情報保護學會論文誌
第11券 第6號, 2001. 12

PKC '98에 제안된 해쉬함수의 공격

한 대 완*, 박 상 우*, 지 성 택*

Cryptanalysis of a Hash Function Proposed at PKC'98

Daewan Han*, Sangwoo Park*, Seongtaek Chee*

요 약

PKC'98에서 신 상욱 등은 SHA-1, RIPEMD-160, HAVAL의 장점을 이용한 새로운 해쉬함수를 제안하였다. 제안자들은 해쉬함수에 사용된 부울함수가 SAC(Strict Avalanche Criterion) 특성을 만족한다고 주장하였으나, 실제로는 SAC을 만족하지 않음을 알 수 있다. 본 논문에서는 PKC'98에 제안된 해쉬함수에 사용된 부울함수가 SAC을 만족하는 경우의 충돌쌍을 찾음으로써, 일반적으로 암호학적으로 안전한 것으로 인식되는 논리를 사용하는 것이 오히려 안전성 저하 요인이 될 수도 있음을 지적한다.

ABSTRACT

At PKC'98, SangUk Shin et al. proposed a new hash function based on advantages of SHA-1, RIPEMD-160, and HAVAL. They claimed that the Boolean functions of the hash function have good properties including the SAC(Strict Avalanche Criterion). In this paper, we first show that some of Boolean functions which are used in Shin's hash function does not satisfy the SAC, and then argue that satisfying the SAC may not be a good property of Boolean functions, when it is used for constructing compress functions of a hash function.

Key word : *Hash function, Boolean function, SAC, Collision*

Ⅰ. 서 론

해쉬함수란 임의의 유한 길이의 비트 스트링을 고정된 길이의 비트 스트링으로 변환하는 함수를 말하며, 디지털 서명과 결합되어 데이터 무결성(data integrity)을 제공하고 컴퓨터 시스템에서 중요 정보의 인증(authentication)과 무결성(integrity)를 제공하는 등, 현대 암호학에서 매우 중요한 역할을 수행한다.[1] 해쉬함수 h가 암호학적으로 안전하기 위해서는 다음 두 가지 성질을 만족해야 한다.

- $y=h(x)$가 주어졌을 때, x를 역계산하는 것은 계산상 불가능하다.(일방향성)
- $h(x)=h(x')$이며, $x\neq x'$인 충돌쌍 (x,x')을 찾는 것은 계산상 불가능하다.(충돌회피성)

해쉬함수의 충돌쌍을 쉽게 찾을 수 있으면 디지털 서명에서 내부부정이 발생할 수 있고 데이터 무결성을 확인할 수가 없게 되는 등 해쉬함수를 응용하는데 있어서 큰 문제가 발생한다. 이러한 이유로 해쉬함수의 안전성은 주로 충돌회피성의 만족 여부를 가지고 논하고, 따라서 공격도 동일한 해쉬값을 가지는 충돌쌍을 찾는 부분에 집중되어 있다.

1990년대로 접어들면서 해쉬 기능만을 목적으로

* 한국전자통신연구원 부설 국가보안기술연구소({dwh,psw,chee}@etri.re.kr)

하는 전용 해쉬함수들이 제안되었는데, RSA Data Seurity Inc.의 R.L.Rivest가 제안한 MD(Message Digest) 계열[2,3] 해쉬함수들과 NIST가 미국 표준으로 제정한 SAH-1(Secure Hash Algorithm-1)[4], HAVAL[5] 등이 현재 가장 주목받는 해쉬함수들이다. 1998년 PKC'98에서 신상욱 등은 SHA-1과 HAVAL의 장점을 이용해 새로운 해쉬함수를 제안하였다.[6] [6]에서 해쉬함수의 제안자들은 해쉬함수에 사용된 부울함수가 SAC(Strict Avalache Criterion)[7]을 만족한다고 주장하였으나, 실제로는 일부 부울함수는 SAC을 만족하지 않음이 확인되었다.

본 논문에서는 PKC'98에 제안된 해쉬함수에 대하여 간단히 살펴보고, 부울함수가 SAC을 만족할 경우를 가정하고 해쉬함수의 충돌쌍을 찾는 공격을 시도한다.

본 논문의 2장에서는 PKC'98에서 제안된 해쉬함수를 구성요소별로 살펴본다. 3장에서는 제안 해쉬함수의 약점을 분석한 후, 4장에서는 충돌쌍을 찾는 공격을 시도하고, 끝으로 5장에서는 결론을 맺는다.

II. PKC'98에 제안된 해쉬함수

지금부터는 PKC'98에서 제안된 해쉬함수를 간단히 P-해쉬함수라고 부르자. 본 장에서는 P-해쉬함수의 알고리즘에 대해서 살펴보자.

먼저 알고리즘에 사용되는 용어와 기호를 정의하면 다음과 같다.

- 워드 : 32 비트 문자열
- 블록 : 512 비트(16 워드) 문자열
- + : 워드 사이의 2^{32}로 법(mod) 덧셈 연산
- $X^{\lll s}$: X를 s비트만큼 왼쪽으로 순환 이동시킨 값
- $X \wedge Y$: X와 Y의 비트별 논리합
- $X \vee Y$: X와 Y의 비트별 논리곱
- $X \oplus Y$: X와 Y의 비트별 배타적 논리합

다음으로 P-해쉬함수의 특징을 부분별로 살펴보자.

2.1 입력 블록의 길이와 패딩과정

512 비트 단위로 입력 메시지를 처리한다. 마지막 메시지 블록은 448 비트가 되도록 '1' 다음에 필요한 수만큼의 '0'을 채운다. 마지막 64비트는 원래 메시지의 길이의 법 2^{64}값으로 채운다.

2.2 초기값(IV)

메시지 처리에 사용되는 5개의 연쇄 변수 (A, B, C, D, E)의 초기값은 다음과 같다.(16진수 표기)

A	B	C	D	E
67452301	efcdab89	98badcfe	10325476	c3d2e1f0

2.3 상수

각 라운드에서 사용되는 상수 K는 다음과 같다.(16진수 표기)

K_1	K_2	K_3	K_4
0	5a827999	6ed9ba1	8f1bbcdc

2.4 메시지 변수 확장

16개 입력 메시지 $X_i(i=0,1,\cdots,15)$에서 8개 메시지 변수를 다음과 같은 방법을 이용하여 추가로 생성한다.

$$X_{16+i} = (X_{0+i} \oplus X_{2+i} \oplus X_{7+i} \oplus X_{12+i})^{\lll 1} \quad (i=0,1,\cdots,7) \qquad (1)$$

2.5 메시지 워드의 적용 순서

메시지 워드 적용 순서를 결정하기 위하여 다음 수열 ρ를 정의한다.

i	0	1	2	3	4	5	6	7
$\rho(i)$	4	21	17	1	23	18	12	10
i	8	9	10	11	12	13	14	15
$\rho(i)$	5	16	8	0	20	3	22	6
i	16	17	18	19	20	21	22	23
$\rho(i)$	11	19	15	2	7	14	9	13

각 라운드에서의 메시지 적용 순서는 다음의 순열에 의하여 결정된다.

라운드	1	2	3	4
순 열	id	ρ	ρ^2	ρ^3

2.6 단계 연산

1 단계에서의 연산은 다음과 같이 정의된다.

$$A=(f(A,B,C,D,E)+X_i+K)^{\lll s},\ B=B^{\lll 10}$$
$$B=A,\ C=B,\ D=C,\ E=D,\ A=E$$

2.7 부울 함수

각 라운드의 단계 연산에서 사용되는 부울 함수는 다음과 같고, f_0는 1라운드에, f_1는 2와 4라운드에, f_2는 3라운드에 각각 사용된다.

$$f_0(x_1,x_2,x_3,x_4,x_5)=(x_1\wedge x_2)\oplus(x_3\wedge x_4)\oplus(x_2\wedge x_3\wedge x_4)\oplus x_5 \quad (2)$$

$$f_1(x_1,x_2,x_3,x_4,x_5)=x_2\oplus((x_4\wedge x_5)\vee(x_1\wedge x_3)) \quad (3)$$

$$f_2(x_1,x_2,x_3,x_4,x_5)=x_1\oplus(x_2\wedge(x_1\oplus x_4))\oplus(((x_1\wedge x_4)\oplus x_3)\wedge x_5) \quad (4)$$

2.8 순환 이동

각 라운드의 단계 연산에서 사용되는 순환 이동 값 s는 입력 메시지에 의존하여 다음처럼 정의된다.

$$s=X_i \bmod 32$$

각 라운드에서 X_i의 순서를 결정하는 순서는 다음과 같다.(ρ는 메시지 적용 순서에서의 치환)

라운드	1	2	3	4
순 열	ρ^3	ρ^2	ρ	id

Ⅲ. P-해쉬함수의 분석

본 장에서는 다음 장에서 수행할 공격에 이론적 근거가 되는 P-해쉬함수의 약점을 분석해 보자.

3.1 메시지 확장

P-해쉬함수는 16개의 입력 메시지 $X_0, \cdots, X_{15}$로부터 식 (1)을 이용하여 8개의 메시지 $X_{16}, \cdots, X_{23}$을 만들어 낸다. 이러한 작업은 하나의 메시지에 준 차분이 다른 메시지에도 영향을 끼침으로써 충돌쌍을 찾는 공격을 시도할 때 어려움을 증가시킨다.

식 (1)을 풀어서 다시 정리하면 아래의 식들과 같다.

$$X_{16}=(X_0\oplus X_2\oplus X_7\oplus X_{12})^{\lll 1} \quad (5)$$

$$X_{17}=(X_1\oplus X_3\oplus X_8\oplus X_{13})^{\lll 1} \quad (6)$$

$$X_{18}=(X_2\oplus X_4\oplus X_9\oplus X_{14})^{\lll 1} \quad (7)$$

$$X_{19}=(X_3\oplus X_5\oplus X_{10}\oplus X_{15})^{\lll 1} \quad (8)$$

$$X_{20}=(X_4\oplus X_6\oplus X_{11}\oplus X_{16})^{\lll 1} \quad (9)$$

$$X_{21}=(X_5\oplus X_7\oplus X_{12}\oplus X_{17})^{\lll 1} \quad (10)$$

$$X_{22}=(X_6\oplus X_8\oplus X_{13}\oplus X_{18})^{\lll 1} \quad (11)$$

$$X_{23}=(X_7\oplus X_9\oplus X_{14}\oplus X_{19})^{\lll 1} \quad (12)$$

식 (5)와 (9)를 보면 X_0가 변하면 X_{16}과 X_{20}이 함께 변함을 알 수 있다. 이런 식으로, 각각의 X_i $(0\le i\le 15)$의 변화가 영향을 주는 $X_j(0\le j\le 15)$들을 표로 작성하면 〔표 1〕과 같다.(표기의 간결함을 위하여 X_i를 i로 표기함)

〔표 1〕 메시지 확장의 영향

0	1	2	3	4	5	6	7
16	17	16	17	18	19	20	16
20	21	18	19	20	21	22	20
		20	21	22	23		21
		22	23				23
8	9	10	11	12	13	14	15
17	18	19	20	16	17	18	19
21	22	23		20	21	22	23
22	23			21	22	23	

〔표 1〕을 살펴보면, X_{10}과 X_{15}가 같은 메시지들에 영향을 미치는 것을 알 수 있다. 따라서, $X_{10}=X_{15}$를 항상 만족시키면서 X_{10}을 변화시키면 그 영향은

X_{10}과 X_{15}에만 미치게 되고, 같은 원리가 X_8, X_{13}의 쌍과 X_9, X_{14}의 쌍에도 적용된다. 이러한 사실은 충돌쌍을 찾는 공격을 시도할 때 매우 유용하게 사용될 수 있다.

3.2 부울 함수 특성 분석

P-해쉬함수의 제안자들은 해쉬함수에 사용된 3개의 부울 함수들이 다음의 3가지 특성을 만족하는 것으로 주장하였다.

- 출력의 0과 1의 개수가 동일하다.
- 비선형성이 높다.
- SAC을 만족한다.

어떤 함수가 SAC을 만족한다는 것은 부울 함수의 입력값의 1비트가 변화했을 때, 출력값이 변화할 확률이 $\frac{1}{2}$이 됨을 말한다.

P-해쉬함수의 제안자들은 f_0, f_1, f_2가 모두 SAC을 만족한다고 주장하였으나, 실제 검증 결과 f_2만이 SAC을 만족하는 것으로 나타났다. f_0가 SAC을 만족하지 않는다는 것은 식 (2)에서 x_5가 독립적으로 XOR됨으로써 x_5가 변하면 출력값도 확률 1로써 변한다는 것을 보면 알 수 있다. 마찬가지로 식 (3)을 보면 x_2의 변화가 확률 1로써 f_1의 출력값에 영향을 미치는 것을 알 수 있다.

제안자들이 의도한 바는 아니었지만, P-해쉬함수에 사용된 부울 함수들이 SAC을 만족하지 않는다는 사실이 오히려 해쉬함수의 안전성을 더 증가시켜주는 요인이 되었다. 다음 절에서 부울 함수가 SAC을 만족하는 P-해쉬함수에 대한 공격을 시도함으로써 그 이유를 밝히고자 한다.

지금부터는 '부울함수가 SAC을 만족하는 P-해쉬함수'를 간단히 'SAC P-해쉬함수'라고 부르기로 하자.

Ⅳ. SAC P-해쉬함수의 공격

이번 장에서는 SAC P-해쉬함수는 2^{-30}의 확률로 충돌쌍이 존재함을 보이고자 한다.

먼저, 이번 절에서 사용하게 될 기호들을 다음과 같이 약속하자.

- $X=(X_i)_{0\le i\le 15}$, $\widetilde{X}=(\widetilde{X}_i)_{0\le i\le 15}$: 충돌쌍이 될 서로 다른 입력 메시지
- A_i, B_i, C_i, D_i, E_i : X를 입력으로 취했을 때 i 단계 후의 버퍼값
- $\widetilde{A}_i, \widetilde{B}_i, \widetilde{C}_i, \widetilde{D}_i, \widetilde{E}_i$: $\widetilde{X}$를 입력으로 취했을 때 i 단계 후의 버퍼값
- s_i : X를 입력으로 취했을 때 i 단계에서 사용하게 될 순환이동 값
- $\widetilde{s}_i$: $\widetilde{X}$를 입력으로 취했을 때 i 단계에서 사용하게 될 순환이동 값

4.1 연속된 6단계 특성 분석

이번 절에서는 SAC P-해쉬함수의 충돌쌍을 찾는데 기초가 되는 논리를 설명하기 위하여 연속된 6단계에서 충돌쌍이 발생할 확률을 계산하기로 한다.

다음 사항을 가정하기로 하자.

- f : SAC을 만족하는 부울 함수
- 6단계의 시작은 버퍼 A부터 갱신
- $A_0=\widetilde{A}_0, B_0=\widetilde{B}_0, C_0=\widetilde{C}_0, D_0=\widetilde{D}_0, E_0=\widetilde{E}_0$
- $X_1\oplus\widetilde{X}_1=1^{\ll 31}$
- $X_2=\widetilde{X}_2,\ X_3=\widetilde{X}_3,\ X_4=\widetilde{X}_4,\ X_5=\widetilde{X}_5$
- $s_i=\widetilde{s}_i\ (1\le i\le 6)$

이상을 가정하여 각 단계에서 갱신되는 버퍼상태를 표로 나타내면 〔표 2〕와 같다. 음영 표시는 새로 갱신되는 변수임을 의미한다.

이제 충돌쌍이 발생할 확률을 계산해 보자.

〔표 2〕 각 단계에서 갱신되는 정보

단계	A	B	C	D	E	입력
1	A_1	B_1	C_1	D_1	E_1	X_1
2	A_2	B_2	C_2	D_2	E_2	X_2
3	A_3	B_3	C_3	D_3	E_3	X_3
4	A_4	B_4	C_4	D_4	E_4	X_4
5	A_5	B_5	C_5	D_5	E_5	X_5
6	A_6	B_6	C_6	D_6	E_6	X_6

4.1.1 단계 1

단계 1에서는 다음과 같은 연산이 이루어진다.

$$A_1=(f(A_0,B_0,C_0,D_0,E_0)+X_1+K)^{\ll s_1}$$
$$\widetilde{A}_1=(f(\widetilde{A}_0,\widetilde{B}_0,\widetilde{C}_0,\widetilde{D}_0,\widetilde{E}_0)+\widetilde{X}_1+K)^{\ll s_1}$$

$$A_0=\widetilde{A}_0,\ B_0=\widetilde{B}_0,\ C_0=\widetilde{C}_0,\ D_0=\widetilde{D}_0,\ E_0=\widetilde{E}_0$$

이므로, 단계 1이 수행되면 X_1과 $\widehat{X_1}$의 차이만큼 A_1과 $\widehat{A_1}$에만 차분이 발생하게 된다. 즉,

$$\Delta X_1 = \alpha \Rightarrow \Delta A_1 = \alpha^{\ll s_1}$$

여기에서 $\alpha = 2^{31}$ 이므로, $A_1 \oplus \widehat{A_1} = 2^{s_1-1}$이 확률 1로 성립하게 된다.

4.1.2 단계 2

단계 2에서의 연산은 다음과 같다.

$$E_2 = (f(E_1, A_1, B_1, C_1, D_1) + X_2 + K)^{\ll s_2}$$
$$\widehat{E_2} = (f(E_1, \widehat{A_1}, B_1, C_1, D_1) + \widehat{X_2} + K)^{\ll s_2}$$

$X_2 = \widehat{X_2}$이므로, $E_2 = \widehat{E_2}$가 되기 위한 조건은

$$f(E_1, A_1, B_1, C_1, D_1) = f(E_1, \widehat{A_1}, B_1, C_1, D_1) \quad (13)$$

가 되는 것이다. 그런데, f의 입력값으로 들어가는 5개의 변수들을 살펴보면, 다른 변수들은 다 같고, A_1과 $\widehat{A_1}$은 단계 1의 결과에 따라 1 비트 위치에서만 다르게 된다. 따라서, f가 SAC을 만족하므로 식 (13)이 만족할 확률은 $\frac{1}{2}$이 되고, 따라서 $E_2 = \widehat{E_2}$도 $\frac{1}{2}$의 확률로 성립하게 된다.

4.1.3 단계 3, 4, 5, 6

단계 2에서 $E_2 = \widehat{E_2}$를 만족했다고 가정하면, 단계 3에서의 연산은 다음과 같게 된다.

$$D_3 = (f(D_2, E_2, A_2, B_2, C_2) + X_3 + K)^{\ll s_3}$$
$$\widehat{D_3} = (f(D_2, E_2, \widehat{A_2}, B_2, C_2) + \widehat{X_3} + K)^{\ll s_3}$$

단계 2에서와 마찬가지로, 단계 3의 부울 함수 f에 입력값으로 들어가는 5개의 변수들도 A_2와 $\widehat{A_2}$의 1비트 위치에서만 다르게 된다. 따라서, 단계 2에서와 같은 논리에 의해 $D_3 = \widehat{D_3}$도 $\frac{1}{2}$의 확률로 성립하게 된다.

단계 4,5,6의 과정에서도 위와 같은 논리가 동일하게 적용되고, 따라서 $C_4 = \widehat{C_4}$, $B_5 = \widehat{B_5}$, $E_6 = \widehat{E_6}$가 성립할 확률도 각각 $\frac{1}{2}$이 되며, 이상의 결과를 종합하면, $A_0 = \widehat{A_0}, \cdots, E_0 = \widehat{E_0}$일 때, $A_6 = \widehat{A_6} \cdots$, $E_6 = \widehat{E_6}$이 성립할 확률은 2^{-5}이 된다.

이상의 논리는 연속된 6단계의 첫 단계에서 어떤 변수가 갱신되던지 무관하며, 따라서 앞에서 가정한 두 번째 사항은 생략되어도 됨을 알 수 있다.

4.2 전체 라운드에의 적용

이번 절에서는 앞 소절의 결과를 SAC P-해쉬함수의 전체 라운드에 적용시켜 보자.

먼저 다음과 같은 조건을 만족하는 $X = (X_i)_{0 \le i \le 15}$와 $\widehat{X} = (\widehat{X_i})_{0 \le i \le 15}$를 정의하자.

$$X_i \text{is arbitrary for } i \neq 8,9,10 \quad (14)$$

$$X_8 = 0x00000016 \quad (15)$$

$$X_9 = X_2 \oplus X_4 \oplus X_{14} \oplus 0x0000000b \quad (16)$$

$$X_{10} = X_{15} \quad (17)$$

$$\widehat{X_i} = X_i \text{ for } i \neq 10,15 \quad (18)$$

〔표 3〕 단계별 메시지 입력 순서

단계	입력	단계	입력	단계	입력	단계	입력
1	X_0	25	X_4	49	X_{23}	73	X_{13}
2	X_1	26	X_{21}	50	X_{14}	74	X_{22}
3	X_2	27	X_{17}	51	X_{19}	75	X_2
4	X_3	28	X_1	52	X_{21}	76	X_{14}
5	X_4	29	X_{23}	53	X_{13}	77	X_3
6	X_5	30	X_{18}	54	X_{15}	78	X_6
7	X_6	31	X_{12}	55	X_{20}	79	X_7
8	X_7	32	X_{10}	56	X_8	80	X_5
9	X_8	33	X_5	57	X_{18}	81	X_{15}
10	X_9	34	X_{16}	58	X_{11}	82	X_0
11	X_{10}	35	X_8	59	X_5	83	X_{18}
12	X_{11}	36	X_0	60	X_4	84	X_{23}
13	X_{12}	37	X_{20}	61	X_7	85	X_{10}
14	X_{13}	38	X_3	62	X_1	86	X_{21}
15	X_{14}	39	X_{22}	63	X_9	87	X_{16}
16	X_{15}	40	X_6	64	X_{12}	88	X_{20}
17	X_{16}	41	X_{11}	65	X_0	89	X_4
18	X_{17}	42	X_{19}	66	X_2	90	X_{17}
19	X_{18}	43	X_{15}	67	X_6	91	X_{12}
20	X_{19}	44	X_2	68	X_{17}	92	X_{19}
21	X_{20}	45	X_7	69	X_{10}	93	X_8
22	X_{21}	46	X_{14}	70	X_{22}	94	X_9
23	X_{22}	47	X_9	71	X_{16}	95	X_{11}
24	X_{23}	48	X_{13}	72	X_3	96	X_1

$$\widetilde{X_{10}} = \widetilde{X_{15}} = X_{15} + 2^{31} \quad (19)$$

식 (15),(16)은 X_8과 X_{18}이 mod 32로 22가 되도록 한 조치이다. 식 (17)~(18)과 같이 X와 $\widetilde{X}$를 정의하고 나면, 3장의 분석에 의하여 X와 $\widetilde{X}$ 사이에는 메시지 확장을 하고 난 후에도 $X = \widetilde{X}_i$ $(16 \le i \le 23)$가 성립하게 된다.

〔표 3〕은 P-해쉬함수의 메시지 입력 순서를 단계별로 표시한 표이다. P-해쉬함수의 총 96 단계 중 우리가 주목하는 단계는 11~16, 32~37, 43~48, 54~59, 69~74, 81~86 단계이다. 각 단계군은 6단계로 구성되어 있고, 6단계 중 첫 단계의 입력 메시지 값은 X_{10}이거나 X_{15}이다. 우리의 목적은 앞 절의 논리를 이용하여 이들 단계군 내에서만 버퍼들 사이에 차분이 발생하고 나머지 단계들에서는 차분이 발생하지 않도록 하는 것이다.

이 중 32~37, 43~48, 54~59, 64~74 단계군에서는 앞 절의 논리가 그대로 적용되어 각각이 2^{-5}의 확률로 차분을 제거할 수 있다. 11~16 단계군에서는 마지막 16단계에서 X_{15}와 $\widetilde{X_{15}}$가 다시 입력으로 들어가기 때문에 앞 절의 논리를 그대로 적용할 수는 없다.

16단계의 연산을 살펴보자.

$$A_{16} = (f_0(A_{15}, B_{15}, C_{15}, D_{15}, E_{15}) + X_{15} + K) \lll s_{16}$$
$$\widetilde{A_{16}} = (f_0(\widetilde{A_{15}}, B_{15}, C_{15}, D_{15}, E_{15}) + \widetilde{X_{15}} + K) \lll s_{16}$$

$A_{16} = \widetilde{A_{16}}$가 성립하기 위해서는 $X_{15} \oplus \widetilde{X_{15}} = 1^{\lll 31}$ 이므로

$$f_0(A_{15}, B_{15}, C_{15}, D_{15}, E_{15}) \oplus f_0(\widetilde{A_{15}}, B_{15}, C_{15}, D_{15}, E_{15}) = 1^{\lll 31} \quad (20)$$

를 만족해야 한다. A_{15}와 $\widetilde{A_{15}}$의 차분은 11단계에서 처음 생기게 되는데, 식 (16)대로 설정한 X_{18}에 의하여 s_{11}은 22가 되고, 따라서 $A_{11} \oplus \widetilde{A_{11}} = 1^{\lll 21}$이 된다. A_{11}과 $\widetilde{A_{11}}$은 12단계에서 다시 10 비트를 순환 이동하기 때문에 $A_{12} \oplus \widetilde{A_{12}} = 1^{\lll 31}$이 되고, 이 값이 15단계에까지 그대로 유지되어 $A_{15} \oplus \widetilde{A_{15}} = 1^{\lll 31}$이 된다. 따라서, f_0가 SAC을 만족하므로 식 (20)을 만족할 확률은 $\frac{1}{2}$이 되고, 결국 앞 절의 주장과 같이 11~16 단계군에서의 버퍼의 차분을 2^{-5}의 확률로 없앨 수 있다. 81~86 단계군도 11~16 단계군과 비슷한 논리를 전개하면 역시 버퍼의 차분을 2^{-5}의 확률로 없앨 수 있다. 이 경우에는, 식 (15)에서 설정한 X_8값이 81단계에서의 순환 이동의 값을 22로 만드는데 쓰이게 된다.

이상을 종합하면, SAC P-해쉬함수에는 2^{-30}의 확률로 충돌쌍이 존재함을 알 수 있다.

4.3 구현(실제 충돌쌍 찾기)

이번 절에서는 이상의 내용을 바탕으로 실제 SAC P-해쉬함수의 충돌쌍을 찾은 과정에 대하여 논한다.

4.3.1 부울 함수 선정

앞에서 기술한 대로 P-해쉬함수의 부울 함수는 SAC을 만족하지 않는다. 따라서, SAC P-해쉬함수의 충돌쌍을 찾기 위해서는 P-해쉬함수의 부울 함수를 SAC을 만족하는 부울 함수로 대체해야 한다. 본 논문의 내용은 특정한 부울 함수의 성질을 이용한 것이 아니므로 SAC을 만족하는 임의의 부울 함수를 사용하더라도 문제가 없을 것으로 예상된다. 따라서, 본 절에서는 P-해쉬함수의 부울 함수들 중 SAC을 만족하는 f_2를 다음과 같이 변형해 사용해 보았다.

$$f_2(x_1, x_2, x_3, x_4, x_5) = x_2 \oplus (x_3 \wedge (x_2 \oplus x_4)) \oplus (((x_2 \wedge x_5) \oplus x_4) \wedge x_1)$$

$$f_2(x_1, x_2, x_3, x_4, x_5) = x_3 \oplus (x_4 \wedge (x_3 \oplus x_1)) \oplus (((x_3 \wedge x_1) \oplus x_5) \wedge x_2)$$

$$f_2(x_1, x_2, x_3, x_4, x_5) = x_1 \oplus (x_2 \wedge (x_1 \oplus x_4)) \oplus (((x_1 \wedge x_4) \oplus x_3) \wedge x_5)$$

$$f_2(x_1, x_2, x_3, x_4, x_5) = x_4 \oplus (x_5 \wedge (x_4 \oplus x_2)) \oplus (((x_4 \wedge x_2) \oplus x_1) \wedge x_3)$$

4.3.2 구현 환경 및 알고리즘

소프트웨어로 프로그램을 구현한 환경은 다음과 같다.

- 프로그래밍 언어 : C 언어
- 프로세서 : PentiumII 450 MHz PC
- O/S : Linux

본 논문은 해쉬함수의 충돌쌍을 찾기 위하여 특정한 알고리즘을 제시한 것이 아니라, 충돌쌍이 발생할 확률을 이론적으로 검증한 것이기 때문에, 실제 충돌쌍을 찾기 위한 특별한 알고리즘이 존재하지 않는다. 또, 이론적으로 제시한 확률이 실제와 부합하는 지를 검증하기 위하여 보다 효율적인 알고리즘을 설계하지 않고, X_i들의 값을 앞 절에서 제시한 조건만 만족하도록 하고 임의로 변화시키면서 충돌쌍이 발생하는지를 관찰하였다.

4.3.3 충돌쌍

실제로 프로그램을 실행시켜 본 결과, 1시간 24분 만에 약 2^{28}회 정도의 연산 후 다음과 같은 충돌쌍을 찾을 수 있었다.

X_0=0xe64ec066	$\widetilde{X}_0$=0xe64ec066
X_1=0xfd126b95	$\widetilde{X}_1$=0xfd126b95
X_2=0x6d80c03e	$\widetilde{X}_2$=0x6d80c03e
X_3=0x09d32e0c	$\widetilde{X}_3$=0x09d32e0c
X_4=0x767d3ff5	$\widetilde{X}_4$=0x767d3ff5
X_5=0x2bc1b633	$\widetilde{X}_5$=0x2bc1b633
X_6=0x40727b94	$\widetilde{X}_6$=0x40727b94
X_7=0xd7e17540	$\widetilde{X}_7$=0xd7e17540
X_8=0x00000016	$\widetilde{X}_8$=0x00000016
X_9=0x278364e1	$\widetilde{X}_9$=0x278364e1
X_{10}=0xe7e7d228	$\widetilde{X}_{10}$=0x67e7d228
X_{11}=0x8014bf7d	$\widetilde{X}_{11}$=0x8014bf7d
X_{12}=0xd5a3b0de	$\widetilde{X}_{12}$=0xd5a3b0de
X_{13}=0x5a70ffd6	$\widetilde{X}_{13}$=0x5a70ffd6
X_{14}=0x3c7e9b21	$\widetilde{X}_{14}$=0x3c7e9b21
X_{15}=0xe7e7d228	$\widetilde{X}_{15}$=0x67e7d228

위 충돌쌍의 해쉬값은 다음과 같다.

0xdfe4e58f 0x1f21fb34 0x9956457f
0x8726dff2 0x0a45bef3

V. 결 론

지금까지 PKC'98에서 신 상욱 등이 제안한 해쉬함수에 대하여 살펴보고, 제안자들이 주장한대로 해쉬함수의 부울 함수들이 SAC을 만족할 경우 충돌쌍이 발생할 확률이 2^{-30}이 되는 것을 이론적으로 검증하고, 실제 충돌쌍을 찾아보았다.

SAC은 암호학에 사용되는 함수가 갖추어야 할 필수적인 성질 중의 하나이지만, 해쉬함수의 부울함수가 SAC을 만족하면 치명적인 약점이 될 수 있음을 본 논문을 통하여 알 수 있었다. 따라서, 앞으로 해쉬함수의 부울함수를 설계할 때에는 이러한 특성을 감안하여 더욱 신중을 기해야 할 것을 권고하면서 본 논문을 마치고자 한다.

참 고 문 헌

[1] Alfred J. Menezes, Paul C. van Oorchot, Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, 4th edition, 1999.

[2] R. Rivest, The MD4 message-digest algorithm, Request For Comments(RFC) 1320, Internet Activities Board, Internet Privacy Task Force, April 1992.

[3] R. Rivest, The MD5 message-digest algorithm, Request For Comments(RFC) 1320, Internet Activities Board, Internet Privacy Task Force, April 1992.

[4] NIST, Secure hash standard, FIPS 180-1, US Department of Commerce, Washington D.C., April 1995.

[5] Y. Zheng. J. Pieprzyk, J. Seberry, HAVAL -a one-way hashing algorithm with variable length and output, Advances in Cryptology-Auscrypt'92, Lecture Notes in Computer Science, vol. 718, Springer-Verlag, pp. 83~104, 1993.

[6] Sang Uk Shin, Kyung Hyune Rhee, Dae Hyun Ryu, Sang Jin Lee, A New Hash Function Based on MDx-family and Its Application to MAC, Public Key Cryptography '98, pp. 234~246, 1998.

[7] A. F. Webster, S. E. Tavares, On the design of S-boxes, Advances in Cryptology-Crypto'85, Lecture Notes in Computer Science vol. 218, Springer-Verlag, pp. 523~534, 1986.

〈著者紹介〉

한 대 완 (Daewan Han) 정회원
1995년 2월 : 서울대학교 수학과(학사)
1997년 2월 : 서울대학교 수학과(석사)
1998년 2월~2001년 1월 : 공군기상전대 수치예보개발장교
2001년 3월~현재 : 국가보안기술연구소 연구원

박 상 우 (Sangwoo Park) 정회원
1989년 2월 : 고려대학교 수학교육과 졸업
1991년 8월 : 고려대학교 수학과 석사
1991년 8월~1999년 12월: 한국전자통신연구원 선임연구원
2000년 1월~현재: 국가보안기술연구소 선임연구원

지 성 택 (Seongtaek Chee) 정회원
1985년 2월 : 서강대학교 수학과 졸업
1987년 2월 : 서강대학교 수학과 석사
1999년 2월 : 고려대학교 수학과 박사
1989년~1999년 12월: 한국전자통신연구원 선임연구원
2000년 1월~현재 : 국가보안기술연구소 책임연구원