MD-계열에 기반한 새로운 해쉬 함수와 MAC에의 응용*

신 상 욱†, 류 대 현‡, 이 상 진‡, 이 경 현†,
† 부경대학교 전자계산학과
‡ 한국전자통신연구원

A new hash function based on MD-family and its application to the MAC

Sang Uk Shin†, Dae Hyun Ryu‡, Sang Jin Lee‡, Kyung Hyune Rhee†
† Dept. of Computer Science, Pukyong National University
‡ Electronics and Telecommunications Research Institute

요 약

암호학적으로 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 가지며, 입력 데이터에 의존한 순환이동(data-dependent rotation)의 특징을 가짐으로써 기존에 알려진 공격에 강인함을 보장한다. 또한 제안된 해쉬 함수를 이용한 메시지 인증 코드(Message Authentication Code:MAC) 구성 알고리즘을 제안한다. MAC은 두 번째 입력, 즉 비밀키를 가진 해쉬 함수로 keyed 해쉬 함수라고 하며, 메시지 출처 인증과 무결성 확인을 제공한다. 제안된 MAC은 최대 160-비트의 키를 사용하며 해쉬 결과보다 같거나 적은 MAC 결과를 가지며, 성능 면에서 사용되는 해쉬 함수에 비해 약 10% 정도의 저하를 초래한다.

1. 서론

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 메

* 이 연구는 1997년 한국전자통신연구원 위탁과제 연구비 지원에 의해 수행되었음.

시지 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[12]과 Damgård[5]의 이론에 기반을 둔 반복적인 형태이다. 1990년 Rivest가 MD4[20][21] 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[4]에 의한 마지막 2 라운드에 대한 공격에 의해 MD4의 취약점이 발견됨에 따라 1991년에 다시 Rivest에 의해 MD4의 취약점을 개선시킨 MD5[22]가 제안되었다. 또한 유럽의 RIPE 컨소시움에서 MD4와 MD5에 대한 독립적인 평가에 기초하여 MD4의 강화된 버전으로 RIPEMD[19]를 1995년에 제안했지만 Dobbertin[6]에 의해 RIPEMD의 축소된 버전에 대한 공격이 1996년에 발견됨에 따라 이를 개선한 RIPEMD-128/160[8]이 Dobbertin, Bosselasers, Preneel에 의해 제안되었다. 이 Dobbertin의 공격 방법은 RIPEMD 외에 MD4와 MD5에 대한 공격으로 확장 가능하여 이들 해쉬 함수에 심각한 위험이 있음을 보였다[7][9]. 1993년 미국의 NIST (National Institute of Standard and Technology)에서 FIPS PUB 180으로 SHA(Secure Hash Algorithm)[13]을 공개하였고 1995년 자체적으로 발견된 약점을 보완하여 SHA-1[14]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[26]이 설계되었고 국내에서는 이필중 교수팀[29] 등이 한국형 디지털 서명 방식에 사용될 수 있는 해쉬 함수로 PMD-V와 PMD-N 을 제안하였다.

메시지 인증 코드(Message Authentication Code : MAC)는 데이터 무결성과 데이터 출처 인증에 널리 사용된다. MAC에 대한 첫 번째 구성은 블록 암호의 CBC(Cipher Block Chaining)와 CFB(Cipher FeedBack) 모드에 기반한 것이다. 대부분의 표준과 응용은 CBC 모드를 사용한 것이고 이에 대한 이론적인 연구가 최근에 Bellare 등에 의해 수행되었다[2]. 또 다른 제안은 Message Authenticator Algorithm(MAA)으로 ISO 표준이며 소프트웨어로 비교적 빠르게 동작한다. 하지만 32 비트 출력 결과가 다양한 응용에 대해 매우 짧다는 단점이 있다. 90년대 이후 매우 빠른 해쉬 함수가 제안되었고 이들에 기반한 MAC이 다른 가능한 구성보다 성능이 좋고, 추가적인 구현 노력이 매우 적은 요인 등으로 MAC 구성에 많이 사용되며, Kerboros와 SNMP(Simple Network Management Protocol)에 채용되었다.

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 제안하고 이를 이용한 효율적인 MAC의 구성 방법을 제안한다. 제안된 알고리즘은 512 비트 블록 단위로 처리하고 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[23]. 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. 또한 제안한 해쉬 함수를 이용한 MAC 구성 알고리즘을 제안한다. 제안된 MAC 구성은 일반적인 MD 계열 해쉬 함수를 이용하여 구성될 수 있으며 해쉬 함수의 내부 구조를 최소한으로 변경하면서 해쉬 함수의 안전성을 그대로 유지하도록 설계되었다. 기반이 되는 해쉬 함수가 안전하다면 MAC에 대한 최상의 공격은 키에 관하여는 exhaustive 탐색이고 위조에 관하여는 Preneel에 의해 발견된 다음의 공격이다[17].

[정리 1] h를 n 비트 연쇄 변수, m 비트 결과, random 함수처럼 동작하는 압축 함수 f, 출력 변환 g를 가진 반복적인 MAC이라 하자. h에 대한 내부 충돌은 u 개의 알려진 평문-MAC 쌍과 v 개의 선택된 평문을 사용하여 발견될 수 있다. 각 평문은 같은 substring $s \geq 0$의 trailing 블록

을 가진다. u와 v의 기대값은 다음과 같다 :

$$u = \sqrt{2/(s+1)} \cdot 2^{n/2} \ ;$$

만약 출력 변환 g가 치환이거나 $s+1 \geq 2^{n-m+6}$ 이면, $v=0$ 이고

다른 경우는 $v = 2 \cdot \frac{2^{n-m}}{(s+1)} + 2\lceil \frac{n-\log_2(s+1)}{m} \rceil$

본 논문은 2절에서 해쉬 함수와 MAC의 정의를 간단히 기술하고, 3절에서 제안된 알고리즘의 세부 사항과 설계 기준 그리고 기존 MD 계열 해쉬 함수와의 성능 비교를 제시하며 4절에서는 제안된 해쉬 함수를 이용한 MAC 구성 알고리즘을 제안한다. 마지막으로 5절에서는 결론과 추후 연구 방향을 기술한다.

2. 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[16].

- ***preimage resistance*** : 어떤 미리 명시된 출력으로 해쉬하는 어떤 입력을 발견하는 것이 계산상 수행불가능하다. 즉, 해쉬 값 y가 주어졌을 때, h(x)=y를 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.

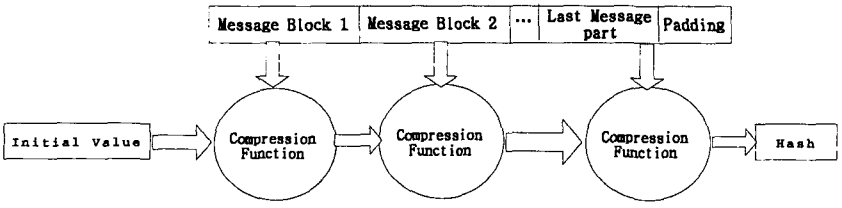
암호학적으로 유용한 해쉬 함수는 다음 성질을 추가로 만족해야 한다.

- ***collision resistance*** : 충돌(같은 결과로 해쉬하는 두 개의 다른 입력)을 발견하는 것이 계산상 수행불가능하다. 즉, h(x)=h(x')을 만족하는 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_1에서 X_t로 나누어진다. 해쉬 함수 h는 다음처럼 기술된다.

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

여기서 f는 압축 함수(compress function)이고, H_i는 단계 $i-1$ 과 단계 i 사이의 연쇄 변수(chaining variable)이고, IV는 초기값(Initial Value)이다. 압축 함수를 사용한 반복적인 해쉬 함수의 블록도가 (그림 1)에 주어졌다.



(그림 1) 반복적인 해쉬 함수에서 압축 함수의 사용

해쉬 값의 계산은 연쇄 변수에 의존한다. 해쉬 계산을 시작할 때, 이 연쇄 변수는 알고리즘의 일부로 명시된 고정된 초기값을 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. DES 나 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[7]의 공격에 의해 완전히 해독되었고 MD5 역시 심각한 취약점이 발견되었다[9]. 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[8]은 두 개의 병렬 라인으로 처리하여 안전성을 향상시킨다는 특징을 가지고 있으며 HAVAL[26]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 라안으로 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[28].

MAC은 두 번째 입력, 즉 비밀키 K를 가진 해쉬 함수이다. h와 입력 x, 그리고 비밀키 K가 주어지면 $h(x)$ 계산은 쉬워야 한다. MAC에 부과되는 가장 강한 조건은 비밀키를 알지 못하는 누군가가 실존의 위조(existential forgery : 임의의 메시지와 그에 대응하는 MAC을 발견하는 것)를 수행하는 것이 계산량적으로 수행불가능해야 한다는 것이다. 이것은 공격자가 자신이 선택한 메시지에 대해 MAC을 결정할 수 있는 선택 위조(selective forgery)와 달라야 한다. 실제 공격에 대해 위조는 검증 가능할 것을 요구하며 이상적인 MAC의 경우에 키를 발견하기 위한 방법은 k-비트 키에 대해 $O(2^k)$ 연산의 exhaustive 탐색과 같은 비용을 가진다. 또한 그러한 공격의 검증을 위해 요구되는 평문-MAC 쌍의 수는 k/m 이며 정확한 키를 확인한 공격자는 어떤 메시지에 대해 MAC을 계산할 수 있다.

암호 해쉬 함수에 기반한 MAC 구성은 Tsudik[25]에 의해 제안되었다. Tsudik은 메시지 M에 대해 다음의 3가지 구성 방법을 제안하고 그 안전성을 분석하였다.('||' 은 스트링의 연결(concatenation)이다)

secret prefix method : $MAC(M) = MD(K \| M)$, K 는 512 비트 비밀키

secret suffix method : $MAC(M) = MD(M \| K)$, K 는 512 비트 비밀키

envelope method : $MAC(M) = MD(M \| K_2)$, 초기값 $IV = K_1$은 128 비트 비밀키, K_2는 512 비트 비밀키

Preneel과 Oorschot는 몇 가지 MAC 구성의 성질과 취약점을 분석했다[17][18]. 특히, 그들은 반복적인 구성에서 birthday 공격 효과를 상세히 기술했고 MDx-MAC을 제안했다[17]. Kaliski와 Robshaw는 다양한 형태의 구성을 비교 논의했다[10]. Bellare 등은 NMAC(Nested MAC)과 HMAC(Hash based MAC)을 제안하고 안전성과 효율성을 분석하였다[1]. 본 논문에서는 기본적으로 MDx-MAC에 기반한 MAC 구성 기법을 제안한다. 제안된 MAC 구성의 특징은 각 라운드에서 사용되는 메시지 워드의 적용 순서를 키 요소를 이용하여 발생시킨 random 치환을 사용하여 메시지 워드의 변화를 랜덤화시키는데 있다.

3. 새로운 해쉬 알고리즘 기술과 설계 기준

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. 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
- ◇ $X^{\lll s}$: 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 비트이다. 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]. 전체 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.

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

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.

(5) 메시지 워드의 적용 순서

메시지 워드 적용 순서는 RIPEMD-160의 설계 원리를 참조하였다[8]. 추가로 생성된 워드들이 충분히 분산되도록 하고 동일한 워드들이 각 라운드마다 인접하지 않고 각 라운드의 각 단계에서 동일한 워드가

사용되지 않도록 하였다.

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	13	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.

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

(7) 부울 함수(boolean function)

부울 함수는 HAVAL의 방법에 기초한다. 기존 MD 계열의 3 변수 부울 함수 대신에 HAVAL에서와 유사한 암호학적으로 강한 성질들을 가지는 5 변수 부울 함수를 사용한다[26][29]. 각 라운드의 단계 연산에서 사용되는 부울 함수는 다음과 같다.

$$f_0(x_1, x_2, x_3, x_4, x_5) = (x_1 \wedge x_2) \oplus (x_3 \wedge x_4) \oplus (x_2 \wedge x_3 \wedge x_4) \oplus x_5$$
$$f_1(x_1, x_2, x_3, x_4, x_5) = x_2 \oplus ((x_4 \wedge x_5) \vee (x_1 \wedge x_3))$$
$$f_2(x_1, x_2, x_3, x_4, x_5) = x_1 \oplus (x_2 \wedge (x_1 \oplus x_4)) \oplus (((x_1 \wedge x_4) \oplus x_3) \wedge x_5)$$

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[24]. 부울 함수 f_1 과 f_2 역시 0-1 balanced를 만족하며 높은 비선형성을 가지며 SAC을 만족하도록 설계되었다[24][26]

각 라운드에서 부울 함수의 적용 순서는 f_0, f_1, f_2, f_1으로 계산량이 가장 적은 f_1을 중복 사용하여 효율성을 고려하였다.

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

(8) 순환 이동

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.

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

$$s = X_i \mod 32$$

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

제안된 알고리즘은 기존에 알려진 den Boer와 Bosselaers의 공격[4]과 Dobbertin의 공격[6][7][9]에 대해 안전하다고 여겨지고 데이터 의존 순환이동으로 인해 각 라운드에서 비트들이 임의 위치로 순환이동하기 때문에 차분 공격(differential cryptanalysis)[3]과 선형 해독(linear cryptanalysis)[11]에 안전할 것이다[23]. 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)의 성능과 비교하였다. 각 알고리즘은 C 언어로 구현하였으며 펜티움 100MHz에서 실험하였고 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 27% 빠르며 SHA-1보다 약 2% 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[28].

4. 제안된 해쉬 함수를 이용한 MAC 구성

위에서 제안한 해쉬 함수를 이용한 새로운 MAC 구성을 제안한다. 다음의 설계 목표를 가지고 MAC 알고리즘을 구성한다[17].

(1) 비밀키는 해쉬 함수의 시작, 끝 그리고 모든 반복에 관계되어야 한다.
(2) 원래 해쉬 함수로부터의 변경이 최소화되어야 한다. 구현 노력을 최소화하고 이미 얻어진 신뢰를 최대화한다.
(3) 성능은 해쉬 함수의 성능에 가까워야 한다.
(4) 추가적인 메모리 요구 사항이 최소화되어야 한다.
(5) 접근법이 일반적이어야 한다. MD 계열의 다른 해쉬 함수에도 적용 가능해야 한다.

새로운 MAC 알고리즘은 최대 160 비트의 키를 사용한다. 해쉬 함수는 MD 계열의 어느 해쉬 함수를 사용해도 가능하다(단지 각 해쉬 함수의 연쇄 변수의 크기에 맞는 키 크기를 사용한다). 제안된 MAC은 세 단계로 구성된다.

MAC 구성의 첫 번째 단계는 키의 추출이다. 키 K가 160 비트 이하라면, 키 K를 충분한 비트 수만큼 그 자체를 연결하여 블록 크기 512 비트로 만들어 해쉬 함수에 적용하여 다음처럼 MAC에서 사용할 160 비트의 키를 구성한다.

$$key = hash(\overline{K})$$

여기서 $\overline{K}$는 키 K를 연결하여 만든 512-비트 블록이고, padding과 길이 추가는 생략하고 해쉬한다.

두 번째 단계로 *key*를 다시 *key*1과 *key*2 두 부분으로 나눈다. 첫 번째 부분 *key*1은 각 라운드에서 적용되는 메시지 워드 순서를 치환하기 위해 사용하는 부분이고, 나머지 부분 *key*2은 각 라운드에서 사용되는 상수와 결합하는 부분이다. 메시지 워드 순서에 대한 random 치환을 발생하기 위해, 크기가 m인 치환과 $0 \sim (m! - 1)$ 사이의 정수를 일대일 대응시키는 Knuth 알고리즘을 이용하여 다음의 선형 합동법

$$Y = X + Q \mod m!$$

에서 발생되는 난수와 일대일 대응되는 치환을 알고리즘에 적용 후 이 알고리즘의 출력 치환 2개를 결합하여 사용한다[27]. 제안된 해쉬 함수는 각 라운드에서 24개 메시지 워드를 사용하기 때문에 160-비트

*key*의 왼쪽 끝으로부터 10-바이트를 사용한다(실제 75-비트). 다음과 같이 각 라운드에서 사용할 메시지 워드의 순서 ρ_i 를 발생시킨다.

[Round 1]
$X1 = key1 + Q \mod 24! \leftrightarrow P1$
$X2 = X1 + Q \mod 24! \leftrightarrow P2$
$\rho_1 = P1 \circ P2$

[Round 2]
$X3 = X2 + Q \mod 24! \leftrightarrow P1$
$X4 = X3 + Q \mod 24! \leftrightarrow P2$
$\rho_2 = P1 \circ P2$

[Round 3] 과 [Round 4]는 [Round 2]와 유사한 방법으로 치환을 발생시킨다. 여기서 Q는 상수로 $m!$과 서로 소인 $m!$ 이하의 정수 값이다.
다음으로 상수값을 수정하는 부분은 *key1* 다음의 8-바이트를 사용한다(*key2*). 각 2 바이트씩으로 분할한 후, 각각을 반복 연결하여 32-비트 워드로 만들어 각 라운드의 상수 K_i 에 더한다.

마지막 단계는 실제 MAC의 계산 단계로 메시지 M의 앞뒤에 키 요소로 추가하여 해쉬 함수에 적용한다.

$$MAC = hash(\overline{key \oplus 0x5c} \parallel \overline{M} \parallel \overline{key \oplus 0x36})$$

여기서 $\overline{M}$ 은 padding과 메시지 길이가 추가된 형태이고 $hash()$는 위에서 키에 의해 random 치환된 메시지 워드 적용 순서와 수정된 상수를 적용한 해쉬 함수이다.
MAC 결과는 해쉬값의 왼쪽부터 m 비트이다. 이것은 1절 [정리 1]의 공격 관점에서 $m = n/2$의 크기가 대부분의 응용에 대해 추천되고 있다.

제안된 MAC 구성의 계산 오버헤드는 키 추출에 사용된 한 블록의 해쉬 계산과 각각 메시지의 앞뒤에 추가된 두 블록의 해쉬 계산, 그리고 메시지 워드 적용 순서 변경을 위한 random 치환 발생이다. 치환 발생은 24!의 값을 가지기 때문에 다정도 연산(multiprecision operation)을 요구하지만 실제 다정도 연산은 75-비트 *key1* 값을 factorial number system으로 변경시킬 때 한번 나눗셈 (multiprecision / int)과 모듈러 (multiprecision mod int)연산이 사용되고 그 이후에는 사용되지 않는다. 따라서 치환 발생은 짧은 시간 안에 발생시킬 수 있다. 실제 소프트웨어 구현시 제안된 해쉬 함수과 비교할 때 단지 10% 정도의 성능 저하를 보일 뿐이다.

마지막 단계에서 메시지 앞뒤에 추가된 키 요소는 envelope 방법의 K_1 , K_2와 유사하다. 그렇지만 divide-and-conquer 공격에 안전하다는 차이를 가진다[17]. 이는 MDx-MAC의 방법과 유사한 방법이다. 그리고 상수 값에 키 요소를 결합시킨 의도는 해쉬 함수의 취약성이 알려진 경우에 envelope 방법에 추가적인 안전성을 제공하는 것이다. 이것 자체는 안전성에 있어 그다지 강한 요소는 아니다. 각 라운드에서의 메시지 워드 적용 순서의 random 치환은 trapdoor 일방향 함수로 ρ_i를 계산하는 것은 쉽지만 그 역은 계산량적으로 어렵다. 또한 각 라운드의 적용 순서가 이전 라운드와 같은 순서 또는 이전 라운드의 역순을 가질 확률은 random 치환의 주기가 24!(상수 Q가 24!에 서로 소인 정수일 경우)로 약 10^{23} 이기 때문에 무시할 수 있다. 그리고 160-비트의 키 K는 exhaustive 탐색에 안전하고, 기반이 되는 해쉬 함수가 안전하다면 키 *key* 에 대한 탐색은 키 K에 대한 exhaustive 탐색만큼 어렵다. 권고되어지는 것

처럼 MAC 결과가 연쇄변수 크기의 절반과 같다면(즉, $m=n/2$), [정리 1]에 의해 제안된 기법에 대한 위조 공격은 $O(2^m/(s+1))$ 선택된 평문-MAC 쌍과 $O(2^m/\sqrt{(s+1)})$ 알려진 평문을 요구한다. 또한 160-비트의 키 크기는 envelope 방법에 대해 이전에 제안된 160+512=672 비트보다 이점이 있다. 그리고 제안된 기법은 해쉬 함수의 내부 구조를 이용한 공격에 대해 각 라운드의 적용되는 메시지 워드 순서를 랜덤하게 비밀로 할 수 있어 더 안전하다고 추측한다. 제안된 MAC의 안전성에 대해 다음과 같이 추측한다.

[추측] 기반이 되는 해쉬 함수가 안전하다면, 제안된 MAC에 대한 최상의 공격은 키에 관하여 exhaustive 탐색, 위조에 관하여는 [정리 1]의 공격이 가능할 뿐이다.

5. 결론

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 27% 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.

그리고 제안된 해쉬 함수에 기반한 MAC 구성 알고리즘을 제안하였다. 제안된 MAC은 최대 160-비트의 키를 사용하고 해쉬 결과의 절반(약 80-비트)의 MAC 결과를 가진다. 제안된 MAC은 해쉬 함수의 구조를 최소한으로 변경하면서, 키가 해쉬 함수의 모든 요소에 관련되도록 설계되었다. 특히 워드 순서를 키에 의해 랜덤하게 라운드마다 적용함으로서 메시지 워드의 순서에 의한 공격 방식[4][6][7][9]과 같은 가능한 공격에 대해 강인하도록 설계하였다. 소프트웨어 구현시 원래 해쉬 함수의 성능에 비해 약 10% 정도의 성능 저하를 보인다. 제안된 MAC의 안전성으로 기반이 되는 해쉬 함수가 안전하다면, 제안된 MAC에 대한 최상의 공격은 키에 관하여 exhaustive 탐색, 위조에 관하여는 [정리 1]의 공격이다라고 추측한다.

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. 아울러 제안된 MAC 구성 알고리즘의 안전성에 관하여도 계속적인 분석과 연구를 추진할 것이다.

[참 고 문 헌]

[1] M. Bellare, R. Canetti, H. Krawczyk, Keying Hash Functions for Message Authentication, Advances in Cryptology-Crypto'96, Lecture Notes in Computer Science, vol.1109, Springer-Verlag, 1996,

[2] M. Bellare, J. Kilian, P. Rogaway, The security of cipher block chaining, Advances in Cryptology-Crypto'94, Lecture Notes in Computer Science, vol.839, Springer-Verlag, 1994

[3] 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

[4] 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

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

[6] H. Dobbertin, RIPEMD with two-round compress function is not collision-free, Journal of

Cryptology, vol.10, no.1, 1997, pp. 51-69

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

[8] 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

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

[10] B. Kaliski, M Robshaw, Message Authentication with MD5, RSA LABs' CryptoBytes, vol.1, no.1, Spring, 1995

[11] 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

[12] 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

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

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

[15] 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

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

[17] B. Preneel, P. van Oorschot, MDx-MAC and Building Fast MACs from Hash Functions, Advances in Cryptology-Crypto'95, Lecture Notes in Computer Science, vol.963, Springer-Verlag, 1995

[18] B. Preneel, P. van Oorschot, On the security of two MAC algorithms, Advances in Cryptology-Eurocrypt'96, Lecture Notes in Computer Science, vol.963, Springer-Verlag, 1996

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

[20] 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

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

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

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

[24] 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, vol.718, Springer-Verlag, 1993, pp. 145-154

[25] G. Tsudik, Message authentication with one-way hash functions, Proceedings of Infocom 92

[26] 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

[27] 고승철, 이경현, 랜덤 치환 고속 발생기, Proceedings of the 1-st workshop in Applied Mathematics, 1993, pp. 379-384

[28] 신상욱, 류대현, 이상진, 이경현, MDx-계열 해쉬 함수에 기반한 새로운 해쉬 함수, 정보처리학회 '97 추계 학술발표 논문집, vol.4, no.2, 1997. 10, pp. 1354-1359

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