情 報 保 護 學 會 誌
第 16 卷 第 4 號, 2006. 8

해쉬함수에 대한 충돌쌍 탐색 공격의 동향

성 수 학*

요 약

중국의 Wang 교수 등은 2004년부터 차분 공격을 이용하여 대표적인 해쉬함수인 MD4, MD5, RIPEMD, HAVAL, SHA-0에 대한 충돌쌍을 찾았다. 그들은 아직까지 SHA-1에 대한 충돌쌍을 찾지는 못했지만 생일 공격보다 빠른 방법으로 SHA-1의 충돌쌍을 찾을 수 있음을 이론적으로 보였으며 58단계 SHA-1(SHA-1의 전체는 80단계)에 대해서는 구체적인 충돌쌍을 찾았다. 본 논문에서는 Wang 교수 등이 개발한 차분 공격법에 대해서 살펴보기로 한다.

Ⅰ. 서 론

해쉬함수(hash function)는 다양한 크기의 입력을 받아 고정된 짧은 크기의 출력(128~ 256비트)을 생성하는 함수이다. 해쉬함수는 디지털 서명(digital signature)시 데이터 무결성(integrity)을 해결하는데 사용된다. 해쉬함수에 의해서 메시지를 축약하고 축약된 메시지를 서명함으로써 원래 메시지의 서명과 비교할 때 필요한 계산량, 메모리, 전송량을 크게 줄일 수 있다.

1990년에 접어들면서 해쉬 기능만을 목적으로 하는 전용 해쉬함수들이 개발되었다. RSA Data Security 사의 R.L. Rivest는 전용 해쉬함수인 MD4를 개발하였다[18]. 그 이후 MD4를 개선한 많은 전용 해쉬함수인 MD 계열 해쉬함수들(예를 들어 MD5[19], HAVAL[26], RIPEMD[17], SHA-0[14], SHA-1[15], HAS-160[2] 등)이 개발되었다.

해쉬함수의 공격에는 세 종류가 있다. 첫째로 주어진 해쉬값을 갖는 입력을 찾는 "preimage attack", 둘째로 의미 있는 입력의 해쉬값과 동일한 해쉬값을 갖는 새로운 입력을 찾는 "2-nd preimage attack"(target attack이라고도 함) 마지막으로 동일한 해쉬값을 갖는 입력쌍(충돌쌍)을 찾는 "collision attack"이 있다. 이 세 가지 공격법 중에서 preimage attack이 공격하기 가장 어려운 방법이며 collision attack이 공격하기 가장 쉬운 방법이다. Collision attack이 되었다고 해서 의미 있는 입력의 해쉬값과 동일한 해쉬값을 갖는 다른 입력을 항상 찾을 수는 없지만 collision attack이 성공하면 해쉬함수는 공격되었다고 말한다.

해쉬함수의 개발과 더불어 개발된 해쉬함수들에 대한 다양한 공격 방법이 개발되었다. MD4 해쉬함수는 3라운드로 구성되어 있다. Merkle은 처음 2라운드의 충돌쌍을 den Boer와 Bosselaers[7]는 마지막 2라운드에 대한 충돌쌍을 그리고 Dobbertin[9,11]은 차분 공격을 이용하여 MD4의 완전한 충돌쌍을 찾았다. MD5 해쉬함수는 MD4를 확장한 것으로 라운드의 수가 3에서 4로 증가된 해쉬함수이다[19]. den Boer와 Bosselaers[8]는 MD5의 초기값을 바꾸었을 때 MD5의 의사 충돌쌍을 찾았으며 Dobbertin[10]은 초기값을 바꾼 MD5에 대한 충돌쌍을 찾았다. HAVAL 해쉬함수는 pass 수에 따라 3-pass, 4-pass, 5-pass 세 종류가 있다[26]. 3-pass HAVAL은 3라운드로 구성된 해쉬함수이나 Kasselman과 Penzhorn[12]은 마지막 2라운드만을 고려하였을 때 박상우, 성수학, 지성택, 임종인[16]은 처음 2라운드만을 고려하였을 때 충돌쌍을 찾았으며 van Rompay, Biryukov, Preneel, Vandewalle[20]은 3-pass HAVAL에 대한 완전한 충돌쌍을 찾았다. 미국 표준 해쉬함수인 SHA-1[15]의 전신인 SHA-0[14]에 대한 충돌쌍

* 배재대학교 전산정보수학과(sungsh@mail.pcu.ac.kr)

공격도 시도되었다. Chabaud와 Joux[6]는 차분 공격을 이용하여 축소 SHA-0에 대한 충돌쌍을 찾았으며, Biham과 Chen[4]은 의사 충돌쌍(near collision)을 찾았으며, Biham et al.[5] 및 Wang et al.[24]은 SHA-0에 대한 완전한 충돌쌍을 찾았다.

2004년부터 중국의 Xiaoyun Wang교수 등은 새로운 차분 공격법을 개발하였다[21-24]. 이 공격법은 대부분의 MD 계열 해쉬함수에 대한 충돌쌍을 찾는데 적용할 수 있는 획기적인 방법이며 현재까지 알려진 공격법 중에서 가장 강력한 것이다. Wang 등은 MD4, MD5, RIPEMD, HAVAL, SHA-0의 충돌쌍을 찾았다[21-24]. 그들은 아직까지 SHA-1에 대한 충돌쌍을 찾지는 못했지만 생일 공격보다 빠른 방법으로 SHA-1의 충돌쌍을 찾을 수 있음을 이론적으로 보였으며 58단계 SHA-1(SHA-1의 전체는 80단계)에 대해서는 구체적인 충돌쌍을 찾았다[22].

본 논문에서는 Wang 교수 등이 개발한 차분 공격법에 대해서 살펴보기로 한다. 2절에서는 해쉬함수의 요구조건을 3절에서는 Wang 등의 차분 공격법을 살펴보기로 한다. 4절에서는 Wang 등의 공격법을 MD4에 적용하며 끝으로 5절에서는 결론을 맺기로 한다.

II. 해쉬함수의 안전성

해쉬함수(hash function)는 다양한 크기의 입력을 받아 고정된 짧은 크기의 출력(128~ 256비트)을 생성하는 함수이다. 본 절에서는 해쉬함수의 요구조건과 안전성에 대해서 살펴보기로 한다.

해쉬함수는 다음 두 성질을 만족하여야 한다.

- 일방향성(one-way): 주어진 출력에 대하여 입력값을 구하는 것이 계산상 불가능하다. 즉 $y=h(x)$가 주어졌을 때 x를 계산하는 것은 계산상 불가능하다.
- 충돌회피성(collision-free): 같은 출력을 갖는 서로 다른 입력 메시지를 찾는 것이 계산상 불가능하다. 즉 $h(x)=h(x')$이며 $x\neq x'$인 충돌쌍 (x,x')을 찾는 것은 계산상 불가능하다.

충돌회피성보다 강한 조건으로는 "preimage resistance"와 "second preimage resistance"가 있다.

- preimage resistance: 입력 x는 모르고 임의의 해쉬값 $y=h(x)$이 주어질 때 같은 해쉬값을 갖는 입력x' $(h(x)=h(x'))$을 찾는 것은 계산상 불가능하다.
- second preimage resistance: 임의의 입력 x에 대해서 $h(x)=h(x')$을 만족하는 $x'(x'\neq x)$을 찾는 것은 계산상 불가능하다.

해쉬함수에 대한 전수 조사 공격(brute-force attack)을 생일 공격(birthday attack)이라고 한다. 생일 문제와 관련된 두 개의 결과를 살펴보자. 이 두 결과는 확률론에서 잘 알려진 결과([13] P. 53 또는 [1] P. 207 참조)이며 해쉬함수 및 메시지 인증의 안전성을 측정하는데 사용되는 중요한 결과이다.

정리 2.1. 1에서 n까지 번호가 적힌 n개의 공이 항아리에 들어 있다고 하자. 항아리에서 k개의 공을 복원 추출하여 공의 번호를 기록한다고 할 때 같은 번호가 나올 확률은 다음과 같다.

$$1-\frac{n(n-1)\cdots(n-k+1)}{n^k}$$

특히 $k=O(\sqrt{n})$일 때 같은 번호가 나올 확률은 다음과 같이 근사값으로 간단히 표시할 수 있다.

$$1-\frac{n(n-1)\cdots(n-k+1)}{n^k}\approx 1-\exp(-\frac{k^2}{2n})$$

위의 정리에서 $n=365$일 때가 바로 원래의 생일 문제로, 23명이 있으면 그 중에 생일이 같은 사람이 있을 확률은 $1-\exp(-\frac{23^2}{2\times 365})\approx 0.51551$이다. 또 해쉬값의 크기가 n비트(총 2^n개의 해쉬값 가능)인 해쉬함수에서 $2^{(n+1)/2}$개의 메시지가 있을 때, 어떤 두 개의 메시지가 같은 해쉬값을 가질 확률은 $1-\exp(-1)\approx 0.63$이다.

정리 2.2. 1에서 n까지 번호가 적힌 n개의 흰 공이 들어 있는 항아리와 1에서 n까지 번호가 적힌 n개의 빨간 공이 들어 있는 항아리가 있다고 하자. 두 항아리에서 각

각 k_1, k_2개의 공을 복원 추출하여 공의 번호를 기록한다고 하자. 두 항아리에서 추출한 공의 번호가 같을(같은 번호를 갖는 흰 공과 빨간 공) 확률은 다음과 같다.

$$1-\frac{1}{n^{k_1+k_2}}\sum_{t_1,t_2} n(n-1)$$

$$\cdots (n-t_1-t_2+1)\,T(k_1,t_1)\,T(k_2,t_2)$$

여기서 $T(n,t)$는 원소의 개수가 n인 집합을 공집합인 아닌 t개의 부분 집합으로 분할하는 경우의 수이다.

특히 n이 충분히 크고 $k_1=k_2=k, k=O(\sqrt{n})$일 때 두 항아리에서 추출한 공의 번호가 같을 확률은 다음과 같이 근사값으로 간단히 표시할 수 있다.

$$1-\frac{1}{n^{2k}}\sum_{t_1,t_2} n(n-1)$$

$$\cdots (n-t_1-t_2+1)\,T(k,t_1)\,T(k,t_2) \approx 1-\exp(-\frac{k^2}{n})$$

위의 정리를 해쉬함수에 적용해 보자. 해쉬값의 크기가 n비트인 해쉬함수에서 $2^{n/2}$개의 메시지를 갖는 두 집합 사이에서 동일한 해쉬값이 존재할 확률은 $1-\exp(-1)\approx 0.63$이다.

위의 두 정리로부터 해쉬값의 크기가 n비트인 해쉬함수를 고려할 때, $2^{n/2}$개 정도의 메시지가 있으면 해쉬값이 같은 메시지(충돌쌍)가 존재할 가능성이 크다. 이런 이유로 해쉬값의 크기가 n비트인 해쉬함수의 안전성을 $2^{n/2}$으로 설정하고, 이를 해쉬함수에 대한 전수 조사 공격의 어려움으로 설정한다. 최근에는 컴퓨터의 발달로 인하여 계산상 불가능의 기준을 2^{80} 이상의 계산 복잡도가 요구되는 것으로 설정하며, 이 기준에 따라 해쉬값의 크기(해쉬함수의 출력 크기)를 160비트 이상으로 하여야 한다.

Ⅲ. Wang 등의 차분 공격법

본 절에서는 Wang 등이 해쉬함수의 충돌쌍을 찾기 위해서 사용한 차분 공격법에 대해서 살펴보기로 한다. 먼저 차분 공격을 기술하는데 필요한 기호를 살펴보자.

- $M=(m_0,m_1,\cdots,m_{15})$, $M'=(m_0',m_1',\cdots,m_{15}')$: 두 개의 512비트 블록
- a_i,b_i,c_i,d_i : M을 작용하였을 때 단계 i에서의 연쇄 변수
- a_i',b_i',c_i',d_i' : M'을 작용하였을 때 단계 i에서의 연쇄 변수
- $x_i=(x_{i,32},\cdots,x_{i,1})$: 워드 x_i의 비트 표시 (단, $x_{i,32}$는 MSB, $x_{i,1}$은 LSB)
- Δx_i : 두 워드 x_i'과 x_i의 덧셈에 대한 차분, 즉 $\Delta x_i = x_i' - x_i \pmod{2^{32}}$
- $x_i[+j]$: M대신 M'을 작용하였을 때 연쇄 변수 x_i는 j번째 비트만 바뀜(0에서 1로)
- $x_i[-j]$: M대신 M'을 작용하였을 때 연쇄 변수 x_i는 j번째 비트만 바뀜(1에서 0으로)
- $x_i' = x_i[\pm j_1,\cdots,\pm j_l]$: M대신 M'을 작용하였을 때 연쇄 변수 x_i는 $j_1,\cdots,j_l$번째 비트만 바뀜

위에서 정의한 $x_i' = x_i[\pm j_1,\cdots,\pm j_l]$을 다음과 같이 쓸 수 있다.

$$x'_{i,k} - x_{i,k} = \begin{cases} \pm 1 \; if \;\; k=j_1,\cdots,j_l \\ 0 \quad otherwise \end{cases}$$

구체적인 예제를 살펴보자.

예제 3.1. $x_i' = x_i[2,-3,4]$라고 하자. 이 경우 x_i'과 x_i는 2,3,4 번째 위치에서만 비트값이 서로 다르다. x_i'의 2 번째 비트값은 1이며 x_i의 2 번째 비트값은 0이다. x_i'의 3 번째 비트값은 0이며 x_i의 3 번째 비트값은 1이다. x_i'의 4 번째 비트값은 1이며 x_i의 4 번째 비트값은 0이다. 즉

$$x_{i,2}' = 1,\; x_{i,2} = 0,\; x_{i,3}' = 0,\; x_{i,3} = 1,\; x_{i,4}' = 1,$$

$$x_{i,4} = 0,\; x_{i,k}' = x_{i,k} \quad if \;\; k \neq 2,3,4$$

Wang 등이 $x_i' = x_i[\pm j_1,\cdots,\pm j_l]$과 같은 기호를 고려한 이유는 덧셈에 대한 차분 뿐만 아니라 두 워

드에서 차이가 나는 비트의 정확한 위치까지도 언급하기 위해서다. 이런 기호를 사용함으로써 다음 단계의 차분을 보다 간단히 계산할 수 있다.

s비트 좌순환($x^{<<s}$, left rotation by s bits)은 많은 해쉬함수에서 사용되는 중요한 연산이다. x를 32비트 수라 하고 $x=(x_{32},\cdots,x_1)$로 표기하기로 하자(일반적인 표기와는 다르며 일반적으로는 $x=(x_{31},\cdots,x_0)$으로 표기함). 여기서 x_1은 x의 최하위 비트이고 x_{32}는 x의 최상위 비트이다. 이 때 x의 s비트 좌순환($x^{<<s}$)은 다음과 같다.

$$x^{<<s}=(x_{32-s},\cdots,x_1,x_{32},\cdots,x_{32-s+1})$$

다음 두 정리로부터 알 수 있듯이 좌순환 연산에 대한 차분을 계산할 때 캐리가 발생한다. 덧셈에 대한 캐리를 고려하여 다음 두 정리를 증명할 수 있으며 여기서는 증명을 생략하기로 한다.

정리 3.1. $x'-x=k$일 때 s비트 좌순환 $x^{<<s}$의 차분은 다음과 같다.

$$x'^{<<s}-x^{<<s}=k^{<<s}-c_{33}^{<<s}+c_{33-s}$$

여기서 c는 $x+k$의 캐리이다.

정리 3.2. $x'-x=-k$일 때 s비트 좌순환 $x^{<<s}$의 차분은 다음과 같다.

$$x'^{<<s}-x^{<<s}=-k^{<<s}+c_{33}^{<<s}-c_{33-s}$$

여기서 c는 $x'+k$의 캐리이다.

Wang 등이 고려한 방법으로 차분을 계산할 때는 캐리가 발생하지 않는다.

정리 3.3. $x_i'=x_i[\pm j_1,\pm j_2,\cdots,\pm j_l]$ $(j_1<j_2<\cdots<j_l)$일 때 차분과 좌순환 연산에 대한 차분은 다음과 같다.

$$x_i'-x_i=\pm 2^{j_1-1}\pm 2^{j_2-1}\pm\cdots\pm 2^{j_l-1}$$

$$x_i'^{<<s}-x_i^{<<s}=[\pm(j_1+s),\pm(j_2+s),\cdots,\pm(j_l+s)]$$

구체적인 예제를 통해서 좌순환 연산에 대한 일반적인 차분과 Wang 등이 고려한 차분을 비교해 보자.

예제 3.2. $x_i'=x_i[2,-3,4]$ 일때$x_i'=(x_{i,32},\ \cdots,\ x_{i,5},\ 1,0,1,x_{i,1})$, $x_i=(x_{i,32},\cdots,x_{i,5},0,1,0,x_{i,1})$ 따라서 차분과 좌순환 연산에 대한 차분은 다음과 같이 쉽게 계산할 수 있다.

$$x_i'-x_i=2^3-2^2+2=6$$

$$x_i'^{\ll 30}=(1,x_{i,1},x_{i,32},\cdots,x_{i,5},1,0)$$

$$x_i^{\ll 30}=(0,x_{i,1},x_{i,32},\cdots,x_{i,5},0,1)$$

$$x_i'^{\ll 30}-x_i^{\ll 30}=-1+2+2^{31}$$

$$=[(2+30),-(3+30),(4+30)]=[32,-1,2]$$

두 워드의 차이가 나는 비트의 위치를 언급하지 않고 단순히 x_i'과 x_i의 차분이 6, 즉 $x_i'-x_i=6$일 때는 정리 3.1에 의해서 $x_i'^{\ll 30}$과 $x_i^{\ll 30}$의 차분 계산에서 캐리가 발생한다. 하지만 Wang 등의 방법으로는 $x_i'^{\ll 30}$과 $x_i^{\ll 30}$의 차분 계산에서 캐리가 발생하지 않을 뿐만 아니라 차이가 나는 비트의 정확한 위치도 알 수 있다.

Wang 등의 방법의 또 다른 특성을 살펴보자. 첫째 라운드 내의 단계에서 작용하는 메시지가 독립변수일 때는 그 단계에서의 차분 특성 확률을 1로 할 수 있다. MD4 해쉬함수의 첫째 라운드 내의 단계 i의 연쇄변수 b_i는 다음과 같이 계산된다.

$$b_i=(a_{i-1}+f(b_{i-1},c_{i-1},d_{i-1})+x_i)^{<<s}$$

단계 i에서 작용하는 메시지 변수 x_i가 독립변수이면 x_i를 변형하여 b_i가 원하는 조건을 만족하게끔 할 수 있다. 이런 방법을 basic modification이라고 한다. 예를 들어 b_i의 $j_1,\cdots,j_k$의 비트값을 $u_{j_1},\cdots,u_{j_k}$로 바꾸고 싶을 때 x_i를 다음과 같이 변형하면 된다.

$b_i \leftarrow b_i \oplus \left((b_{i,j_1} \oplus u_{j_1}) * 2^{j_1-1} \oplus \cdots \oplus (b_{i,j_k} \oplus u_{j_k}) * 2^{j_k-1}\right)$

$x_i \leftarrow b_i^{\gg s} - \left(a_{i-1} + f(b_{i-1}, c_{i-1}, d_{i-1})\right)$

Basic modification을 사용하기 위해서는 메시지 변수 x_i가 독립변수이어야 한다. MD4, MD5, SHA-1 해쉬함수들은 단계 1부터 단계 16까지 작용하는 메시지 변수가 모두 독립변수이므로 이러한 해쉬함수에 대해서는 단계 1부터 단계 16까지 차분 특성 확률이 1이 되게 할 수 있다. 하지만 국내 표준 해쉬함수인 HAS-160은 단계 1부터 단계 16까지 작용하는 메시지 모두는 독립변수가 아니므로 basic modification을 사용하여 단계 16까지 차분 특성 확률을 1로 할 수 없어서 HAS-160은 다른 해쉬함수에 비해서 Wang 등의 공격에 보다 안전한 것으로 여겨진다.

차분 공격을 이용하여 해쉬함수에 대한 충돌쌍을 찾는 연구는 이미 오래 전부터 행하여 졌다. 물론 이전에서 Wang 등이 고려한 것과는 달리 단순히 차분만을 고려하였다. Berson은 차분 공격을 이용하여 1라운드 MD5를 공격하였다[3]. Dobbertin[9,11]도 차분 공격을 이용하여 MD4를 완전히 공격하였으나 공격 방법은 Wang 등[21]의 것에 비교하면 아주 복잡하다. 다음 절에서는 Wang 등의 방법으로 MD4의 충돌쌍을 찾는 과정을 보다 상세히 살펴보기로 한다.

Ⅳ. 해쉬함수에 대한 충돌쌍 탐색

Wang 등은 2005년부터 차분 공격을 이용하여 대표적인 해쉬함수인 MD4, MD5, RIPEMD, HAVAL, SHA-0에 대한 충돌쌍을 찾았다. 본 절에서는 Wang 등의 방법을 MD4에 적용하여 충돌쌍을 찾는 과정을 설명하고자 한다.

4.1. MD4 해쉬함수

여기서는 차분 공격을 이용하여 MD4의 충돌쌍을 찾는 방법을 설명하는데 필요한 MD4의 구조에 대해서 살펴보기로 한다. 특히 512비트 메시지 블록이 MD4의 기초해쉬함수에 어떻게 작용하는지를 살펴보기로 한다.

512비트 메시지 블록을 $M = \mathrm{m_i}, 0 \le \mathrm{i} \le 15$이라고 두자. MD4의 4개의 32비트 초기값(initial value)은 다음과 같다.

a_0=0x67452301, b_0=0xefcdab89,
c_0=0x98badcfe, d_0=0x10325476

MD4는 3개의 라운드로 구성되어 있으며 각 라운드는 다시 16단계로 구성되어 있다. 첫째, 둘째, 셋째 라운드에 작용하는 부울함수는 각각 다음과 같다.

$f(x,y,z) = (x \wedge y) \vee (\neg x \wedge z)$
$g(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$
$h(x,y,z) = x \oplus y \oplus z$

단계 $i-1$의 상태값으로부터 단계 i의 상태값은 다음과 같이 계산된다.

$$(a,b,c,d) \leftarrow (d, (a + f(b,c,d) + x + k)^{<<s}, b, c)$$

즉, 단계 $i(1 \le i \le 48)$의 상태값을 (a_i, b_i, c_i, d_i)라고 할 때

$a_i = d_{i-1}$
$c_i = b_{i-1}$
$d_i = c_{i-1}$
$b_i = (a_{i-1} + f(b_{i-1}, c_{i-1}, d_{i-1}) + x + k)^{<<s}$

MD4의 기초해쉬함수는 총 48단계로 구성되어 있으며 단계별 계산 과정은 다음과 같다.

초기 단계: $a \leftarrow a_0,\ b \leftarrow b_0, c \leftarrow c_0,\ d \leftarrow d_0$

단계 1: $(a,b,c,d) \leftarrow (d, (a + f(b,c,d) + m_0)^{\ll 3}, b, c)$

단계 2: $(a,b,c,d) \leftarrow (d, (a + f(b,c,d) + m_1)^{\ll 7}, b, c)$

단계 3: $(a,b,c,d) \leftarrow (d, (a + f(b,c,d) + m_2)^{\ll 11}, b, c)$

단계 4: $(a,b,c,d) \leftarrow (d, (a + f(b,c,d) + m_3)^{\ll 19}, b, c)$

단계 5: $(a,b,c,d) \leftarrow (d, (a + f(b,c,d) + m_4)^{\ll 3}, b, c)$

단계 6: $(a,b,c,d) \leftarrow (d, (a + f(b,c,d) + m_5)^{\ll 7}, b, c)$

단계 7: $(a,b,c,d) \leftarrow (d, (a + f(b,c,d) + m_6)^{\ll 11}, b, c)$

단계 8: $(a,b,c,d) \leftarrow (d,(a+f(b,c,d)+m_7)^{\lll 19}, b,c)$

$\vdots$

단계 41: $(a,b,c,d) \leftarrow (d,(a+h(b,c,d)+m_1 + 0x6ed9eba1)^{\lll 3},b,c)$

단계 42: $(a,b,c,d) \leftarrow (d,(a+h(b,c,d)+m_9 + 0x6ed9eba1)^{\lll 9},b,c)$

단계 43: $(a,b,c,d) \leftarrow (d,(a+h(b,c,d)+m_5 + 0x6ed9eba1)^{\lll 11},b,c)$

단계 44: $(a,b,c,d) \leftarrow (d,(a+h(b,c,d)+m_{13} + 0x6ed9eba1)^{\lll 15},b,c)$

단계 45: $(a,b,c,d) \leftarrow (d,(a+h(b,c,d)+m_3 + 0x6ed9eba1)^{\lll 3},b,c)$

단계 46: $(a,b,c,d) \leftarrow (d,(a+h(b,c,d)+m_{11} + 0x6ed9eba1)^{\lll 9},b,c)$

단계 47: $(a,b,c,d) \leftarrow (d,(a+h(b,c,d)+m_7 + 0x6ed9eba1)^{\lll 11},b,c)$

단계 48: $(a,b,c,d) \leftarrow (d,(a+h(b,c,d)+m_{15} + 0x6ed9eba1)^{\lll 15},b,c)$

마지막 단계: $(a,b,c,d) \leftarrow (a+a_0, b+b_0, c+c_0, d+d_0)$

4.2. MD4 해쉬함수의 충돌쌍 탐색

먼저 MD4의 단계 $i(1 \le i \le 48)$에서의 차분 특성을 계산해 보자. 3절에서 정의한 기호를 사용하기로 하며, 편의상 단계 i에 작용하는 메시지 워드 및 상수를 각각 x_i와 k_i로 표기하기로 한다. 또 $f(b_{i-1},c_{i-1},d_{i-1})$ 및 $f(b_{i-1}',c_{i-1}',d_{i-1}')$를 각각 f 및 f'으로 표기하기로 한다.

$b_i = (a_{i-1}+f(b_{i-1},c_{i-1},d_{i-1})+x_i+k_i)^{<<s}$이므로

$$\begin{aligned} b_i'^{\gg s} &= a_{i-1}' + f' + x_i' + k_i \\ &= a_{i-1} + f + x_i + k_i + (a_{i-1}' - a_{i-1}) \\ &\quad + (f' - f) + (x_i' - x_i) \\ &= b_i^{\gg s} + (a_{i-1}' - a_{i-1}) + (f' - f) + (x_i' - x_i) \end{aligned}$$

따라서 단계 i의 차분 특성은 다음과 같다.

- $a_i' - a_i = d_{i-1}' - d_{i-1}$
- $b_i'^{\gg s} - b_i^{\gg s} = (a_{i-1}' - a_{i-1}) + (f' - f) + (x_i' - x_i)$
- $c_i' - c_i = b_{i-1}' - b_{i-1}$
- $d_i' - d_i = c_{i-1}' - c_{i-1}$

이젠 M 및 M'이 MD4의 충돌쌍이 되기 위한 차분 특성을 찾아보자. Wang 등이 사용한 것과 같이 다음 조건을 만족하는 두 개의 메시지 블록을 고려하기로 한다.

$$m_1' - m_1 = 2^{31},\ m_2' - m_2 = 2^{31} - 2^{28},$$

$$m_{12}' - m_{12} = -2^{16}, m_i' = m_i (i \neq 1,2,12)$$

단계 1. $a_0' = a_0,\ b_0' = b_0,\ c_0' = c_0,\ d_0' = d_0,\ s = 3,$ $x_1 = m_0$이므로 $f' = f, x_1' = x_1$이다. 따라서 단계 1에서의 차분 특성은 $a_1' = a_1,$ $b_1' = b_1,\ c_1' = c_1,\ d_1' = d_1$이다.

단계 2. $a_1' = a_1,\ b_1' = b_1,\ c_1' = c_1,\ d_1' = d_1,\ s = 7,$ $x_1 = m_1$이므로 $f' - f = 0,\ x_1' - x_1 = 2^{31}$이다. $(b_2^{\gg 7})_{32} = 0$으로 선택하면 $(b_2'^{\gg 7})_{32} = 1$이다. 즉

```
번지:32 31 30 29 28 … 1
b2^≫7 = 0  *  *  *  * … *
+) (a1'-a1)+(f'-f)+(x'-x) = 1
-----------------------------
b2'^≫7 = 1  *  *  *  * … *
```

따라서 $b_2' = b_2[(32+7)] = b_2[7]$이 되기 위한 충분조건은 $b_{2,7} = 0$이다. 또한 $a_2' = a_2,\ c_2' = c_2,\ d_2' = d_2$이다. 여기서 $b_{2,7} = 1$로 선택할 수도 있으나 이 경우 $b_2' = b_2[-7]$이다.

단계 3.

$a_2' = a_2,\ b_2' = b_2[7],\ c_2' = c_2,\ d_2' = d_2,\ s = 11,$ $x_3 = m_2$이므로 $x_3' - x_3 = 2^{31} - 2^{28}$이다. f'과 f는 7번째 이외의 위치에서는 같은 값을 갖는다. $f(b,c,d) = bc \oplus bd \oplus d$이므로 f'과 f의 7번째 비트 값은 다음과 같다.

$f(b_{2,7}', c_{2,7}', d_{2,7}') = b_{2,7}'c_{2,7}' \oplus b_{2,7}'d_{2,7}' \oplus d_{2,7}' =$

$c_{2,7} \oplus d_{2,7} \oplus d_{2,7} = c_{2,7}$

$f(b_{2,7}, c_{2,7}, d_{2,7}) = b_{2,7}c_{2,7} \oplus b_{2,7}d_{2,7} \oplus d_{2,7} = d_{2,7}$

$c_{2,7} = d_{2,7}$로 선택하면 f'과 f의 7번째 비트값도 같다. 또한$(b_3^{\gg 11})_{29} = 1, (b_3^{\gg 11})_{32} = 0$으로 선택하면 $(b_3'^{\gg 11})_{29} = 0, (b_3'^{\gg 11})_{32} = 1$이다. 즉

	번지:32 31 30 29 28 … 1
	$b_3^{\gg 11} = 0 \quad * \quad * \quad 1 \quad * \ldots *$
+)	$(a_2' - a_2) + (f' - f) + (x' - x) = 1 \quad * \quad * - 1$
	$b_3'^{\gg 11} = 1 \quad * \quad * \quad 0 \quad * \ldots *$

따라서 $b_3' = b_3[-(29+11), (32+11)] = b_3[-8, 11]$이 되기 위한 충분조건은$b_{3,8} = 1, b_{3,11} = 0, c_{2,7} = d_{2,7}$이다. 또한 $a_3' = a_3, c_3' = c_3[7], d_3' = d_3$이다.

위와 같은 방법으로 단계별 차분 특성과 차분 특성이 만족되게 하는 충분조건을 찾을 수 있다. 단계 41 이후에서의 차분 특성을 살펴보자.

단계 41. $a_{40}' = a_{40}[-32]$, $b_{40}' = b_{40}$, $c_{40}' = c_{40}$, $d_{40}' = d_{40}$, $s = 3$, $x_{41} = m_1$이 므 로 $x_{41}' - x_{41} = 2^{31}$이다. f'과 f는 모든 위치에서 같은 값을 갖는다. 즉 $f' - f = 0$이다. 따라서 아무런 추가 조건 없이 $b_{41}' = b_{41}$이다 (다음 그림 참조). 또한 $a_{41}' = a_{41}, c_{41}' = c_{41}$, $d_{41}' = d_{41}$이다.

	번지:32 31 30 29 28 … 1
	$b_{41}^{\gg 3} = * \quad * \quad * \quad * \quad * \quad \ldots *$
	$a_{40}' - a_{40} = -1$
	$f' - f =$
+)	$x_{41}' - x_{41} = 1$
	$b_{41}'^{\gg 3} = * \quad * \quad * \quad * \quad * \quad \ldots *$

단계 41에서의 차분이 0(즉, $a_{41}' = a_{41}$, $b_{41}' = b_{41}$, $c_{41}' = c_{41}$, $d_{41}' = d_{41}$)이며, 단계 42부터 단계 48까지 작용하는 메시지 차분이 0(즉, $x' = x$)이므로 단계 42부터 단계 48까지의 차분도 모두 0이다. 단계 48에서의 차분이 0이므로 위에서 구성한 차분 특성을 만족하는 메시지 블록 M과 M'을 찾으면 M과 M'은 충돌쌍이 된다. 한편 M과 M'은 다음과 같은 조건을 만족하므로 M을 찾으면 M'은 저절로 찾게 된다.

$m_1' - m_1 = 2^{31}$, $m_2' - m_2 = 2^{31} - 2^{28}$,

$m_{12}' - m_{12} = -2^{16}$, $m_i' = m_i (i \neq 1, 2, 12)$

각 단계별로 차분 특성을 만족하게 하는 충분조건을 찾았으며 이러한 충분조건을 모두 만족하게 하는 메시지 M을 찾으면 M과 M에 대응되는 M'은 충돌쌍이 된다.

이젠 충분조건을 모두 만족하게 하는 메시지 M을 찾는 방법에 대해서 살펴보자. MD4의 첫째 라운드 내에서 작용하는 16개의 메시지 워드들은 서로 독립이므로 3절에서 언급한 basic modification에 의해서 첫째 라운드에서는 확률 1로 충분조건을 모두 만족하게 하는 메시지 M을 찾을 수 있다. 예를 들어 단계 2의 차분 특성을 만족하게 하는 충분조건은 $b_{2,7} = 0$이며, 다음과 같이 m_1을 조절하여 $b_{2,7} = 0$되게 할 수 있다.

$b_2 \leftarrow b_2 \oplus \left((b_{2,7} \oplus 0) * 2^6\right)$,

$m_1 \leftarrow b_2^{\gg 7} - \left(a_1 + f(b_1, c_1, d_1)\right)$

둘째 라운드 및 셋째 라운드 내의 단계에서의 충분조건을 만족하게 하는 메시지M을 확률 1로 찾을 수는 없다, 왜냐하면 이미 첫째 라운드 내의 단계에서의 충분조건을 만족하게끔 메시지를 조절하였기 때문이다. 이 경우 충분조건을 추가로 부여하여 메시지를 조절(advanced modification)하는 방법도 있으나 일반적으로 반복 시행으로 충분조건을 만족하게 하는 메시지를 찾는다. 즉, 랜덤한 512비트 메시지를 선택하면 첫째 라운드 후에는 basic modification에 의해서 메시지가 약간 조정된다. 조정된 메시지가 둘째 라운드 및 셋째 라운드 내의 충분조건을 모두 만족하는지 조사한다. 랜덤한 메시지를 적용하는 횟수는 둘째 및 셋째 라운드의 차분 특성 확률의 역수

정도이다. 따라서 차분 특성 확률이 클수록 충분조건을 모두 만족하게 하는 메시지를 빨리 찾을 수 있다.

V. 결 론

본 논문에서서 최근에 Wang 교수 등이 개발한 차분 공격법을 살펴보았다. 차분 공격법은 현재 가장 강력한 공격법으로 대부분의 전용 해쉬함수에 적용 가능하며 국내표준 해쉬함수인 HAS-160에도 적용 가능하다. HAS-160의 안전성을 평가하기 위해서 차분 공격에 얼마나 안전한지를 연구해야 할 것이다. 또한 차분 공격에 안전한 새로운 해쉬함수를 개발하는 연구도 병행해야 할 것이다.

참 고 문 헌

[1] 강주성, 김재헌, 박상우, 박춘식, 지성택, 하길찬, 한재우, 현대암호학, 경문사, 2000.

[2] 한국정보통신기술협회, "해쉬함수알고리즘표준(HAS-160)", TTAS.KO-12.0011/R1, TTA, 2000.

[3] T.A. Berson, "Differential cryptanalysis mod 2^{32} with applications to MD5", Eurocrypt'92, pp. 71-80, 1993.

[4] E. Biham and R. Chen, "Near collision of SHA-0", Crypto'04, LNCS 3152, pp. 290-305, 2004.

[5] E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, and W. Jalby, "Collision of SHA-0 and reduced SHA-1", Eurocrypt'05, LNCS 3494, pp. 36-57, 2005.

[6] F. Chabaud and A. Joux, "Differential collisions in SHA-0", Crypto'98. LNCS 1462, pp. 56-71, 1999.

[7] B. den Boer and A. Bosselaers, "An attack on the last two rounds of MD4", Advances in Cryptology-Crypto'91, Lecture Notes in Computer Science, Springer-Verlag, pp. 194-203, 1991.

[8] B. den Boer and A. Bosselaers, "Collision for the compression function of MD5", Advances in Cryptology-Eurocrypt'93, Lecture Notes in Computer Science, Springer-Verlag, pp. 293-304, 1994.

[9] H. Dobbertin, "Cryptanalysis of MD4", Fast Software Encryption, LNCS 1039, pp. 53-69, 1996.

[10] H. Dobbertin, "Cryptanalysis of MD5 compress", Presented at the rump session of Eurocrypt'96.

[11] H. Dobbertin, "Cryptanalysis of MD4", J. Cryptology 11, pp. 253-271, 1998.

[12] P.R. Kasselman and W.T. Penzhorn, "Cryptanalysis of reduced version of HAVAL", Electronics Letters 36, pp. 30-31, 2000.

[13] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997.

[14] NIST, "Secure hash standard", Federal information processing standard, FIPS-180, 1993.

[15] NIST, "Secure hash standard", Federal information processing standard, FIPS-180-1, 1995.

[16] S. Park, S.H. Sung, S. Chee, and J. Lim, "On the Security of Reduced Versions of 3-pass HAVAL", LNCS 2384, pp. 406-419, 2002.

[17] RIPE, "Integrity primitive for secure information system", Final report of RACE integrity primitive evaluation (RIPE-RACE 1040), LNCS 1007, 1995.

[18] R.L. Rivest, "The MD4 message digest algorithm", Advance in Cryptology-Crypto'90, pp. 303-311, 1991.

[19] R.L. Rivest, "The MD5 message digest algorithm", Request for Comments(RFC) 1321, Internet Activities Board, Internet Privacy Task Force, 1992.

[20] B. van Rompay, A. Biryukov, B. Preneel, and J. Vandewalle, "Cryptanalysis of 3-pass HAVAL", Asiacrypt'03, pp. 228-245, 2003.

[21] X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu, "Cryptanalysis of the hash functions MD4 and RIPEMD", Advances in Cryptology-Eurocrypt'05, Lecture Notes in Computer Science 3494, Springer-Verlag, pp. 1-18, 2005.

[22] X. Wang, Y.L. Yin, and H. Yu, "Finding collisions in the full SHA-1", Advances in Cryptology-Crypto'05, Lecture Notes in Computer Science 3621, Springer-Verlag, pp. 17-36, 2005.

[23] X. Wang and H. Yu, "How to break MD5 and other hash functions", Advances in Cryptology-Eurocrypt'05, Lecture Notes in Computer Science 3494, Springer-Verlag, pp. 19-35, 2005.

[24] X. Wang, H. Yu, and Y.L. Yin, "Efficient collision search attacks on SHA-0", Advances in Cryptology-Crypto'05, Lecture Notes in Computer Science 3621, Springer-Verlag, pp. 1-16. 2005.

[25] A. Yun, S.H. Sung, S. Park, D. Chang, S. Hong, and H. Cho, "Finding collision on 45-step HAS-160", LNCS 3935, 2006.

[26] Y. Zheng, J. Pieprzyk, and J. Seberry, "HAVAL-A one-way hashing algorithm with variable length of output", Auscrypt'92, pp. 83-104, 1993.

〈著 者 紹 介〉

성 수 학 (Soo Hak Sung)

1982년: 경북대학교 수학과 졸업 (이학사)

1985년: KAIST 응용수학과 졸업 (이학석사)

1988년: KAIST 응용수학과 졸업 (이학박사)

1988년~1991년: 한국전자통신연구원 선임연구원

1991년~현재: 배재대학교 전산정보수학과 교수

관심분야: 암호이론