通信情報保護學會論文誌
第 7 卷 第 4 號. 1997. 12

MDx-계열 해쉬 함수에 기반한 새로운 해쉬 함수

신 상 욱*, 류 대 현**, 이 상 진**, 이 경 현*

A new hash function based on MDx-family hash functions

Sang Uk Shin, Dae Hyun Ryu, Sang Jin Lee, Kyung Hyune Rhee

요 약

암호적으로 안전한 해쉬 함수는 디지털 서명, 메시지 인증, 키 유도와 같은 분야에서 중요한 암호 도구이다. 현재까지 제안된 소프트웨어로 고속 수행이 가능한 해쉬 함수들의 대부분은 Rivest가 제안한 MD4의 설계 원리에 기반을 두고 있다. 이들 MD 계열 해쉬 함수 중에서 현재 안전하다고 알려진 전용 해쉬 함수는 SHA-1, RIPEMD-160, HAVAL 등이다. 본 논문에서는 이들 세 가지 해쉬 함수들의 장점에 기반하여 이들 함수들이 가지는 안전성을 최대한 유지하면서 보다 효율적인 새로운 해쉬 함수를 제안한다. 제안된 해쉬 함수는 임의 길이 메시지를 512 비트 단위로 처리하여 160 비트의 출력을 가진다. 제안된 해쉬 함수는 입력 데이터에 의존한 순환이동(data-dependent rotation)의 특징을 가짐으로써 기존에 알려진 공격에 강인함을 보장하며 두 개의 충돌 메시지 발견을 위해서는 생일공격에 의해 2^{80}연산이 요구되어진다고 추측된다. 제안된 해쉬 함수의 성능은 수행 속도면에서 RIPEMD-160보다 약 30%, SHA-1보다는 약 7% 효율적이다.

Abstract

Cryptographic hash functions are an important tool for applications such as digital signature, message authentication code and key derivation. During the last years, several fast software hash functions have been proposed; most of them are based on the design principles of R. Rivest's MD4. Up to now SHA-1, RIPEMD-160, and HAVAL are known as secure dedicated hash functions in MD-family hash functions. In this paper we propose a new hash function based on advantages of these three hash functions, which keeps the maximum security of these three hash functions and is more efficient. The proposed hash function processes arbitrary finite message by 512-bit block and outputs 160 bits digest.

* 부경대학교 전자계산학과

* * 한국전자통신연구원

† 이 논문은 1997년 한국전자통신연구원 위탁과제 연구비에 의해 수행되었음.

The feature of the proposed hash function is *data-dependent rotation*. The proposed hash function guarantee the strength against existing known attacks and it is conjectured that finding two collision messages requires the order of 2^{80} operations by the Birthday attack. In aspect of processing speed, the proposed hash function is about 30% faster than RIPEMD-160, and about 7% faster than SHA-1.

Key words : cryptographic hash function, dedicated hash function

1. 서 론

정보화 시대를 맞이하여 정보 보호 문제의 필요성이 사회적으로 크게 대두되고 있으며 특별히 쌍방의 정보 교환시에 메시지의 도청 및 수정, 삽입, 그리고 송수신자의 위장 등의 문제가 발생할 수 있다. 이와 같은 환경에서 인증과 무결성은 기밀성과 더불어 필수적인 요구 사항이다. 중요 정보의 무결성 확인과 메시지 인증 코드(Message Authentication Code : MAC)의 구성, 디지털 서명(digital signature)의 효율성 증대 등의 목적으로 해쉬 함수가 사용된다. 해쉬 함수는 임의 길이의 비트스트링을 입력으로 받아 고정된 짧은 길이(주로 128, 160 비트)의 비트스트링을 출력하는 함수이다. 많은 양의 정보에 대한 인증을 제공하는 효율적인 방법으로는 그 정보로부터 계산된 짧은 해쉬 결과에 대해 인증을 제공하는 것이다. 디지털 서명 기법에 해쉬 함수를 사용하므로서 더 짧은 서명을 얻고, 계산이 더 용이하게 되어 디지털 서명의 효율성을 증대시킬 수 있다.

현재까지 많은 해쉬 함수가 제안되어져왔지만, 오늘날 널리 사용되는 대부분의 해쉬 함수는 기본적으로 Merkle[9]과 Damgard[3]의 이론에 기반을 둔 반복적인 형태이다. 1990년 Rivest가 MD4[15][16] 해쉬 함수를 제안한 이후 대부분의 해쉬 함수는 MD4의 설계 원리에 기반을 두고 개발되었으며 MD4는 32-비트 기계에서 빠르게 동작하도록 설계되었다. 그러나 MD4에 대해 Merkle의 처음 1, 2 라운드에 대한 공격과 Bosselaers[2]에 의한 마지막 2 라운드에 대한 공격에 의해 MD4의 취약점이 발견됨에 따라 1991년에 다시 Rivest에 의해 MD4의 취약점을 개선시킨 MD5[17]가 제안되었다. 또한 유럽의 RIPE 컨소시움에서 MD4와 MD5에 대한 독립적인 평가에 기초하여 MD4의 강화된 버전으로 RIPEMD[14]를 1995년에 제안했지만 Dobbertin[4]에 의해 RIPEMD의 축소된 버전에 대한 공격이 1996년에 발견됨에 따라 이를 개선한 RIPEMD-128/160[6]이 Dobbertin, Bosselaers, Preneel에 의해 제안되었다. 이 Dobbertin의 공격 방법은 RIPEMD 외에 MD4와 MD5에 대한 공격으로 확장가능하여 이들 해쉬 함수에 심각한 위험이 있음을 보였다[5][7]. 1993년 미국의 NIST(National Institute of Standard and Technology)에서 FIPS PUB 180으로 SHA(Secure Hash Algorithm)[10]을 공개하였고 1995년 자체적으로 발견된 약점을 보완하여 SHA-1[11]을 발표하였으나, 구체적인 설계 기준이나 공격 사례는 공개되지 않았으며 현재 미국 연방 정부의 표준 해쉬 함수로 공인되어 있다. 그리고 1992년 Zheng, Pieprzyk, Seberry에 의해 가변 길이의 해쉬 값을 요구하는 여러 응용에 적합한 해쉬 함수로 HAVAL[20]이 설계되었고 국내에서는 이필중 교수팀[21] 등이 한국형 디지털 서명 방식에 사용될 수 있는 해쉬 함수로 PMD-V와 PMD-N을 제안하였다.

본 논문에서는 MD 계열 해쉬 함수 - MD4, MD5, RIPEMD, RIPEMD-128/160, SHA-1,

HAVAL, PMD-N, PMD-V - 의 구체적인 설계 원리에 기초하여 지금까지 알려진 공격에 대해 안전한 새로운 해쉬 함수를 제안한다. 제안된 알고리즘은 512 비트 블록 단위로 처리하며 4 라운드로 구성되며 160 비트의 연쇄 변수와 출력을 가진다. 제안된 해쉬 함수는 SHA-1에서의 메시지 확장과 HAVAL에서의 암호학적으로 강한 성질을 만족하는 부울 함수를 적용한다. 그리고 제안된 알고리즘에서 가장 중요한 초점은 데이터 의존 순환이동(data-dependent rotation)이다[18]. 단계 연산에서 순환이동을 기존의 MD 계열 해쉬 함수에서의 고정된 값이 아닌 가변적인 입력 데이터에 의존하는 순환이동을 사용하므로서 해쉬 결과가 좀더 강하게 입력 메시지에 의존하도록 한다. 제안된 알고리즘은 두 개의 충돌 메시지를 발견하는데 2^{80}연산이 요구되는 것으로 추측된다.

본 논문의 구성으로 II절에서는 해쉬 함수의 정의와 일반적인 모델을 기술하고, III절에서 제안된 알고리즘을 기술한다. IV절에서는 알고리즘의 설계 기준과 기존 MD 계열 해쉬와의 성능 비교를 제시하고 V절에서는 결론과 추후 연구 방향을 기술한다.

II. 해쉬 함수의 정의와 일반적인 모델

해쉬 함수(hash function), 좀더 정확하게 암호학적 해쉬 함수(cryptographic hash function)는 임의의 유한 길이 비트스트링을 고정된 길이의 스트링으로 사상시키는 함수이다. 이 출력은 흔히 해쉬 값(hash value), 메시지 다이제스트(message digest) 또는 fingerprint 등으로 불린다. 함수 h와 입력 x가 주어지면, $h(x)$를 계산하는 것은 쉬워야 한다. 일방향 해쉬 함수는 다음 성질을 만족해야 한다[13].

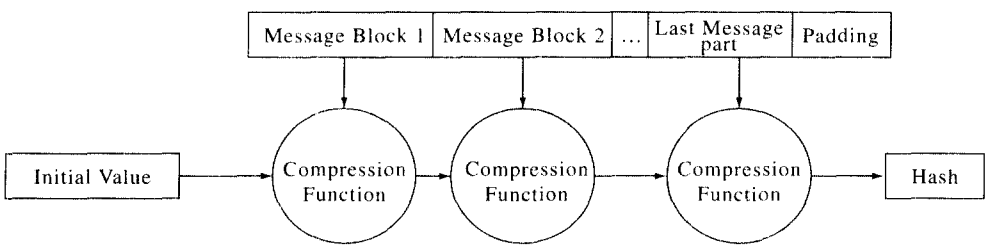


그림 1. 반복적인 해쉬 함수에서 압축 함수의 사용

Fig. 1. The use of a compression function in an iterative hash function

- ***preimage resistance*** : 어떤 미리 명시된 출력으로 해쉬하는 어떤 입력을 발견하는 것이 계산상 수행불가능하다. 즉, 해쉬 값 y가 주어졌을 때, $h(x)=y$를 만족하는 입력 x를 발견하는 것이 계산상 수행불가능하다.
- ***second preimage resistance*** : 어떤 명시된 입력과 같은 출력을 가지는 어떤 두 번째 입력을 발견하는 것이 계산상 수행불가능하다. 즉, 입력 x와 출력 $h(x)$가 주어졌을 때, $h(x)=h(x')$을 만족하는 입력 $x'\neq x$를 발견하는 것이 계산상 수행불가능하다.

암호학적으로 유용한 해쉬 함수는 다음 성질을 추가로 만족해야 한다.

- ***collision resistance*** : 충돌(같은 결과로 해쉬하는 두 개의 다른 입력)을 발견하는 것이 계산상 수행불가능하다. 즉, $h(x)=h(x')$을 만족하

는 임의의 서로 다른 두 입력 쌍 x와 x'을 발견하는 것이 수행불가능하다.

거의 대부분의 해쉬 함수의 처리 과정은 입력을 연속적인 고정된 블록들로 나누어 처리함으로서 임의 길이 입력을 해쉬하는 반복적인 처리 과정이다. 먼저 입력 X는 블록 길이의 배수가 되도록 padding되고 t 개의 블록 X_i에서 X_t로 나누어진다. 해쉬 함수 h는 다음처럼 기술된다.

$$H_0=IV$$
$$H_i=f(H_{i-1}, X_i),\ 1\le i\le t$$
$$h(X)=H_t$$

여기서 f는 압축 함수(compress function)이고, H_i는 단계 i-1과 단계 i 사이의 연쇄 변수(chaining variable)이고, IV는 초기값(Initial Value)이다. 압축 함수를 사용한 반복적인 해쉬 함수의 블록도가 그림 1에 주어졌다. 해쉬 값의 계산은 연쇄 변수에 의존한다. 해쉬 계산을 시작할 때, 이 연쇄 변수는 알고리즘의 일부로 명시된 고정된 초기 값을 가진다. 압축 함수는 해쉬되어질 메시지 블록을 입력으로 받아 이 연쇄 변수의 값을 갱신한다. 이 과정이 모든 메시지 블록에 대해 순환적으로 반복되고, 연쇄 변수의 마지막 값이 그 메시지에 대한 해쉬 값으로 출력된다.

해쉬 함수는 크게 블록 암호 알고리즘을 이용한 해쉬 함수와 전용 해쉬 함수 (dedicated hash functions)로 분류할 수 있다. DES나 IDEA와 같은 블록 암호를 이용하는 경우는 기존에 구현되어있는 블록 암호를 이용할 수 있다는 장점이 있지만 대부분의 블록 암호의 처리속도가 느리고, 또한 이를 기본으로 이용하는 해쉬 함수의 경우 속도가 훨씬 더 저하되므로, 현재 대부분의 응용에서는 주로 전용 해쉬 함수를 사용한다. 전용 해쉬 함수의 대표적인 예로 MD 계열의 해쉬 함수(MD2, MD4, MD5, RIPEMD, RIPEMD-128/160, SHA-1, HAVAL 등)가 있다. MD4는 Dobbertin[15]의 공격에 의해 완전히 해독되었고 MD5 역시 심각한 취약점이 발견되었다[17]. 현재까지 안전하다고 생각되는 함수로 SHA-1, RIPEMD-160, HAVAL 등이 있다. 1995년에 발표된 SHA-1의 가장 큰 특징은 입력 메시지로부터 새로운 메시지 변수를 생성하는 메시지 확장이며 이는 기존의 메시지 적용의 단순성에 기인한 공격에 대해 강한 저항력을 가진다. RIPEMD-160[16]은 두 개의 병렬 라인으로 처리하여 안전성을 향상시킨다는 특징을 가지고 있으며 HAVAL[20]은 가변적인 출력과 패스를 가진다는 것과 함께 기존 MD 계열 해쉬 함수와는 다른 암호학적으로 강한 7변수 부울 함수를 사용하여 안전성을 증가시키려는 특징을 가지고 있다.

본 논문에서는 위의 세 가지 알고리즘의 장점에 기초하여 안전하고 효율적인 새로운 해쉬 알고리즘을 제안한다. 메시지 적용의 단순성을 제거하기 위해 병렬 라인으로 구성할 수 있으나 이는 효율성에 심각한 영향을 미치므로 SHA-1에서의 메시지 확장을 적용하고 HAVAL에서처럼 암호학적으로 강한 부울 함수를 적용하여 기존 알고리즘이 가지는 안전성을 최대한 유지하면서 성능면에서 보다 효율적인 새로운 해쉬 알고리즘을 제안한다.

III. 새로운 해쉬 알고리즘 기술

이 절에서는 새로운 해쉬 함수에 관하여 세부적으로 기술한다. 앞으로 사용할 용어와 기호는 다음과 같다.

◇ 워드(word) : 32 비트 스트링
◇ 블록(block) : 해쉬 함수에서 압축 함수의 입력 단위로 512 비트를 사용한다.
◇ + : 워드간의 법 2^{32}에서의 덧셈 연산

◇ $X^{<<s}$: X를 s 비트만큼 왼쪽으로 순환이동 (circular shift) 연산
◇ $X \wedge Y$: X와 Y의 비트별 논리 AND 연산
◇ $X \vee Y$: X와 Y의 비트별 논리 OR 연산
◇ $X \oplus Y$: X와 Y의 비트별 논리 XOR 연산

모든 메시지는 512 비트(16 워드) 단위로 처리되고 해쉬 결과와 연쇄 변수(chaining variable)는 160 비트이며 라운드 수는 4 라운드를 이용한다. 16개 입력 워드로부터 추가로 8개의 메시지 변수를 생성하여 총 24개 워드를 사용하여 각 라운드에서 24 단계 연산을 수행한다.

(1) 입력 블록 길이 및 덧붙이기(padding)

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

(2) 초기값(Initial Value) IV

메시지 처리에 사용될 5개의 연쇄 변수 (A, B, C, D, E)의 초기값은 다음과 같다.

A	B	C	D	E
0x67452301	0xefcdab89	0x98badcfe	0x10325476	0xc3d2e1f0

(3) 상수(constant)

각 라운드에서 사용될 상수 K는 다음과 같다.

K_1	K_2	K_3	K_4
0	0x5a827999 $\lfloor 2^{30} \cdot \sqrt{2} \rfloor$	0x6ed9eba1 $\lfloor 2^{30} \cdot \sqrt{3} \rfloor$	0x8f1bbcdc $\lfloor 2^{30} \cdot \sqrt{5} \rfloor$

(4) 메시지 변수 확장

16개 입력 메시지 $X_i (i<16)$에서 8개 메시지 변수를 추가로 생성한다.

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

(5) 메시지 워드의 적용 순서

다음의 순열 ρ로부터 각 라운드에서의 메시지 워드 적용 순서를 정의한다.

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

1 라운드	2 라운드	3 라운드	4 라운드
i	ρ	ρ^2	ρ^3

(6) 단계 연산

단계 연산은 다음과 같이 정의된다. 각 단계 연산이 수행된 후 연쇄 변수는 오른쪽으로 회전 이동한 후 다음 단계 연산이 적용된다.

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

$B=B^{<<10}$

(7) 부울 함수(boolean function)

각 라운드의 단계 연산에서 사용되는 부울 함수는 다음과 같다.

$f_0(x_1, x_2, x_3, x_4, x_5)=x_1x_2\oplus x_3x_4\oplus x_2x_3x_4\oplus x_5$

$f_1(x_1, x_2, x_3, x_4, x_5)=x_2x_3\oplus x_4x_5\oplus x_1$

$f_2(x_1, x_2, x_3, x_4, x_5)=x_1x_3\oplus x_2x_5\oplus x_3x_5\oplus x_4$

1 라운드	2 라운드	3 라운드	4 라운드
f_0	f_1	f_2	f_1

(8) 순환 이동

각 라운드의 단계 연산에서 사용되는 순환 이동 값 s는 입력 메시지에 의존(입력 메시지의 하위 5 비트에 의존)하여 다음처럼 정의된다.

$s=X_i \bmod 32$

여기서 X_i는 다음과 같다(ρ는 메시지 적용 순서에서의 치환 ρ이다).

1 라운드	2 라운드	3 라운드	4 라운드
ρ^3	ρ^2	ρ	i

Ⅳ. 새로운 해쉬 함수의 설계 기준과 성능 비교

제안된 해쉬 함수는 기존의 MD 계열 해쉬 함수의 설계 원리를 근간으로 한다. 기존의 해쉬 함수에서 이미 얻어진 신뢰를 최대화하기 위해 가능한한 적은 변화를 취했다. 압축 함수의 전체 구조나 덧붙이기(padding), 초기값 IV, 상수 값 등은 대부분의 기존 해쉬 함수와 유사하다. 제안된 해쉬 알고리즘의 특징은 다음과 같다.

(1) 출력 길이

제안된 알고리즘은 160 비트의 출력 길이를 가진다. HAVAL과 PMD-V와 같이 가변 출력을 고려해볼 수도 있으나 그로 인한 장점이 아직은 의문시되고, 128 비트 출력의 경우는 전탐색 공격(brute force attack)에 대해 충분한 안전성을 제공하지 못하므로 고정된 160 비트의 출력을 가지도록 설계했다[12].

(2) 메시지 워드의 확장 및 적용 순서

기존의 대부분의 공격이 메시지 워드 적용의 단순함을 이용하므로 하나의 메시지 워드가 가능한한 많은 단계에 영향을 주도록 설계하였다. 입력된 16개 메시지 워드로부터 8개의 메시지를 추가로 생성하여 적용한다. SHA-1에서 입력된 16개 워드를 사용하여 64개의 메시지를 추가로 생성하여 처리하는데 너무 많은 수의 메시지 확장은 효율성에 영향을 미치므로 기본적으로 SHA-1에서의 메시지 확장 형태를 이용하되 16개의 입력 워드가 유사한 빈도로 적용되면서 빠르게 구현되도록 설계하였다.

메시지 워드 적용 순서는 RIPEMD-160의 설계 원리를 참조하였다[6]. 추가로 생성된 워드들이 충분히 분산되도록 하고 동일한 워드들이 각 라운드마다 인접하지 않고 각 라운드의 각 단계에서 동일한 워드가 사용되지 않도록 하였다.

(3) 순환이동

제안된 알고리즘에서 가장 중요한 요소로 데이터 의존 순환이동을 사용한다. 각 라운드의 단계 연산에서 사용되는 순환이동 값은 기존의 MD 계열의 고정된 순환이동과는 다른 가변적인 값을 가지게 하므로서 안전성을 향상시킨다. 순환이동 값은 입력 메시지 워드에 의존(입력 메시지 워드의 하위 5 비트에 의존)하는 값을 가진다. 메시지 의존 순환이동을 사용하므로서 출력이 좀더 입력 메시지에 의존하게 되어 안전성이 증가된다. 그리고 단계 연산에서 사용되는 메시지 워드(부울 함수의 결과에 더해지는 메시지 워드)와 다른 메시지 워드에 의존하도록 하므로서 가능한 공격에 대해 좀더 안전하도록 하였다.

(4) 부울 함수

부울 함수는 HAVAL의 방법에 기초한다. 기존 MD 계열의 3 변수 부울 함수 대신에 HAVAL에서와 유사한 암호학적으로 강한 성질들을 가지는 5 변수 부울 함수를 사용한다[20][21]. 부울 함수 f_0는 4 변수 bent 함수를 이용하여 만들어지며 0-1 balanced를 만족하고 높은 비선형성(high nonlinearity)을 가지며 SAC(Strict Avalanche Criterion)을 만족한다[19]. 부울 함수 f_2 역시 0-1 balanced를 만족하며 높은 비선형성을 가지며 SAC을 만족하도록 설계되었다[19][20]. 부울 함수 f_1은 4 변수 부울 함수에 한 변수를 XOR하여 얻어지며, XOR된 한 변수를 제외하면 0-1 balanced와 높은 비선형성(high nonlinearity) 그리고 SAC을 만족한다[21]. 각 라운드에서 부울 함수의 적용 순서는 f_0, f_1, f_2, f_1으로 계산량이 가장 적은 f_1을 중복 사용하여 효율성을 고려하였다.

(5) 문자열과 정수간의 변환(Endianness)

제안된 알고리즘은 32 비트 프로세서의 little-endian 구조 컴퓨터에서 빠르게 동작하도록 설계되었다. 따라서 big-endian 구조에서는 추가적으로 메시지 워드들의 바이트 순서를 역순으로 바꾸어주어야 한다. 이는 SHA-1을 제외한 대부분의 MD 계열 해쉬 함수들이 32 비트 little-endian 구조로 설계되었기 때문이다.

제안된 알고리즘은 기존에 알려진 den Boer와 Bosselaers의 공격[2]과 Dobbertin의 공격[4][5][17]에 대해 안전하다고 여겨지고 데이터 의존 순환이동으로 인해 각 라운드에서 비트들이 임의 위치로 순환이동하기 때문에 차분 공격(differential cryptanalysis)[1]과 선형 해독(linear cryptanalysis)[8]에 안전할 것이다[18]. 그리고 충돌 발견에 대해 최상의 공격은 생일

공격(birthday attack)을 사용하는 것으로 추측된다. 그러한 공격에 대해 공격자는 두 개의 2^{80}개의 다른 메시지 집합을 준비하여 해쉬값을 계산해야 한다.

제안된 알고리즘의 성능을 MD5, SHA-1, RIPEMD-160, HAVAL(5 패스, 160 비트)과 비교한 결과를 표 1에 보인다. 각 알고리즘은 C 언어로 구현되었고 펜티움 100MHz 에서 실험하였다. 입력 파일은 10Mbytes로 8Kbytes의 메시지 버퍼를 사용하였다. 사용된 프로그램은 최적화된 것이 아니며 실행 속도를 측정하는 clock() 함수에 의해 약간의 편차가 존재한다. 속도를 비교해보면 제안된 알고리즘은 RIPEMD-160보다 약 30% 빠르며 SHA-1보다 약 7% 빠르다. MD5의 성능이 가장 좋은 것으로 나타나지만 이는 MD5의 해쉬 결과가 128 비트라는 것을 고려하고 또한 현재 SHA-1과 RIPEMD-160이 안전하다고 여겨지므로 이들에 대한 대안으로 제안된 알고리즘이 사용될 수 있을 것이다. 한편 표 2에 기 개발된 방식과 제안 방식 사이의 알고리즘에 대한 비교를 나타내었다.

표 1. MD 계열 해쉬 함수와 제안된 해쉬 함수의 성능

Table 1. The performance of MD-family hash functions and the proposed hash function

성능 / 해쉬함수	초당 처리되는 bits 수 (Mbits/second)	상대적인 성능
MD5	6.12	1.00
SHA-1	2.38	0.39
RIPEMD-160	1.77	0.29
HAVAL(5 PASS,160 bits)	3.18	0.52
제안 방식	2.55	0.42

표 2. 기 개발된 방식과 제안 방식의 비교

Table 2. The comparison of existing developed methods and the proposed method

	MD5	SHA-1	RIPEMD-160	HAVAL	제안방식
digest길이	128비트	160비트	160비트	128~256비트	160비트
처리블록단위	512비트	512비트	512비트	1024비트	512비트
단계 수	64	80	160	96~160	96
부울함수개수	4	3	5	3~5	3
사용되는 상수	64	4	8	64~128	3

V. 결 론

기존 MD 계열 해쉬의 설계 원리와 공격 사례를 분석하여 새로운 해쉬 함수를 제안하였다. 제안된 알고리즘은 임의의 길이 메시지를 512 비트 단위로 처리하여 160 비트의 결과를 출력한다. 전체 구조는 4 라운드로 구성되며 각 라운드는 24 단계 연산을 수행한다. 또한

매 라운드마다 16개 입력 워드를 24개 워드로 확장하여 처리하며 각 단계 연산에서는 암호학적으로 강한 성질을 가지는 부울 함수를 사용한다. 기존의 해쉬 함수와 구별되는 가장 큰 특징 중의 하나는 단계 연산의 순환이동이 고정된 값을 사용하는 것이 아니라 입력 메시지에 의존하는 가변적인 값을 가진다는 것이다. 이로 인해 출력이 기존의 알고리즘보다 좀더 입력 메시지에 강하게 의존하게 되어 안전성이 증가된다. 제안된 알고리즘은 수행 속도면에서 RIPEMD-160보다 약 30% 빠르며 SHA-1보다 약간 빠름을 실험 결과를 통해 알 수 있었다. 현재 SHA-1과 RIPEMD-160이 안전하다고 여겨지므로 이들에 대한 대안으로 제안된 알고리즘이 사용될 수 있을 것이다.

제안된 알고리즘에서 충돌 메시지 쌍을 발견하기 위해 2^{80}연산이 요구된다고 추측된다. 제안된 해쉬 함수는 기존의 알려진 공격에 대해 안전하며 성능면에서 효율적이라고 여겨지지만 향후 안전성과 구현상의 최적화 등을 고려한 심도 깊은 분석을 통해 제안된 알고리즘을 개선시켜 나갈 것이다. 또한 제안된 해쉬 함수를 사용한 효율적인 MAC의 구성에 대해서도 추후 연구할 것이다.

Appendix. 제안된 알고리즘의 pseudo-code

제안된 해쉬 함수는 32-비트 워드로 연산하는 반복적인 해쉬 함수이다. 라운드 함수는 5개 워드의 연쇄 변수와 24개 메시지 워드를 입력으로 받아 이것을 새로운 연쇄 변수로 사상시킨다. 덧붙이기(padding)은 MD4와 동일하다. 먼저 모든 상수와 부울 함수를 정의한다.

부울 함수

$f_j(x_1, x_2, x_3, x_4, x_5)=x_1x_2\oplus x_3x_4\oplus x_2x_3x_4\oplus x_5$

$(0\le j\le 23)$

$f_j(x_1, x_2, x_3, x_4, x_5)=x_2x_3\oplus x_4x_5\oplus x_1$

$(24\le j\le 47,\ 72\le j\le 95)$

$f_2(x_1, x_2, x_3, x_4, x_5)=x_1x_3\oplus x_2x_5\oplus x_3x_5\oplus x_4$

$(48\le j\le 71)$

상수

K_j=0x00000000$(0\le j\le 23)$

K_j=0x5a827999$(24\le j\le 47)(\lfloor 2^{30}\cdot 2\rfloor)$

K_j=0x6ed9eba1$(48\le j\le 71)(\lfloor 2^{30}\cdot 3\rfloor)$

K_j=0x8f1bbcdc$(72\le j\le 95)(\lfloor 2^{30}\cdot \mathrm{sqrt}\ 5\rfloor)$

메시지 워드의 선택

$r(j)=j(0\le j\le 23)$

$r(24\ldots47)$=4, 21, 17, 1, 23, 18, 12, 10, 5, 16, 8, 0, 20, 3, 22, 6, 11, 19, 15, 2, 7, 14, 9, 13

$r(48\ldots71)$=23, 14, 19, 21, 13, 15, 20, 8, 18, 11, 5, 4, 7, 1, 9, 12, 0, 2, 6, 17, 10, 22, 16, 3

$r(72\ldots95)$=3, 9, 17, 22, 1, 12, 10, 18, 6, 4, 15, 13, 8, 14, 11, 7, 23, 19, 20, 2, 5, 16, 0, 21

순환이동시 적용되는 메시지 워드 선택

$s(0\ldots23)$= 3, 9, 17, 22, 1, 12, 10, 18, 6, 4, 15, 13, 8, 14, 11, 7, 23, 19, 20, 2, 5, 16, 0, 21

$s(24\ldots47)$= 23, 14, 19, 21, 13, 15, 20, 8, 18, 11, 5, 4, 7, 1, 9, 12, 0, 2, 6, 17, 10, 22, 16, 3

$s(48\ldots71)$=4, 21, 17, 1, 23, 18, 12, 10, 5, 16, 8, 0, 20, 3, 22, 6, 11, 19, 15, 2, 7, 14, 9, 13

$s(72\ldots95)$=0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23

초기값 IV

h_0=0x67452301, h_1=0xefcdab89 , h_2=0x98badcfe, h_3=0x10325476, h_4=0xc3d2e1f0

padding 후의 메시지는 t 개의 16-워드 블록, $X_i[j](0 \le i \le t-1,\ 0 \le j \le 15)$으로 구성된다.

1. Pseudo-code

for i=0 **to** t-1 {
  $A=h_0$; $B=h_1$; $C=h_2$; $D=h_3$; $E=h_4$;
  for j=0 **to** 7
    $X_i[16+i]=(X_i[0+i] \oplus X_i[2+i] \oplus X_i[7+i] \oplus X_i[12+i])^{<<1}$;
  for j=0 **to** 95 {
    $T=(f(A, B, C, D, E)+X_i[j]+K[j])^{<<(X_i[s(j)]\&31)}$;
    $A=E$; $E=D$; $D=C$; $C=B^{<<10}$; $B=T$;
  }
  $h_0+=C$; $h_1+=D$; $h_2+=E$; $h_3+=A$; $h_4+=B$;
}

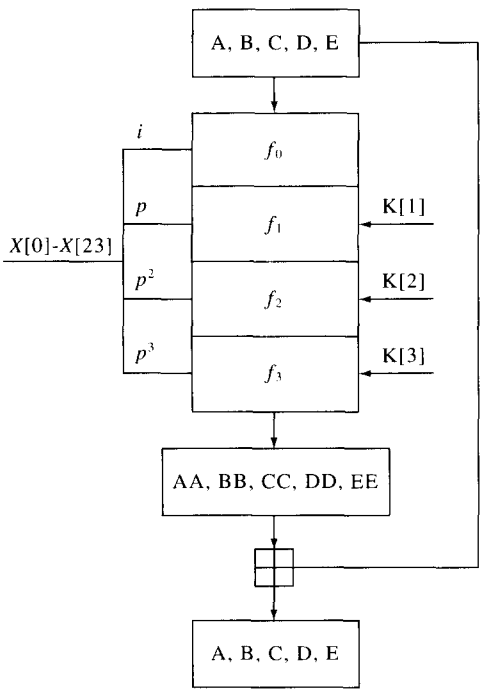


그림 2. 제안된 해쉬 함수에서 압축 함수의 구조

Fig. 2. Outline of the compression function of the proposed hash function

2. Test Values

* message: "" (empty string)
  hashcode: 74b3fef703000b9d3484c0f660b7c34c2eac74cd
* message: "a"
  hashcode: 4b1760ef9e73f6e4deb720545c22ef51d823719e
* message: "abc"
  hashcode: b71e348472a8d534bdc0eb2ab00f79d6e3dd9b3c
* message: "message digest"
  hashcode: b6419615a25ebc29c670867b027b0d8759c8990c
* message: "abcdefghijklmnopqrstuvwxyz"
  hashcode: 00af1ba0c308f16cb4f55c503eb78fcc2fb809ea
* message: "abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq"
  hashcode: 832e40051ef2f6e88b18b10d6a9a0ff3913b8128
* message: "A...Za...z0...9"
  hashcode: ed4f896660c6b0aaba0b18053794eac71bc6bf4e
* message: 8 times "1234567890"
  hashcode: 003cb245f56b5a2bdf8706f3970c0dab814ab664
* message: 1 million times "a"
  hashcode: 8b0f5260dc288ed64308417a3e083e421f657a0b

참 고 문 헌

[1] E. Biham, A. Shamir, Differential

cryptanalysis of DES-like cryptosystems, Advances in Cryptology- Crypto'90, Lecture Notes in Computer Science, vol.537, Springer-Verlag, 1991, pp. 2-21

[2] B. den Boer, A. Bosselaers, An attack on the last two rounds of MD4, Advances in Cryptology-Crypto'91, Lecture Notes in Computer Science, vol.576, Springer-Verlag, 1992, pp. 194-203

[3] I.B. Damg rd, A design principle for hash functions, Advances in Cryptology-Crypto'89, Lecture Notes in Computer Science, vol.435, Springer-Verlag, 1990, pp. 416-427

[4] H. Dobbertin, RIPEMD with two-round compress function is not collision-free, Journal of Cryptology, vol.10, no.1, 1997, pp. 51-69

[5] H. Dobbertin, Cryptanalysis of MD4, Fast Software Encryption-Cambridge Workshop, Lecture Notes in Computer Science, vol.1039, Springer-Verlag, 1996, pp. 53-69

[6] H. Dobbertin, A. Bosselaers, B. Preneel, RIPEMD-160: A strengthened version of RIPEMD, Fast Software Encryption-Cambridge Workshop, Lecture Notes in Computer Science, vol.1039, Springer-Verlag, 1996, pp. 71-82

[7] H. Dobbertin, The status of MD5 after recent attack, CryptoBytes, 2(2), Sep. 1996, pp. 1-6

[8] M. Matsui, The first experimental cryptanalysis of the Data Encryption Standard, Advances in Cryptology-Crypto' 94, Lecture Notes in Computer Science, vol.839, Springer-Verlag, 1994, pp. 1-11

[9] R. Merkle, One way hash functions and DES, Advances in Cryptology-Crypto' 89, Lecture Notes in Computer Science, vol.435, Springer-Verlag, 1990, pp. 428-446

[10] NIST, Secure hash standard, FIPS 180, US Department of Commerce, Washington D.C., 1993

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

[12] P.C. van Oorshot, M.J. Wiener, Parallel collision search with applications to hash functions and discrete logarithms, Proc. of the 2nd ACM Conference on Computer and Communications Security, ACM, 1994, pp. 210-218

[13] B. Preneel, Analysis and design of cryptographic hash functions, Doctoral Dissertation, Katholieke Universiteit Leuven, 1993

[14] RIPE Consortium : RIPE Integrity Primitives - Final report of RACE Integrity Primitives Evaluation(R1040), Lecture Notes in Computer Science, vol.1007, Springer-Verlag, 1995

[15] R. Rivest, The MD4 message-digest algorithm, Advances in Cryptology-Crypto' 90, Lecture Notes in Computer Science, vol.537, Springer- Verlag, 1991, pp. 303-311

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

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

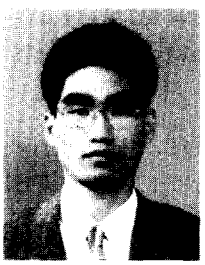
[18] R. Rivest, The RC5 Encryption Algorithm, CryptoBytes, 1(1);9-11, 1995 (Revised 3, 20, 1997, http://theory.lcs.mit.edu/~rivest/rc5rev.ps)

[19] J. Seberry, X. M. Zhang, Highly nonlinear 0-1 balanced boolean functions satisfying strict avalanche criterion, Advances in Cryptology- Auscrypt'92, Lecture Notes in Computer Science, 718, Springer-Verlag, 1993, pp. 145-154

[20] 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, 1993, pp. 83-104

[21] 임채훈, 박난정, 이은정, 이필중, 출력길이 선택이 가능한 새로운 해쉬 함수의 제안, preprint, 1997

□ 著者紹介



신 상 욱

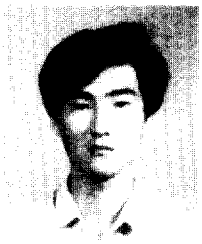
1995년 부산수산대학교(현 부경대학교) 전자계산학과 졸업(이학사)
1997년 부경대학교 전자계산학과 대학원 졸업(이학석사)
1997년 ~ 현재 부경대학교 전자계산학과 박사과정

※ 주관심분야 : 정보보호론, 컴퓨터 보안, 암호학, 네트워크 이론(성능분석), 대기체계론

류 대 현

1983년 2월 부산대학교 전기기계공학과 졸업(공학사)
1985년 2월 부산대학교 전자공학과(공학석사)
1987년 2월 부산대학교 전자공학과(공학박사)
1987년 3월 ~ 현재 전자통신연구원 재직중

※ 주관심분야 : 신경회로망, 영상처리, 정보보호 및 암호 등임



이 상 진

1987년 2월 고려대학교 이과대학 수학과(이학사)
1989년 2월 고려대학교 대학원 수학과(이학석사)
1994년 8월 고려대학교 대학원 수학과(이학박사)
1989년 ~ 현재 한국전자통신연구원 선임연구원

※ 주관심 분야 : 응용대수학 및 정수론, 암호론

이 경 현

1982년 경북대학교 사범대학 수학교육과 졸업(이학사)
1985년 한국과학기술원 응용수학과 졸업(이학석사)
1992년 한국과학기술원 수학과 졸업(이학박사)
1985년 2월 ~ 1991년 2월 한국 전자 통신 연구소 연구원
1991년 3월 ~ 1993년 2월 한국 전자 통신 연구소 선임 연구원
1993년 3월 ~ 현재 부경대학교(구 부산수산대학교) 전임강사, 조교수
1995년 7월 ~ 1996년 7월 Univ. of Adelaide, 응용수학과, Australia 방문교수

※ 관심분야 : 정보보호론, 컴퓨터 보안, 암호학, 네트워크 이론(성능분석), 광대역 통신망, 대기체계론