情報保護學會論文誌
第19卷 第2號, 2009. 4

HMAC/NMAC-MD4에 대한 향상된 키 복구 공격*

강 진 건,[1†] 이 제 상,[1] 성 재 철,[2‡] 홍 석 희,[1] 류 희 수[3]
[1]고려대학교, [2]서울시립대학교, [3]경인교육대학교

Improved Key-Recovery Attacks on HMAC/NMAC-MD4*

Jinkeon Kang,[1†] Jesang Lee,[1] Jaechul Sung,[2‡] Seokhie Hong,[1] Heuisu Ryu[3]
[1]Korea University, [2]University of Seoul, [3]Gyeongin National University of Education

요 약

2005년 Wang이 MD 계열 전용 해쉬함수에 대한 분석 결과를 발표함에 따라, MD 계열 전용 해쉬함수를 이용한 메시지 인증 프로토콜에 대한 분석 결과 또한 다수 발표되었다. CRYPTO'07에서 Fouque, Leurent, Nguyen은 Wang의 충돌쌍 공격을 이용한 MD4, MD5 기반 HMAC/NMAC의 내부 키 복구 공격과 외부 키 복구 공격을 소개하였다[4]. EUROCRYPT'08에서 Wang, Ohta, Kunihiro는 근사 충돌쌍(near-collision)을 이용한 MD4 기반 HMAC/NMAC의 외부 키 복구 공격을 소개하였다[2]. 즉, Wang 등은 새로운 근사 충돌쌍 특성을 이용하여 기 제안된 Fouque의 분석 방법을 향상시킴으로서 오라클 질의량을 2^{88}에서 2^{72}으로 감소시켰으며, MD4 계산량 또한 2^{95}에서 2^{77}으로 감소시켰다. 본 논문에서는 분할-정복(divide and conquer) 공격을 이용한 새로운 MD4 기반 HMAC/NMAC의 외부 키 복구 공격 알고리즘을 제시한다. 본 논문에서 제시하는 분할-정복 공격을 이용한 공격알고리즘은 $2^{77.1246}$의 오라클 질의량과 2^{37}의 MD4 계산량으로 HMAC/NMAC의 외부 키를 복구할 수 있는 향상된 공격 방법이다.

ABSTRACT

In 2005, Wang et al. discovered devastating collision attacks on the main hash functions from the MD4 family. After the discovery of Wang, many analysis results on the security of existing hash-based cryptographic schemes are presented. At CRYPTO'07, Fouque, Leurent and Nguyen presented full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5[4]. Such attacks are based on collision attacks on the underlying hash function, and the most expensive stage is the recovery of the outer key. At EUROCRYPT'08, Wang, Ohta and Kunihiro presented improved outer key recovery attack on HMAC/NMAC-MD4, by using a new near collision path with a high probability[2]. This improves the complexity of the full key-recovery attack on HMAC/NMAC-MD4 which proposed by Fouque, Leurent and Nguyen at CRYPTO'07: The MAC queries decreases from 2^{88} to 2^{72}, and the number of MD4 computations decreases from 2^{95} to 2^{77}. In this paper, we propose improved outer key-recovery attack on HMAC/NMAC-MD4 with $2^{77.1246}$ MAC queries and 2^{37} MD4 computations, by using divide and conquer paradigm.

Keywords: HMAC, NMAC, MD4, Key-Recovery, Near Collision, Differential Path

I. 서 론

해쉬함수는 임의의 길이의 비트열(메시지)을 입력받아 고정된 길이의 비트열(해쉬값)을 출력하는 메시지 압축함수로, 전자서명, 메시지 인증, 전자화폐, 키 생성 알고리즘 등 여러 분야에서 암호시스템의 핵심 요소로서 널리 사용되고 있다. 해쉬함수가 암호시스템

접수일(2008년 12월 20일), 수정일(2009년 4월 9일), 게재확정일(2009년 4월 9일)
* 이 논문은 2009년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임. (No. 2009-0060420)
† 주저자, jkkang@cist.korea.ac.kr
‡ 교신저자, jcsung@uos.ac.kr

(프로토콜)에서 암호학적으로 안전하게 이용되기 위해서는 기본적으로 다음과 같은 세 가지 성질을 만족하여야 한다.

- o 역상저항성: y가 주어졌을 때, $h(x)=y$를 만족하는 x를 찾는 것이 계산상 불가능하다.
- o 제 2 역상저항성: $h(x)=y$을 만족하는 x, y가 주어졌을 때, $x \neq x'$, $h(x')=y$를 만족하는 x'을 찾는 것이 계산상 불가능하다.
- o 충돌 저항성: $h(x)=h(x')$을 만족하는 서로 다른 x, x'을 찾는 것이 계산상 불가능하다.

위의 세 가지 성질을 모두 만족해야만 해쉬함수는 암호시스템에서 안전하게 사용될 수 있다. 하지만, 2005년 중국의 Wang은 현재 가장 널리 사용되고 있는 전용 해쉬함수들(MD4, MD5, HAVAL, RIPEMD, SHA-0/1)에 대한 일반화된 충돌쌍 공격 기법을 소개하였다[6-9]. Wang의 충돌쌍 공격이 소개된 이후, 많은 암호학자들은 Wang의 공격기법을 토대로 더욱 더 강력하고 일반화된 해쉬함수 분석방법들을 소개하였다. 더 나아가 Wang의 분석기법을 토대로 완성된 해쉬함수 분석방법은 해쉬함수가 기반이 되는 실제 프로토콜의 취약점 분석에도 이용되었다. FSE'07에서 Leurent는 Wang의 충돌쌍 공격을 이용하여 MD4와 MD5에 기반을 둔 APOP 프로토콜의 비밀키 복구 공격을 소개하였으며[1], AFRICACRYPT'08에서 Sasaki 등은 블록암호의 불능차분공격 기법을 이용하여 MD4 기반 도전-응답 프로토콜의 비밀키 복구 공격을 소개하였다[10,11].

ASIACRYPT'06에서 Contini 등은 해쉬함수의 충돌쌍 공격을 이용하여 MD4, MD5, SHA-0, reduced SHA-1을 기반으로 하는 HMAC/NMAC에 대한 구별 공격, 내부 키 복구 공격 및 위조 공격을 소개하였다[5]. CRYPTO'07에서 Fouque 등은 MD4와 MD5를 사용하는 HMAC/NMAC에 대한 외부 키 복구 공격을 소개하였다[4]. 기존에 제시된 키 복구 공격은 충돌쌍을 발생시키는 차분 경로를 이용하였다. 즉, 충돌쌍을 발생시키는 차분 경로를 이용하여 공격을 시도하였다. 그러나 키 복구 공격의 경우 충돌쌍을 발생시키는 차분 경로가 아니라 높은 확률로 특정 차분을 발생시키는 차분 경로가 존재한다면 좀 더 효율적으로 공격을 성공시킬 수 있다.

EUROCRYPT'08에서 Lei Wang 등은 그들이 발견한 MD4 근사 충돌쌍 공격을 이용한 HMAC/NMAC-MD4 키 복구 공격과, MD5 pseudo 충돌쌍 공격을 이용한 HMAC/NMAC-MD5 키 복구 공격을 소개하였다[2].

본 논문에서는 EUROCRYPT'08에서 Lei Wang 등이 소개한 HMAC/NMAC-MD4 키 복구 공격을 향상시킨 분석방법을 소개한다. Lei Wang 등은 근사 충돌쌍을 발생시키는 새로운 차분 경로를 제시하고, 외부 키를 복구하기 위하여 근사 충돌쌍 특성을 이용하였다. 하지만 Lei Wang의 HMAC/ NMAC-MD4 키 복구 공격의 경우, 외부 키 복구 알고리즘에서 공격자가 복구하고자 하는 외부 키 128비트 중 특성식이 존재하지 않는 비트들을 전수조사를 통해 복구하기 때문에 매우 비효율적이다. 본 논문에서는 HMAC/NMAC-MD4의 외부 키 128비트를 특성식을 기준으로 하위 비트들로 분할하고 순차적으로 부분 키 비트를 복구하는 분할-정복(divide and conquer) 공격 알고리즘을 제시한다. [표 1]은 본 논문에서 제시하는 외부 키 복구 공격 알고리즘을 이용한 분석 결과와 기존의 결과를 비교한 것이다.

[표 1] 기 제안된 분석 결과와 본 논문의 분석 결과 비교

HMAC/NMAC-MD4	Fouque [4]	Lei Wang [2]	본 논문
온라인 질의 복잡도	2^{88}	2^{72}	$2^{77.1246}$
오프라인 MD4 복잡도	2^{95}	2^{77}	2^{37}

본 논문은 구성은 다음과 같다. 2장에서는 공격을 소개하기에 앞서 MD4 알고리즘과, HMAC/NMAC 알고리즘을 소개한다. 3장에서는 기 제안된 HMAC/NMAC 공격 과정을 간략히 소개하고, 4장에서는 3장에서 제시된 공격 과정을 토대로 향상된 외부 키 복구 공격 알고리즘을 소개하고 공격 복잡도를 계산한다. 5장은 결론이다.

II. 배경 이론

본 절에서는 MD4 해쉬함수를 간략히 소개하고, HMAC/NMAC 알고리즘에 대하여 소개한다.

2.1 MD4 해쉬함수

MD4는 1990년 Rivest에 의하여 개발된 해쉬함수로서 임의의 길이 메시지를 입력받아 128비트 해쉬값을 출력한다. 임의의 메시지가 주어지면 MD4 해쉬

〔표 2〕 MD4 압축함수의 메시지 확장 과정

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
π_i $(0 \le i < 16)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
π_i $(16 \le i < 32)$	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11	15
π_i $(32 \le i < 48)$	0	8	4	12	2	10	6	14	1	9	5	13	3	11	7	15

〔표 3〕 MD4 압축함수의 i번째 단계에 필요한 순환 이동값

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s_i $(0 \le i < 16)$	3	7	11	19	3	7	11	19	3	7	11	19	3	7	11	19
s_i $(16 \le i < 32)$	3	5	9	13	3	5	9	13	3	5	9	13	3	5	9	13
s_i $(32 \le i < 48)$	3	9	11	15	3	9	11	15	3	9	11	15	3	9	11	15

〔표 4〕 MD4 압축함수

o 단계함수: $b_i = (b_{i-1} \boxplus f_i(b_{i-1}, c_{i-1}, d_{i-1}) \boxplus m_k \boxplus k_i) \lll s_i$

$a_i = d_{i-1}, c_i = b_{i-1}, d_i = c_{i-1}$

- 입력값: $a_0 \| b_0 \| c_0 \| d_0$
- 출력값: $a_0 \boxplus a_{48} \| b_0 \boxplus b_{48} \| c_0 \boxplus c_{48} \| d_0 \boxplus d_{48}$

함수는 메시지의 길이가 512비트의 배수가 되도록 덧붙이기(padding)를 수행하고, Merkle-Damgård 구성 방법을 이용하여 128비트 해쉬값을 출력한다. MD4의 압축함수는 32비트 워드 단위로 동작하며 효율성을 극대화하기 위해서 다음과 같은 단순한 연산들의 조합으로 설계되었다.

- o $\lll$: 왼쪽 순환 이동 연산
- o $\boxplus$: 워드간의 2^{32}을 법으로 하는 덧셈 연산
- o $\neg, \vee, \wedge, \oplus$: 각각 비트별 NOT, AND, OR, XOR 연산
- o $A \mathrel{+}= B$: 임의의 A, B에 대하여 A와 B를 더한 값을 A에 입력

MD4 압축함수는 3라운드로 구성되며 각 라운드는 16단계로 수행된다. 각 라운드는 다음과 같은 비선형 부울함수를 이용한다. 단, X, Y, Z는 32비트 워드이다.

- o 1라운드 $(0 \le i < 16)$:
  $f_i(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$
- o 2라운드 $(16 \le i < 32)$:
  $f_i(X, Y, Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$
- o 3라운드 $(32 \le i < 48)$:
  $f_i(X, Y, Z) = X \oplus Y \oplus Z$

MD4 압축함수에 입력된 512비트 메시지는 16개의 메시지 워드로 분할되며, 메시지 확장과정을 통하여 48개의 메시지 워드로 확장된다. MD4 해쉬함수의 메시지 확장과정은 다음과 같이 메시지를 재배열하며 메시지 재배열 규칙은 〔표 2〕와 같다.

$m_i = M_{\pi_i}$.

MD4 압축함수는 4개의 내부 상태값을 가지며, 각 단계마다 1개의 내부 상태값이 갱신되며 본 연구에서는 i번째 단계에 갱신되는 내부 상태값을 a_i, b_i, c_i, d_i라 한다. k_i와 s_i를 각각 i번째 단계에 사용되는 32비트 상수와 순환 이동값이라고 한다면 MD4 압축함수는 〔표 4〕와 같이 정의된다. 각 단계별 순환 이동값 s_i는 〔표 3〕과 같다.

2.2 HMAC/NMAC 알고리즘

HMAC과 NMAC은 Bellare, Canetti, Krawczyk에 의하여 제안된 해쉬함수 기반 메시지 인증 코드〔3〕로서 다음과 같은 이유에서 매우 흥미로운 연구분야다.

- o HMAC은 표준화됨: ANSI, IETF, ISO, NIST
- o HMAC은 다양한 프로토콜 환경에서 이용됨:

SSL, TLS, SSH, IPsec
- o HMAC과 NMAC는 안정성 증명이 이루어짐
- o HMAC과 NMAC는 단순한 구조를 가지고 있음

M은 메시지, k는 비밀키, $\bar{k}$는 해쉬함수의 블록 크기만큼 k에 0을 덧붙이기 한 값($k \| 00...0$)이라고 할 때, HMAC은 다음과 같이 정의된다.

$$HMAC_k(M) = H(\bar{k} \oplus opad \| H(\bar{k} \oplus ipad \| M)).$$

단, opad와 ipad는 고정된 1 블록 크기의 상수 값이다. (k_1, k_2)를 비밀키라고 할 때, NMAC은 다음과 같이 정의된다.

$$NMAC_{k_1,k_2}(M) = H_{k_1}(H_{k_2}(M)).$$

따라서 덧붙이기에 포함되는 메시지 길이의 변화를 제외하면 HMAC은 $NMAC_{H(\bar{k} \oplus opad),\, H(\bar{k} \oplus ipad)}$로 정의할 수 있다. 우선적으로 NMAC에 대한 안정성 증명이 이루어졌으며, 이를 이용하여 HMAC에 대한 안전성으로 확장되었다. 본 논문에서는 k_1과 k_2를 각각 외부 키와 내부 키를 표기한다. 따라서 k_1과 k_2는 각각 외부 해쉬함수 $H_{k_1}(\cdot)$와 내부 해쉬함수 $H_{k_2}(\cdot)$에 대응한다.

III. 기 제안된 HMAC/NMAC-MD4 키 복구 공격 시나리오

본 절에서는 ASIACRYPT'06에서 Contini와 Yin이 제시한 내부 키 복구 공격[5]과 CRYPTO'07에서 Fouque, Leurent, Nguyen이 제시한 외부 키 복구 공격[4]을 소개한다. 더 나아가, EUROCRYPT'08에서 Wang, Ohta, Kunihiro가 제시한 향상된 외부 키 공격을 소개한다[2].

3.1 Contini와 Yin이 제안한 내부 키 복구 공격(CY 공격)[5]

Contini와 Yin은 MD4의 충돌쌍 공격을 이용한 내부 키 복구 공격을 소개하였다. Contini와 Yin은 내부함수 $H_{k_2}(\cdot)$가 충돌하면 반드시 $H_{k_1}(H_{k_2}(M))$가 충돌하는 성질을 이용하였다[5]. 즉, 공격자가 구성한 메시지 차분을 만족하는 서로 다른 메시지쌍 (M, M')에 대하여 내부함수 $H_{k_2}(\cdot)$의 결과값이 충돌할 확률은 매우 높으며 내부충돌이 발생하면 반드시 $NMAC(M) = = NMAC(M')$을 만족한다. 반면, 내부 충돌이 발생하지 않은 조건하에 외부 함수 $NMAC(M) = NMAC(M')$을 만족할 확률은 2^{-64}으로 매우 낮다. 따라서 메시지쌍 (M, M')에 대하여 $NMAC(M) = NMAC(M')$을 만족하면, 내부 충돌이 발생함을 확인할 수 있고, 공격자가 구성한 메시지 차분 ΔM과 메시지 차분 경로 DP을 이용하여 내부 키 k_2를 복구할 수 있다. Contini와 Yin이 제안한 내부 키 k_2의 복구 공격은 다음과 같다.

① MD4 충돌쌍 공격을 위한 메시지 차분 ΔM과 메시지 차분 경로 DP를 결정한다. n은 충돌쌍 공격을 성공시키기 위한 충분조건의 개수이다.
② 랜덤한 메시지 블록 M_i을 생성하고, HMAC/NMAC에 대한 충돌쌍 $(M_i, M_i + \Delta M)$을 찾을 때까지, 메시지쌍 $(M, M+\Delta M)$를 HMAC/NMAC 오라클에게 질의한다. 충분조건의 개수가 n개이기 때문에 평균적으로 2^n개의 질의를 하면 1개의 충돌쌍 $(M_i, M_i + \Delta M)$을 찾을 수 있다.
③ 충돌쌍 $(M_i, M_i + \Delta M)$의 충분조건을 이용하여, 내부함수 $H(k_2, M_i)$의 첫 번째 라운드 특정 t 단계에 갱신되는 내부 상태값 Q_t를 알 수 있다.
④ t 단계의 내부 상태값과 메시지 M_i값을 이용하여 MD4 단계함수의 역 연산을 통하여 내부키 k_2를 복구할 수 있다.

3.2 Fouque, Leurent, Nguyen이 제안한 외부 키 복구 공격(FLN 공격)[4]

Fouque, Leurent, Nguyen은 CRYPTO'07에서 MD4 기반 HMAC/NMAC에 대한 전체 키 복구 공격을 소개하였다[4]. Contini와 Yin이 제안한 내부 키 공격을 이용하여 내부 키를 복구하면, 외부 키 k_1를 찾을 수 있다. 복구된 내부 키 k_2를 이용하여 메시지 M에 대한 내부 해쉬함수의 결과값을 $H_{k_2}(M)$ 계산할 수 있다. 따라서 충돌쌍 공격을 위해 구성한 메시지 차분을 ΔM이라고 할 때, 생일 공격을 이용하여

$$H_{k_2}(M') = H_{k_2}(M) + \Delta M$$

을 만족하는 메시지 쌍 (M, M')을 생성할 수 있다. 생성된 메시지 쌍 (M, M')은 $H_{k_2}(M') = H_{k_2}(M) + \Delta M$을

만족하며, 비록 내부 충돌이 발생하지는 않더라도 외부 충돌이 발생한다. 즉, $NMAC(M)=NMAC(M')$을 만족할 확률이 매우 높다. 공격자는 내부 해쉬함수의 결과값 $H_{k_2}(M)$을 계산할 수 있으므로, 공격자가 구성한 메시지 차분 ΔM과 메시지 차분 경로 DP을 이용하여 외부 키 k_1을 복구할 수 있다. Fouque, Leurent, Nguyen이 제안한 외부 키 k_1의 복구 공격은 다음과 같다.

① MD4 충돌쌍 공격을 위한 메시지 차분 ΔM과 외부 키 k_1의 한 비트에 의하여 종속적으로 결정되는 충분조건을 갖는 메시지 차분 경로 DP를 결정한다. n은 충돌쌍 공격을 성공시키기 위한 충분조건의 개수이다.

② $H_{k_2}(M')=H_{k_2}(M)+\Delta M$을 만족하는 메시지쌍 (M, M')을 생성한다.

③ 메시지쌍 (M, M')을 HMAC/NMAC 오라클에게 질의한다. 충분조건의 개수가 n개이기 때문에 평균적으로 2^n개의 질의를 하면 1개의 충돌쌍 (M_i, M_i')을 찾을 수 있다. 충돌쌍 (M_i, M_i')에 대하여 외부 키 k_1의 특정비트는 충분조건을 만족한다. 따라서 2^{n+1}개의 질의를 통하여 외부 키 k_1의 한 비트를 복구 할 수 있다.

④ 메시지 차분 ΔM과 메시지 차분 경로 DP를 변경하여 외부 키 k_1의 다른 비트들을 복구한다.

3.3 Wang, Ohta, Kunihiro가 제안한 향상된 외부 키 복구 공격[2]

Wang, Ohta, Kunihiro는 EUROCRYPT'08에서 Fouque, Leurent, Nguyen이 제안한 외부 키 복구 공격을 근사 충돌쌍 공격을 이용하여 향상시켰다[2]. 즉, Fouque, Leurent, Nguyen이 외부 키 복구 공격을 위한 특성식으로 충돌쌍 경로를 이용하였다면 Wang, Ohta, Kunihiro는 외부 키 복구 공격을 위한 특성식으로 근사 충돌쌍 경로를 이용한다. Wang, Ohta, Kunihiro가 제안한 근사 충돌쌍 경로가 기 제안된 충돌쌍 경로보다 만족할 확률적으로 매우 높기 때문에 (충분조건의 개수가 작음), 외부 키 k_1을 복구하는 공격 복잡도가 매우 향상되었다.

본 논문에서는 Wang, Ohta, Kunihiro의 외부 키 복구 공격 개념을 이용하므로, 본 절에서 Wang, Ohta, Kunihiro의 외부 키 복구 공격을 자세히 살펴본다. 본 소절에서는 Contini와 Yin이 제안한 내부 키 공격을 이용하여 내부 키를 복구하였다고 가정한다. 각각의 워드를 32비트라고 할 때, HMAC/NMAC에 대한 외부 키 k_1은 (k_a, k_b, k_c, k_d)로 표기하고, 외부 해쉬함수 $H_{k_1}(\cdot)$의 48번째 단계의 출력값은 $(a_{48}, b_{48}, c_{48}, d_{48})$로 표기한다. 따라서 HMAC/NMAC의 출력값은 다음과 같다.

$$(h_a, h_b, h_c, h_d)=(k_a+a_{48}, k_b+b_{48}, k_b+c_{48}, k_b+d_{48}).$$

위 수식을 이용하여 다음의 성질을 이끌어 낼 수 있다.

$$\Delta h_a=\Delta a_{48},\ \Delta h_b=\Delta b_{48},$$
$$\Delta h_c=\Delta c_{48},\ \Delta h_d=\Delta d_{48}.$$

따라서 HMAC/NMAC의 출력값으로부터 외부해쉬함수 $H_{k_1}(\cdot)$의 출력 차분을 알 수 있으며, 결과적으로 마지막 네 단계(45-단계, 46-단계, 47-단계, 48-단계)에서 갱신되는 내부 상태값의 차분 형태를 알 수 있다. 위 특성을 이용하여 k_a, k_c, k_d의 특정 비트값들을 복구할 수 있다. 본 절에서는 Wang, Ohta, Kunihiro의 MD4 근사 충돌쌍 공격을 소개하고, 이를 이용한 MD4 기반 HMAC/NMAC 키 복구 공격을 소개한다.

3.3.1 MD4 근사 충돌쌍 공격

Wang, Ohta, Kunihiro의 근사 충돌쌍 공격을 위한 메시지 차분 형태는 다음과 같다[2].

$$\Delta M=M'-M=(\Delta m_0, \Delta m_1, \cdots, \Delta m_{15}),$$
$$\Delta m_3=2^i\,(i=3\sim5, 7\sim17, 20\sim25),$$
$$\Delta m_j=0(0\le j\le15, j\neq3).$$

즉, 메시지 m_3에만 차분이 존재하며, $i=3$일 때 메시지 차분에 대한 차분경로를 부록 A에 제시한다. 단지 $i=3\sim5, 7\sim17, 20\sim25$일 때만 주어진 차분경로가 유지된다. 그 밖의 경우에는 차분경로에서 발생하는 올림(carry)이 끊기기 때문에 차분경로를 만족하지 않는다. 위에서 제시한 메시지 차분에 대한 차분경로는 29번째 단계에서 충돌을 발생시키며, 45번째 단계에서 추가적으로 메시지 차분이 입력되기 때문에, 45번째 입력된 차분에 의하여 48번째 단계의 출력값의 차분 형태가 결정되며 근사 충돌을 발생시킨다. 즉,

MD4에 대한 근사 충돌쌍 형태는 메시지 m_3의 차분에 따라 결정된다.

3.3.2 온라인 과정(데이터 수집 과정)

메시지 차분 $\Delta M(\Delta m_3 = 2^i (i = 3 \sim 5, 7 \sim 17, 20 \sim 25))$을 만족하는 메시지 쌍 (M, M')들에 대하여 다음 과정에 따라 각각의 조건을 만족하는 메시지로 분류한다.

① 생일공격을 이용하여 $MD4(k_2, M') = MD4(k_2, M) + \Delta M$을 만족하는 메시지 쌍 (M, M')을 생성한다.

② 생성된 메시지 쌍 (M, M')을 HMAC/NMAC 오라클에 질의하고, 응답을 받아 근사 충돌이 발생하는 메시지 쌍 (M, M')을 다음과 같이 분류한다.

ⓐ $(M_a^i, M_a^{i\prime})$: HMAC/NMAC의 출력차분이 $\Delta h_a = 2^{i+3}, \Delta h_c = *\,2^{i+23} \pm 2^{i+14} \pm 2^{i+15}, \Delta h_d = *\,2^{i+12}$을 만족하는 메시지 쌍

ⓑ $(M_c^i, M_c^{i\prime})$: HMAC/NMAC의 출력차분이 $\Delta h_a = 2^{i+3}, \Delta h_c = *\,2^{i+23} * 2^{i+14}, \Delta h_d = *\,2^{i+12}$을 만족하는 메시지 쌍

ⓒ $(M_d^i, M_d^{i\prime})$: HMAC/NMAC의 출력차분이 $\Delta h_a = 2^{i+3}, \Delta h_c = *\,2^{i+14} \pm 2^{i+23} \pm 2^{i+24}, \Delta h_d = *\,2^{i+12}$을 만족하는 메시지 쌍

③ 가능한 모든 메시지 차분 i에 대하여 차분의 위치 i를 변경하여 ①과 ②의 과정을 반복한다.

단, $*$는 어떠한 부호여도 상관없다는 뜻이며, $\pm$는 부호의 형태가 + 혹은 −로 동일하게 유지되어야 함을 뜻한다. HMAC/NMAC의 출력값 중 3개의 워드가 조건을 만족하여야 하므로, 임의의 메시지쌍에 대하여 주어진 근사 충돌 형태를 가질 확률은 2^{-96}이다. 하지만, 주어진 메시지 차분 ΔM에 대하여 주어진 근사 충돌 형태를 가질 확률은 $2^{-64}(2^{-(60+4)})$이다. 29번째 단계까지의 차분 경로를 만족할 확률이 2^{-60}(충분조건의 개수가 60)이고, 근사 충돌을 발생하기 위한 확률이 2^{-4}이다. 자세한 설명은 [2]를 참고하여라.

위에서 제시한 데이터 수집과정을 통하여 수집된 데이터는 다음을 만족한다. 자세한 설명은 [2]를 참고하여라.

$$M_a^i : a_{48,i+3} = 1;\ M_c^i : c_{48,i+23} = 1;\ M_d^i : d_{48,i+12} = 1$$

위의 식에서 M_a^i, M_c^i, M_d^i에 대한 i 값은 다음의 조건일 때만 만족한다.

- o M_a^i: $i = 3 \sim 5, 7 \sim 15, 20 \sim 25$일 때만 만족
- o M_c^i: $i = 3 \sim 5, 9 \sim 17, 20 \sim 23$일 때만 만족
- o M_d^i: $i = 3 \sim 5, 9 \sim 17, 21 \sim 25$일 때만 만족

그 밖에는 차분 경로에서 발생하는 올림이 끊기기 때문에 차분경로를 만족하지 않는다. 여기서 i는 메시지의 차분이 존재하는 비트의 위치를 뜻한다.

3.3.3 오프라인 과정(키 복구 과정)

k_a, k_c, k_d를 복구하는 방법은 모두 동일하며 본 소절에서는 단지 k_a를 복구하는 방법만을 설명한다. k_a를 복구하는 과정은 다음과 같다.

- o Wang, Ohta, Kunihiro의 HMAC/NMAC 외부 키 $k_1 = (k_a, k_b, k_c, k_d)$ 복구 알고리즘

① 메시지 차분 M_a^{i-3}으로부터 $a_{48,i}$의 등식을 획득할 수 없는 $i = 0 \sim 5, 9, 19 \sim 22, 29 \sim 31$에 대하여, $k_{a,i}$의 값을 추측한다(총 14비트). 이 비트들은 이후 전수조사를 통하여 복구할 것이다.

② k_a의 다른 비트들은 최하위비트에서 최상위비트로 M_a^{i-3}을 이용하여 복구할 수 있다. 우선 $k_{a,6}$을 M_a^6을 이용하여 다음과 같이 복구한다.

- $k_{a,6}$ 복구 방법: $k_{a,5\sim0}$과 $h_{a,5\sim0}$의 값을 비교한다.

ⓐ 만약 $k_{a,5\sim0} > h_{a,5\sim0}$이면, $k_a + a_{48}$을 계산할 때 5번째 비트에서 6번째 비트로 올림이 존재한다.

ⓑ 만약 그렇지 않으면, $k_a + a_{48}$을 계산할 때 5번째 비트에서 6번째 비트로 올림이 존재하지 않는다.

ⓒ $a_{48,6} = 1$을 만족하며, $h_{a,6}$의 값, 그리고 5번째 비트에서 6번째 비트로 올림의 값을 알기 때문에, $k_{a,6}$의 값을 계산할 수 있다.

$$\begin{array}{lll} & \overset{known}{\overbrace{carry}} & \underbrace{unknown}_{} \\ a_{48,6} & & a_{48,5\sim0} \\ +\ k_{a,6} & & \underbrace{k_{a,5\sim0}}_{known} \\ \hline h_{a,6} & & \underbrace{h_{a,5\sim0}}_{known} \end{array}$$

③ 따라서 $i = 6 \sim 8, 10 \sim 18, 23 \sim 28$에 대하여, M_a^{i-3}

을 이용하여 최하위비트부터 최상위비트로 $k_{a,i}$를 단계적으로 복구할 수 있다.

3.3.4 공격 복잡도

3.3.3에서 제시한 키 복구 과정을 이용하면, k_a의 18비트, k_c의 17비트, k_d의 16비트를 복구할 수 있다. 따라서 총 51비트를 복구할 수 있으며, 나머지 77(128−51)비트는 복구할 수 없으므로 전수조사를 통하여 복구한다. 따라서 128비트 키를 모두 복구하기 위해서는 2^{77}의 MD4 연산이 필요하다. 데이터 수집 과정에서는 51비트의 키를 복구하기 위한 근사 충돌쌍을 수집하여야 한다. 각각의 M_a^i, M_c^i, M_d^i을 만족하는 근사 충돌쌍이 발생할 확률은 2^{-64}이므로 $51\times 2^{64}\le 2^{72}$의 질의량이 필요하다.

IV. 개선된 HMAC/NMAC-MD4 외부 키 복구 공격

본 절에서는 Wang, Ohta, Kunihiro 등이 제안한 HMAC/NMAC 외부 키 복구 과정을 향상시킨 새로운 외부 키 복구 공격 알고리즘을 소개하고 공격 복잡도를 계산한다.

4.1 향상된 HMAC/NMAC-MD4 키 복구 과정

향상된 HMAC/NMAC 키 복구 공격 중 온라인 과정(데이터 수집과정)은 Wang, Ohta, Kunihiro의 방법과 동일하므로 본 소절에서는 설명을 생략한다. 본 소절에서는 오프라인 과정(키 복구 과정)만을 설명한다. HMAC/NMAC의 외부 키 복구 과정을 설명하기에 앞서 몇 가지 표기법을 정의한다.

- $KeySet_a^i$: k_a의 0번째 비트부터 i번째 비트까지 $i+1$비트로 구성된 키 후보 집합으로 2^{i+1}개의 원소로 초기화된다. 그러나 하위 비트를 복구한 경우 키 후보 집합의 원소의 개수는 복구된 키 비트만큼 줄어든다.
- $DataSet_a^i$: $k_{a,i\sim 0}$을 복구하기 위해 필요한 데이터 집합으로 M_a^{i-3}을 만족하는 해쉬값 $h_{a,i\sim 0}$을 원소로 갖는다. $k_{a,i\sim 0}$을 복구하기 위해 필요한 데이터 집합 $DataSet_a^i$의 양은 다음 소절에서 자세히 제시한다.

k_a, k_c, k_d를 복구하는 방법은 모두 동일하며 본 소절에서는 단지 k_a를 복구하는 방법만을 설명한다. k_a를 복구하는 과정은 다음과 같다.

- 개선된 HMAC/NMAC 외부 키 $k_1=(k_a,k_b,k_c,k_d)$ 복구 알고리즘

① 가능한 모든 키 후보 $k_{a,i\sim 0}$을 키 후보 집합 $KeySet_a^i$에 저장한다.

② M_a^{i-3}을 만족하는 해쉬값 $h_{a,i\sim 0}$을 데이터 집합 $DataSet_a^i$에 저장한다.

③ 데이터 집합 $DataSet_a^i$에 저장된 $h_{a,i\sim 0}[j]$에 대하여 다음 과정을 반복한다.

ⓐ 키 후보 집합 $KeySet_a^i$에 속한 모든 원소 $k_{a,i\sim 0}[l]$에 대하여 다음 과정을 반복한다.

㉠ 만약 $k_{a,(i-1)\sim 0}[l] > h_{a,(i-1)\sim 0}[j]$이면, $k_a + a_{48}$을 계산할 때 5번째 비트에서 6번째 비트로 올림이 존재한다.

㉡ 만약 그렇지 않으면, $k_a + a_{48}$을 계산할 때 5번째 비트에서 6번째 비트로 올림이 존재하지 않는다.

㉢ $a_{48,i}=1$을 만족하며, $h_{a,i}$의 값, 그리고 5번째 비트에서 6번째 비트로 올림의 값을 알기 때문에, $k_{a,i}'$의 값을 계산할 수 있다.

$$\begin{array}{rcl} & \overbrace{carry}^{known} & \overbrace{unknown}^{} \\ a_{48,i} & & a_{48,(i-1)\sim 0} \\ +\ k_{a,i}' & & \overbrace{k_{a,(i-1)\sim 0}}^{known} \\ \hline h_{a,i} & & \underbrace{h_{a,(i-1)\sim 0}}_{known} \end{array}$$

㉣ $k_{a,i}=k_{a,i}'$이면 키 후보 집합 $KeySet_a^i$에 남겨두고, 그렇지 않으면 버린다.

④ 만약 키 후보 집합 $KeySet_a^i$의 원소가 한 개면 ⑤번을 실행하고, 그렇지 않으면 $j=j+1$하고 ③을 실행한다.

⑤ 따라서 $i=6\sim 8, 10\sim 18, 23\sim 28$에 대하여, ①을 실행한다. M_a^{i-3}을 만족하는 데이터 집합 $DataSet_a^i$을 이용하여 최하위부터 최상위비트로 $k_{a,i\sim 0}$를 복구할 수 있다. $i=6$일 때 $k_{a,6\sim 0}$을 복구할 수 있으며, $i=7$일 때는 이미 $k_{a,6\sim 0}$을 복구하였으므로 $k_{a,7}$만을 복구하면 된다. 같은 방법으로 키의 상위비트를 순차적으로 복구해 나간다.

〔표 5〕 k_a를 복구하기 위하여 필요한 데이터 양

$k_{a,i\sim 0}$:i	28	27	26	25	24	23	⋯	18	17	16	15	14	13	12	11	10	⋯	8	7	6	합계
$D(i)$	1	1	1	1	1	31	⋯	1	1	1	1	1	1	1	1	3	⋯	1	1	127	162

〔표 6〕 k_c를 복구하기 위하여 필요한 데이터 양

$k_{c,i\sim 0}$:i	31	30	29	⋯	25	24	23	⋯	10	9	8	⋯	5	4	3	2	1	0	합계
$D(i)$	1	1	15	⋯	1	1	$2^{13}-1$	⋯	1	1	7	⋯	1	1	1	1	1	1	8225

〔표 7〕 k_d를 복구하기 위하여 필요한 데이터 양

$k_{d,i\sim 0}$:i	29	28	27	26	25	24	23	22	21	⋯	17	16	15	⋯	6	5	4	3	2	1	합계
$D(i)$	1	1	1	1	1	1	1	1	15	⋯	1	1	511	⋯	1	1	1	1	1	3	544

4.2 공격 복잡도

본 소절에서는 HMAC/NMAC의 외부 키 $k_1=(k_a,k_b,k_c,k_d)$를 복구하기 위해 필요한 공격 복잡도를 계산한다. 우선 공격을 위해 필요한 데이터양을 계산하고, 필요한 데이터를 획득하기 위해 HMAC/NMAC 오라클에 질의하여야 하는 양을 계산한다. 그리고 HMAC/NMAC 오라클 질의에 의하여 획득한 데이터를 이용하여 HMAC/NMAC의 외부 키 k_1을 복구하기 위한 공격 복잡도를 계산한다.

4.2.1 온라인 질의량 (데이터 복잡도)

본 소절에서는 옳은 키 $k_{a,i\sim 0}$을 유일하게 결정하기 위해서 필요한 데이터 집합 $DataSet_a^i$의 크기를 계산한다. 만약 추측한 키 $k_{a,i\sim 0}$이 c비트라고 한다면 한 개의 데이터를 이용하여 $k_{a,i\sim 0}$ 키 후보 집합을 계산하면 정확하게 2^{c-1}개가 존재한다. 왜냐하면 c비트 덧셈에서 최상위 비트가 특정 충분조건($k_{a,i}=k_{a,i}'$)을 만족할 확률은 2^{-1}이기 때문이다. 남겨진 2^{c-1}개의 후보 키에 대하여 추가적으로 두 개의 데이터를 이용하여 c비트 키 후보 집합을 계산하면 확률적으로 2^{c-2}개가 존재한다. 왜냐하면, 덧셈 연산 시 $c-1$번째 키 비트에 의하여 올림이 발생하여 c번째 비트에 영향을 줄 확률이 2^{-1}이기 때문이다. 따라서 추가적으로 약 두 개의 데이터를 이용하면 후보 키를 2^{c-2}로 줄일 수 있다. 2^{c-2}개의 후보 키에 대하여 추가적으로 네 개의 데이터를 이용하여 c비트 키 후보 집합을 계산하면 확률적으로 2^{c-3}개가 존재한다. 왜냐하면, 덧셈 연산 시 $c-2$ 번째 키 비트에 의하여 올림이 발생하여 c번째 비트에 영향을 줄 확률이 2^{-2}이기 때문이다. 따라서 추가적으로 약 네 개의 데이터를 이용하면 후보 키를 2^{c-3}로 줄일 수 있다. 이와 같은 성질을 이용하여 공격자가 유일한 키를 결정하기 위한 데이터양을 계산할 수 있다.

$D(i)$를 키 $k_{a,i\sim 0}$를 유일하게 결정하기 위해서 필요한 데이터 집합 $DataSet_a^i$의 크기라고 한다면 $D(i)$는 다음과 같이 계산 할 수 있다.

$$D(i)=1+2^1+2^2+\cdots+2^i$$
$$=\frac{1\times(2^{i+1}-1)}{2-1}=2^{i+1}-1.$$

따라서 $i=6$일 경우 필요한 데이터양은 2^7-1이다.

그리고 키 $k_{a,i\sim 0}$을 유일하게 결정하기 위해서 필요한 데이터를 얻기 위해서 공격자가 HMAC/ NMAC 오라클에게 질의하여야 할 메시지쌍의 개수를 $QN(i)$라 할 때 $QN(i)$는 다음과 같이 계산 할 수 있다.

$$QN(i)=D(i)\times 2^{64}.$$

따라서 $i=6$일 경우 필요한 데이터양은 $(2^7-1)\times 2^{64}\approx 2^{71}$이다. 〔표 5〕는 k_a를 복구하기 위해 필요한 데이터양을 나타낸다. k_a의 최하위부터 순차적으로 키를 복구하기 때문에 상위비트 복구하기 위해서 추가적으로 하위 비트의 키를 복구하지 않아도 된다. 예를 들어 $i=7$일 때, $k_{a,6\sim 0}$은 복구되었으므로 $k_{a,7}$만 복구하면 된다. ③-ⓐ-㉣에서 $k_{a,7}=k_{a,7}'$ 조건을 만족할 확

〔표 8〕 k_a를 복구하기 위한 공격복잡도

$k_{a,i\sim 0}$:i	28	27	26	25	24	23	…	18	17	16	15	14	13	12	11	10	…	8	7	6	합계
$O(\cdot)$	2	2	2	2	2	$5\cdot 2^5$	…	2	2	2	2	2	2	2	2	$2\cdot 2^2$	…	2	2	$7\cdot 2^7$	1086

〔표 9〕 k_c를 복구하기 위한 공격복잡도

$k_{c,i\sim 0}$:i	31	30	29	…	25	24	23	…	10	9	8	…	5	4	3	2	1	0	합계
$O(\cdot)$	2	2	$4\cdot 2^4$	…	2	2	$13\cdot 2^1$	…	2	2	$3\cdot 2^3$	…	2	2	2	2	2	2	106608

〔표 10〕 k_d를 복구하기 위한 공격복잡도

$k_{d,i\sim 0}$:i	29	28	27	26	25	24	23	22	21	…	17	16	15	…	6	5	4	3	2	1	합계
$O(\cdot)$	2	2	2	2	2	2	2	2	$4\cdot 2^4$	…	2	2	$9\cdot 2^9$	…	2	2	2	2	2	$2\cdot 2^2$	4710

률은 2^{-1}이므로 $k_{a,7}$을 복구하기 위해서 필요한 데이터 양은 1개이다.

〔표 5, 6, 7〕에 의하여 k_a, k_c, k_d를 복구하기 위해 필요한 질의량은 다음과 같다.

$(162+8225+544)\times 2^{64} = 8931\times 2^{64} \approx 2^{77.1246}$.

4.2.2 공격 복잡도

k_a, k_c, k_d를 복구하기 위한 공격 복잡도 계산 방법은 모두 동일하다. 따라서 본 소절에서는 k_a를 복구하기 위해 필요한 공격 복잡도 계산 방법만을 설명하고, k_c, k_d를 복구하기 위해 필요한 공격 복잡도는 결과만을 제시한다. 앞서 공격 방법에서 설명하였듯이 k_a의 하위비트부터 순차적으로 키 비트를 복구해 가기 때문에 순차적으로 공격 복잡도를 계산한다. $O(i)$는 $k_{a,i\sim 0}$을 복구하기 위해 필요한 공격 복잡도이다.

- $k_{a,6\sim 0}$을 복구하기 위한 공격 복잡도 : *DataSet*[6]에 포함된 $h_{a,6\sim 0}[1]$를 이용하여 2^7개의 후보 키에 대하여 ③-ⓐ를 수행하면 후보 키는 2^6개로 줄어들고, 두 개의 데이터 $h_{a,6\sim 0}[2]$, $h_{a,6\sim 0}[3]$을 이용하여 2^6개의 후보에 대하여 ③-ⓐ를 수행하면 후보 키는 2^5개로 줄어든다. 이와 같은 과정을 반복하므로, $k_{a,6\sim 0}$을 복구하기 위한 공격 복잡도는 다음과 같이 계산된다.

$$O(6) = 2^7\times 1 + 2^6\times 2^1 + 2^5\times 2^2 + 2^4\times 2^3 + 2^3\times 2^4 + 2^2\times 2^5 + 2^1\times 2^6 = 7\cdot 2^7.$$

- $k_{a,7}$, $k_{a,8}$을 복구하기 위한 공격 복잡도: 이미 $k_{a,6\sim 0}$을 복구하였으므로, $k_{a,7}$을 복구하기 위해서는 $h_{a,7}[1]$을 이용하여 2^1개의 후보 키에 대하여 ③-ⓐ를 수행하면 $k_{a,7}$을 복구할 수 있다. 마찬가지로 $k_{a,7\sim 0}$을 복구하였으므로, $k_{a,8}$을 복구하기 위해서는 $h_{a,8}[1]$을 이용하여 2^1개의 후보 키에 대하여 ③-ⓐ를 수행하면 $k_{a,8}$을 복구할 수 있다.
- $k_{a,10\sim 9}$를 복구하기 위한 공격 복잡도: 이미 $k_{a,8\sim 0}$을 복구하였으므로, $k_{a,10\sim 9}$를 복구하기 위해서는 $h_{a,10}[\cdot]$을 이용하여 2^2개의 후보 키에 대하여 ③, ④를 수행하면 $k_{a,10\sim 9}$를 복구할 수 있다. $k_{a,10\sim 9}$를 복구하기 위한 공격 복잡도는 다음과 같이 계산된다.

$$O(10) = 2^2\times 1 + 2^1\times 2^2 = 2\cdot 2^2.$$

위에 설명된 방법에 의하여 키 k_a를 복구할 수 있으며, k_a를 복구하는데 필요한 공격 복잡도는 〔표 8〕와 같다. 또한 키 k_c, k_d를 복구하기 위해 필요한 공격 복잡도는 〔표 9, 10〕과 같다.

본 논문에서 제시한 향상된 HMAC/NMAC 외부 키 복구 방법을 이용하면 $1086+106608+4710 = 112404 \approx 2^{16.7783}$의 공격 복잡도로 k_a는 29비트, k_c는 32비트, k_d는 30비트를 복구 할 수 있다. 따라서 그 밖의 키 37(128−91)비트는 전수조사를 통하여 복구할 수 있다. 따라서 128비트 키를 모두 복구하기 위해서는 2^{37}의 MD4 연산이 필요하다.

V. 결 론

본 논문에서는 EUROCRYPT'08에서 Lei Wang 등이 소개한 HMAC/NMAC-MD4 키 복구 공격을 향상시킨 분석 방법을 소개하였다. Lei Wang 등은 근사 충돌쌍을 발생시키는 새로운 차분 경로를 제시하고, 외부 키를 복구하기 위하여 근사 충돌쌍 특성을 이용하였다. 하지만 Lei Wang의 HMAC/NMAC- MD4 키 복구 공격의 경우, 외부 키 복구 알고리즘에서 공격자가 복구하고자 하는 외부 키 128비트 중 특성식이 존재하는 51비트만을 특성식을 통하여 복구하고, 그 이외의 77비트는 전수조사를 통하여 복구하기 때문에 매우 비효율적이다. 본 논문에서는 HMAC/NMAC-MD4의 외부 키 128비트 중 특성식이 존재하는 비트의 하위 비트를 특성식을 기준으로 분할하여 순차적으로 외부 키 128비트 복구하기 때문에 Lei Wang의 HMAC/ NMAC-MD4 외부 키 복구 공격을 향상시킬 수 있었다. 본 논문에서 제시한 분할-정복(divide and conquer)을 이용한 새로운 외부 키 복구 공격 알고리즘은 HMAC/NMAC 오라클에 대한 질의량 $2^{77.1246}$과 MD4 계산량 2^{37}으로 외부 키를 복구할 수 있다. [표 1]에서 제시한 바와 같이 본 논문의 분석 결과는 기 제안된 Lei Wang의 분석 방법의 MD4 계산량은 2^{77}에서 2^{37}으로 대폭 감소시켰다.

참 고 문 헌

[1] G. Leurent, "Message Freedom in MD4 and MD5 Collisions: Application to APOP," Advances in Cryptology - FSE 2007, LNCS 4593, pp. 309-328, 2007.

[2] L. Wang, K. Ohta, and N. Kunihiro, "New Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5," Advances in Cryptology - EUROCRYPT 2008, LNCS 4965, pp. 237-253, 2008.

[3] M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication," Advances in Cryptology - CRYPTO 1996, LNCS 1109, pp. 1-15, 1996.

[4] P.A. Fouque, G. Leurent, and P.Q. Nguyen, "Full Key-Recovery Attacks on HMAC/NMAC- MD4 and NMAC-MD5," Advances in Cryptology - CRYPTO 2007, LNCS 4622, pp. 13-30, 2007.

[5] S. Contini and Y.L. Yin, "Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions," Advances in Cryptology - ASIACRYPT 2006, LNCS 4284, pp. 37-53, 2006.

[6] X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu, "Cryptanalysis of the Hash Functions MD4 and RIPEMD," Advances in Cryptology - EUROCRYPT 2005, LNCS 3494, pp. 1-18, 2005.

[7] X. Wang and H. Yu, "How to Break MD5 and Other Hash Functions," Advances in Cryptology - EUROCRYPT 2005, LNCS 3494, pp. 19-35, 2005.

[8] X. Wang, H. Yu, and Y.L. Yin, "Efficient Collision Search Attacks on SHA-0," Advances in Cryptology - CRYPTO 2005, LNCS 3621, pp. 1-16, 2005.

[9] X. Wang, Y.L. Yin, and H. Yu, "Finding Collisions in the Full SHA-1," Advances in Cryptology - CRYPTO 2005, LNCS 3621, pp. 17-36, 2005.

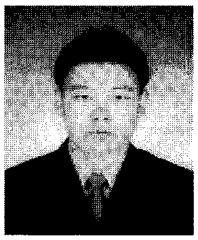
[10] Y. Sasaki, G. Yamamoto, and K. Aoki, "Practical Password Recovery on an MD5 Challenge and Response," IACR ePrint archive 2007-101, Mar. 2007.

[11] Y. Sasaki, L. Wang, K. Ohta, and N. Kunihiro, "Password Recovery on Challenge and Response: Impossible Differential Attack on Hash Function," Advances in Cryptology - AFRICACRYPT 2008, LNCS 5023, pp. 290-307, 2008.

〔부록 A〕 근사 충돌쌍을 발생시키는 차분경로 DP와 충분조건 SC

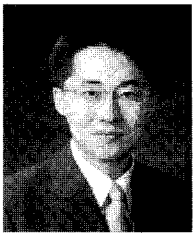
단계 i	순환이동값 s_i	$\triangle m_{i-1}$	$\triangle b_i$	
			덧셈차분 $(=\triangle b_i = b_i' - b_i)$	충분조건
1	3			
2	7			
3	11			$b_{3,22} = b_{2,22}$
4	19	2^3	2^{22}	$b_{4,22} = 0$
5	3			$b_{5,22} = 0$
6	7			$b_{6,22} = 1$
7	11			$b_{7,9} = b_{6,9}$
8	19		2^9	$b_{8,9} = 0$
9	3			$b_{9,9} = 0$
10	7			$b_{10,9} = 1$
11	11			$b_{11,28} = b_{10,28}$
12	19		2^{28}	$b_{12,28} = 0$
13	3			$b_{13,28} = 0$
14	7			$b_{14,28} = 1$
15	11			$b_{15,15} = b_{14,15}$
16	19		2^{15}	$b_{16,15} = 0$
17	3			$b_{17,15} = b_{15,15}$
18	5			$b_{18,15} = b_{17,15}$
19	9			$b_{19,28} = b_{18,28}, b_{19,29} \neq b_{18,29}, b_{19,30} = b_{18,30}$
20	13		$2^{28}(28 \sim 30)$	$b_{20,0} = b_{19,0}, b_{20,28\sim29} = 1, b_{20,30} = 0$
21	3		-2^0	$b_{21,0} = 1, b_{21,28\sim30} = b_{19,28\sim30}$
22	5			$b_{22,0} = b_{20,0}, b_{22,28\sim30} = b_{21,28\sim30}$
23	9			$b_{23,0} = b_{22,0}, b_{23,9} = b_{22,9}$
24	13		2^9	$b_{24,3\sim8} = b_{23,3\sim8}, b_{24,9} = 0$
25	3		$-2^3(3 \sim 9)$	$b_{25,3\sim8} = 0, b_{24,9} = 1$
26	5			$b_{26,3\sim8} = b_{24,3\sim8}$
27	9			$b_{27,3\sim8} = b_{26,3\sim8}, b_{27,9} \neq b_{26,9}$
28	13			
29	3	2^3		
30	5			

〈著者紹介〉

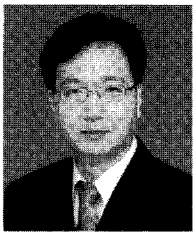


강 진 건 (Jinkeon Kang) 학생회원
2007년 8월: 고려대학교 산업시스템정보공학과 학사
2007년 9월~현재: 고려대학교 정보경영공학전문대학원 석박사통합과정
〈관심분야〉 해쉬 함수의 분석 및 설계

이 제 상 (Jesang Lee) 학생회원
2003년 2월: 고려대학교 수학과 학사
2006년 8월: 고려대학교 정보보호대학원 석사
2006년 9월~현재: 고려대학교 정보경영공학전문대학원 박사과정
〈관심분야〉 대칭키 암호의 분석 및 설계



성 재 철 (Jaechul Sung) 종신회원
1997년 8월: 고려대학교 수학과 학사
1999년 8월: 고려대학교 수학과 석사
2002년 8월: 고려대학교 수학과 박사
2002년 8월~2004년 1월: 한국정보보호진흥원 선임연구원
2004년 2월~현재: 서울시립대학교 수학과 조교수
〈관심분야〉 암호 알고리즘 설계 및 분석



홍 석 희 (Seokhie Hong) 종신회원
1995년 2월: 고려대학교 수학과 학사
1997년 2월: 고려대학교 수학과 석사
2001년 2월: 고려대학교 수학과 박사
1999년 8월~2004년 2월: (주)시큐리티 테크놀로지스 선임연구원
2004년 4월~2005년 2월: K.U.Leuven 박사후연구원
2005년 3월~2008년 8월: 고려대학교 정보경영공학전문대학원 조교수
2008년 9월~현재: 고려대학교 정보경영공학전문대학원 부교수
〈관심분야〉 암호 알고리즘 설계 및 분석, 컴퓨터 포렌식

류 희 수 (Heuisu Ryu) 정회원
1989년 2월: 고려대학교 수학과 이학사
1992년 2월: 고려대학교 수학과 이학석사
1999년 5월: 미국 Johns Hopkins 수학과 Ph.D
2002년 9월~현재: 경인교육대학교 수학교육과 조교수
〈관심분야〉 암호이론 및 알고리즘, 정보보호교육, 정수론, 수학교육