" có nghĩa là "không thể làm trong một lượng thời gian hợp lý"). Thứ hai, chúng là không va chạm. Điều đó có nghĩa là không tìm được 2 văn bản mà khi băm cho cùng một giá trị. Ý nghĩa mật mã đằng sau 2 tính chất này là tinh tế, tôi xin mời các bạn đọc tò mò hãy tìm hiểu thêm trong cuốn sách của tôi "Applied Cryptography" (www.schneier.com/book-applied.html).

> Phá một hàm băm có nghĩa là chỉ ra hoặc một trong 2 tính chất này, hoặc cả 2 tính chất là không đúng.

Đầu tuần này, 3 nhà mật mã học người Trung Quốc đã chỉ ra rằng SHA-1 không phải là không va chạm. Có nghĩa là, họ đã phát triển được một thuật toán để tìm những va chạm nhanh hơn so với vét cạn.

SHA-1 sinh ra giá trị băm 160-bit. Có nghĩa là, mỗi văn bản được băm thành một số 160-bit. Giả thiết rằng có vô số các văn bản mà được băm thành mỗi giá trị có thể, cho nên có vô số các va chạm có thể. Nhưng bởi vì số các giá trị băm có thể là rất lớn, nên khả năng tìm thấy một va chạm một cách tình cờ là nhỏ không đáng kể (một trong 2^{80}, nếu muốn chính xác). Nếu bạn băm 2^{80} văn bản ngẫu nhiên, bạn sẽ tìm được một cặp mà được băm tàhnh cùng một giá trị. Đó là cách "vét cạn" để tìm các va chạm, và nó chỉ phụ thuộc vào độ dài của giá trị băm. "Phá" hàm băm có nghĩa là có thể tìm các va chạm nhanh hơn như thế. Đó chính là cái mà những người Trung Quốc đã làm.

Họ có thể tìm các va chạm của SHA-1 trong 2^{69} lần tính toán, khoảng 2000 lần nhanh hơn tấn công vét cạn. Thực ra, cái đó còn xa so với tính khả thi của công nghệ hiện tại. Hai tính toán lớn so sánh được minh hoạ điểm này.

Vào năm 1999, một nhóm các nhà mật mã học xây dựng một máy phá DES (www.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/). Nó có thể thực hiện 2^{56} phép mã DES trong vào 56 giờ. Để thiết kế máy nài tốn 250,000 USD, mặc dù để nhân bản nó chỉ tốn khoảng 50-70 nghìn USD. Ngoại suy rằng máy này tuân theo luật Moore, thì ngày nay một máy tương tự có thể xây dựng để thực hiện 2^{60} phép mã trong 56 giờ, và 2^{69} phép mã trong 3 năm 3 tháng. Hoặc là, một máy giá 25 –> 38 triệu USD có thể tính 2^{69} phép mã trong 56 giờ.

Về phía phần mềm, so sánh chính là 2^{64} phép tìm khoá đã được làm bởi distributed.net và đã kết thúc vào năm 2002 (stats.distributed.net). Một bài báo (www.windowsitpro.com/Article/ArticleID/26857/26857.html) đã đưa thông tin sau: "Trong một cuộc thi, có 331,252 người tham gia bằng cách cho phép sử dụng các nhịp vi xử lý không dùng đến ở máy tính của họ vào việc khôi phục khoá. Sau 1,757 ngày (4.81 năm), những người tham gia ở Nhật đã tìm được khoá thắng lợi." Luật Moore có nghĩa là ngày nay, tính toán này chỉ chiếm ¼ thời gian đó, hoặc là chỉ yêu cầu ¼ số máy tính, cho nên ngày nay, 2^{69} lần tính sẽ chiếm 8 lần thời gian lâu hơn, hoặc đòi hỏi 8 lần nhiều máy tính hơn.

> Tầm quan trọng của các kết quả này phụ thuộc vào bạn là ai. Nếu bạn là nhà mật mã học, đó là một công việc lớn. Trong khi chưa có một cuộc cách mạng, các kết quả này là các tiến bộ nối tiếp trong lĩnh vực. Các kỹ thuật được mô tả bởi các nhà nghiên cứu là dường như có các tác dụng khác, như là một kết quả, chúng ta sẽ có thể thiết kế các hệ thống mật mã tốt hơn. Đó là cách mà khoa học mật mã phát triển: chúng ta học cách thiết kế các thuật toán mới bằng cách phá các thuật toán khác. Thêm vào đó, các thuật toán từ NSA được xem như một dạng công nghệ xa lạ: chúng đến từ một bộ tộc siêu đẳng mà không có giải thích. Mỗi thành công mã thám chống lại thuật toán của NSA là một điểm dữ kiện thú vị trong câu hỏi ngàn đời xem chúng thực tốt như thế nào.

Đối với một người sử dụng Internet trung bình, các thông tin này không gây ra một nỗi hoảng sợ. Không ai có thể phá các chữ ký số hoặc đọc các văn bản đã được mã trong một thời gian gần. Thế giới điện tử không ít an toàn hơn sau những công bố này so với nó trước đây.

Nhưng có một thành ngữ cũ trong NSA: "Các tấn công luôn trở nên tốt hơn; chúng không bao giờ tồi đi." Chính tấn công của tuần này được xây dựng trên những bài báo mô tả các tấn công chống lại các phiên bản được đơn giản của SHA-1, đó là SHA-0, MD4 và MD5, các nhà nghiên cứu khác sẽ xây dựng tiếp dựa trên kết quả này. Tấn công chống lại SHA-1 sẽ được tiếp tục hoàn thiện, do những người khác đọc nó và phát triển các mẹo nhanh hơn, làm các tối ưu hoá, vv... Và luật Moore sẽ tiếp tục đúng về phía trước, làm cho tấn công đang tồn tại nhanh hơn và chấp nhận được.

Jon Callas (là CTO của PGP), đã nói nói như: " Đã đến lúc phải đi, nhưng chưa phải chạy, đến lối thoát hiểm. Bạn chưa nhìn thấy khói, nhưng còi báo cháy đã kêu." Về cơ bản, đó là những gì tôi đã nói vào tháng 8 năm 2004.

> Đã đến lúc chúng ta phải tránh khỏi SHA-1.
>
> Thật may mắn, có những cái thay thế. NIST đã có các chuẩn cho các hàm băm dài hơn và khó phá hơn: SHA-224, SHA-256, SHA-384 và SHA-512 (csrc.nist.gov/CryptoToolkit/tkhash.html). Chúng là các chuẩn của chính phủ, và đã sẵn sàng để sử dụng. Đây là một cái để lấp chỗ trống tốt, nhưng tôi muốn thấy nhiều hơn.
>
> Tôi muốn nhìn thấy NIST đứng ra bắt nhịp một cuộc thi trên toàn thế giới cho một hàm băm mới, giống như họ đã làm cho thuật toán mã mới, AES, thay cho DES. NIST nên đưa ra lời kêu gọi cho các thuật toán, tổ chức các vòng đánh giá, khi đó cộng đồng mật mã phân tích các đề xuất khác nhau với mục đích thiết lập một thuật toán mới.
>
> Phần lớn các hàm băm mà chúng ta có, và tất cả những cái được sử dụng rộng rãi, là dựa vào các nguyên tắc chung của MD4. Rõ ràng là chúng ta đã học được khá nhiều về các hàm băm trong mười năm gần đây, và chúng tôi nghĩ chúng ta có thể bắt đầu áp dụng hiểu biết này để sáng tạo ra một cái gì đó an toàn hơn.

Các hàm băm là thành tố mật mã được hiểu biết ít nhất, các kỹ thuật băm được phát triển ít hơn so với các kỹ thuật mã hoá. Thường xuyên có những kết quả bất ngờ trong băm. Tôi có một bài báo (eprint.iacr.org/2004/304), viết cùng với John Kelsey, nó mô tả thuật toán để tìm tiền ảnh thứ hai cho SSHA-1, kỹ thuật này có thể tổng quát hoá cho hầu hết mọi hàm băm mật mã , trong 2^{106} lần tính: nhỏ hơn nhiều so với 2^{160} lần tính của vét cạn. Tấn công này là hoàn toàn về mặt lý thuyết và còn xa so với thực hành, nhưng nó biểu thị rằng chúng ta phải học nhiều về cách băm.

Cần phải đọc lại những gì tôi đã viết vào tháng 9 năm ngoái, rằng tôi chờ đợi việc này xảy ra, nhưng không xảy ra nhanh như vậy và không ấn tượng đến như vậy. Các nhà

mật mã học người Trung Quốc xứng đáng được ghi nhận vì công trình của họ, và chúng ta cần phải làm việc để thay thế SHA.

(Đưa lên mạng ngày 18 tháng 2 năm 2005, lúc 11: 24 PM)

Tấn công tìm kiếm va chạm đối với SHA-1
(http://theory.csail.mit.edu/~yiqun/shanote.pdf)

Xiaoyun Wang, Shandong University, China, email: xywang@sdu.edu.cn
Yiqun Lisa Yin, Independent security consultant, USA, email: yyin@princeton.edu
Hongbo Yu, Shandong University, Chian, email: yhb@mail.sdu.edu.cn

Ngày 13 tháng 2 năm 2005

1. Mở đầu

Trong thông báo này, chúng tôi tóm tắt các kết quả của các tấn công tìm kiếm va chạm của chúng tôi đối với SHA-1. Các chi tiết kỹ thuật sẽ được công bố trong một bài báo sắp được công bố.

Chúng tôi đã phát triển một tập các kỹ thuật mới mà rất hiệu quả để tìm kiếm các va chạm trong SHA-1. Phân tích của chúng tôi chỉ ra rằng các va chạm của SHA-1 có thể tìm thấy với độ phức tạp nhỏ hơn 2^{69} phép băm. Đây là tấn công đầu tiên đối với SHA-1 có đầy đủ 80 vòng với độ phức tạp nhỏ hơn cận lý thuyết là 2^{80}. Dựa trên đánh giá của chúng tôi, chúng tôi chờ đợi rằng các va chạm thực của SHA-1 được rút gọn xuống 70 vòng có thể được tìm thấy sử dụng các siêu máy tính hiện nay.

Trong một số năm gần đây, nhiều tiến bộ nghiên cứu đáng kể đã được làm trong việc phân tích các hàm băm. Các kỹ thuật đã được phát triển trong các công trình trước đây cung cấp nền móng quan trọng cho các tấn công mới của chúng tôi đối với SHA-1. Đặc biệt, phân tích của chúng tôi dựa vào tấn công lượng sai nguyên thuỷ đối với SHA-0, tán công va chạm gần đối với SHA-0, các kỹ thuật va chạm nhiều khối cũng như các kỹ thuật căn chỉnh văn bản đã được sử dụng trong tấn công tìm va chạm đối với MD5. Việc phá SHA-1 không thể có nếu thiếu các kỹ thuật phân tích mạnh này.

Các tấn công của chúng tôi áp dụng một cách tự nhiên cho SHA-0 và tất cả các phiên bản rút gọn của SHA-1. Với SHA-0, tấn công là rất hiệu quả đến mức chúng tôi có thể tìm được các va chạm thực sự của SHA-0 đầy đủ với ít hơn 2^{39} phép băm. Chúng tôi cũng áp dụng tấn công đối với SHA-1 được thu gọn còn 58 vòng và tìm được va chạm thực sự với ít hơn 2^{33} phép băm. Hai ví dụ va chạm được đưa ra trong thông báo này.

2. Ví dụ va chạm cho SHA-0

$h_1 = \text{compress}(h_0, M_0)$
$h_2 = \text{compress}(h_1, M_1) = \text{compress}(h_1, M_1')$

h_0: `67452301 efcdab89 98badcfe 10325476 c3d2e1f0`

```
M0:   65c24f5c 0c0f89f6 d478de77 ef255245
      83ae3a1f 2a96e508 2c52666a 0d6fad5a
      9d9f90d9 eb82281e 218239eb 34e1fbc7
      5c84d024 f7ad1c2f d41d1a14 3b75dc18

h1:   39f3bd80 c38bf492 fed57468 ed70c750 c521033b

M1:   474204bb 3b30a3ff f17e9b08 3ffa0874
      6b26377a 18abdc01 d320eb93 b341ebe9
      13480f5c ca5d3aa6 b9f3bd88 21921a2d
      4085fca1 eb65e659 51ac570c 54e8aae5

M1': c74204f9 3b30a3ff 717e9b4a 3ffa0834
      6b26373a 18abdc43 5320eb91 3341ebeb
      13480f1c 4a5d3aa6 39f3bdc8 a1921a2f
      4085fca3 6b65e619 d1ac570c d4e8aaa5

h2:   2af8aee6 ed1e8411 62c2f3f7 3761d197 0437669d
```

Bảng 1: Một va chạm của SHA-0 đầy đủ 80 vòng. Hai văn bản va chạm là (M_0, M_1) và (M_0, M_1'). Lưu ý rằng các qui tắc đệm dữ liệu không được áp dụng đối với văn bản.

3. Ví dụ va chạm cho SHA-1

$$h_1 = compress(h_0, M_0) = compress(h_0, M_0')$$

```
h0:   67452301 efcdab89 98badcfe 10325476 c3d2e1f0

M0:   132b5ab6 a115775f 5bfddd6b 4dc470eb
      0637938a 6cceb733 0c86a386 68080139
      534047a4 a42fc29a 06085121 a3131f73
      ad5da5cf 13375402 40bdc7c2 d5a839e2

M0': 332b5ab6 c115776d 3bfddd28 6dc470ab
      e63793c8 0cceb731 8c86a387 68080119
      534047a7 e42fc2c8 46085161 43131f21
      0d5da5cf 93375442 60bdc7c3 f5a83982

h1:   9768e739 b662af82 a0137d3e 918747cf c8ceb7d4
```

Bảng 2: Một va chạm của SHA-1 được rút gọn còn 53 vòng. Hai văn bản va chạm là M_0 M_0'. Lưu ý rằng các qui tắc đệm dữ liệu không được áp dụng đối với văn bản.

Tìm tiền ảnh thứ hai cho các hàm băm n-bit
với ít hơn nhiều so với 2^n lần băm
(eprint.iacr.org/2004/304)

John Kelsey và Bruce Schneier

Tóm tắt: Chúng tôi cung cấp một tấn công tìm tiền ảnh thứ hai đối với tất cả các hàm băm lặp n-bit với làm mạnh của Damgard-Merkle và các trạng thái trung gian n-bit, cho

phép tiền ảnh thứ hai có thể tìm thấy cho một văn bản có 2^k khối văn bản với khoảng $k \times 2^{n/2+1} + 2^{n-k+1}$ lần tính. Sử dụng SHA-1 như một ví dụ, tấn công của chúng tôi tìm thấy tiền ảnh thứ hai cho một văn bản dài 2^{60} byte trong 2^{106} phép băm, thay cho khối lượng 2^{160} chờ đợi trước đây. Chúng tôi cũng cung cấp các cách rẻ hơn một chút để tìm nhiều va chạm so với phương pháp của Joux [Joux, Multicollisions in Iterated Hash Functions. Applications to Cascaded Constructions, Advances in Cryptology- Crypto'2004 Proceeding, Springer-Verlag, 2004]. Cả hai kết quả này là dựa vào các văn bản mở rộng được (expandable message) (các mẫu để sinh ra các văn bản có độ dài thay đổi, tất cả chúng va chạm trên kết quả băm trung gian ngay lập tức sau khi xử lý văn bản. Chúng tôi cũng cung cấp các thuật toán để tìm các văn bản mở rộng được cho một hàm băm, bằng cách chỉ sử dụng một bội số nhỏ của khối lượng công việc được làm nhằm tìm một va chạm duy nhất trong hàm băm.

Update on SHA-1

Vincent Rijmen và Elisabeth Oswald

Việc nghiên cứu các tấn công đối với SHA-1 trong thời gian từ cuối năm 2004 được đẩy mạnh. Tại Rump session của Crypto'2004 người ta đã dự báo về khả năng sắp bị phá của SHA-1. Bài báo eprint.iacr.org/2005/010 của Vincent Rijmen và Elisabeth Oswald "Update on SHA-1" là một ví dụ. Phiên bản trước của kết quả này đã xuất hiện trong tuyển tập hội nghị CT-RSA 2005, LNCS 3376, Springer-Verlag do A. J. Menezes biên tập (các trang 58-71). Phiên bản mới này có sửa một số lỗi. Sau đây là tóm tắt của bài báo "Update on SHA-1":

> Chúng tôi báo cáo các thí nghiệm mà chúng tôi đã thực hiện để đánh giá độ an toàn của SHA-1 đối với tấn công của Chabaud và Joux [Florent Chabaud, Antoine Joux, Differential Collisions in SHA-0, Advances in Cryptology-Crypto'98, LNCS 1462, H. Krawczyk, Ed., Springer-Verlag, 1998, pp. 56-71]. Chúng tôi trình bày một số ý tưởng để tối ưu hoá tấn công và một số tính chất của thủ tục mở rộng văn bản. Cuối cùng, chúng tôi chỉ ra rằng phiên bản được rút gọn của SHA-1 với 53 vòng thay cho 80 vòng có thể tìm được các va chạm với ít hơn 2^{80} phép băm.
