通信情報保護學會論文誌
第 8 卷 第 4 號. 1998. 12

셀룰러 오토마타에 기반한 안전한 해쉬함수

신 상 욱*, 윤 재 우**, 이 경 현*

A secure hash function based on cellular automata

Sang Uk Shin, Yoon, Jae Woo, Kyung Hyune Rhee

요 약

해쉬함수는 데이터 무결성과 인증을 달성하기 위해 현대 암호학에서 빈번하게 사용되는 중요한 도구이다. 본 논문에서는 셀룰러 오토마타에 기반한 새로운 해쉬함수를 제안한다. 제안된 해쉬함수의 안전성은 셀룰러 오토마타 이론에 기초하여 분석된다. 그러한 분석은 제안된 해쉬함수가 알려진 공격에 대해 안전하다는 것을 나타내며, 제안된 해쉬함수의 또다른 중요한 특징은 매우 빠르고 하드웨어 구현에 적합하다는 것이다.

Abstract

A hash Hash function is an important tool which is frequently used for achieving message authentication and integrity. In this paper, we propose a new hash function based on cellular automata and analyze its security by using the theory of the cellular automata. The analysis indicates that the proposed hash function is secure against known attacks. Also another feature of the proposed scheme is that it is fast and suitable for hardware implementation.

Ⅰ. 서 론

해쉬함수는 임의 길이의 비트 스트링을 입력으로 받아 고정된 짧은 길이(주로 128, 160 비트)의 비트 스트링을 출력하는 함수로, 해쉬함수는 현대 암호학에서 중요한 역할을 한다. 해쉬함수의 기본 원리는 해쉬값이 입력 스트링의 압축된 대표 이미지로 작용하고, 그 스트링을 유일하게 식별할 수 있는 것으로 사용될

* 부경대학교
** 한국전자통신연구원

수 있다는 것이다. 해쉬함수는 하나의 입력 파라메터, 메시지를 가지는 unkeyed 해쉬함수와 두 개의 다른 입력 파라메터, 메시지와 비밀키를 가지는 keyed 해쉬함수의 두 분류로 나누어질 수 있다[9]. 본 논문에서는 unkeyed 해쉬함수에 관해서만 다룬다. 해쉬함수는 중요 정보의 무결성(integrity) 확인과 메시지 인증 코드(Message Authentication Code : MAC)의 구성, 디지털 서명(digital signature)의 효율성 증대 등의 목적으로 사용된다.

본 논문에서는 새로운 해쉬함수를 제안하고 분석한다. 제안된 해쉬함수는 현재까지 제안된 해쉬함수와는 매우 다른 접근법으로 프로그램 가능한 셀룰러 오토마타(programmable cellular automata)에 기반한 구성을 가진다. 셀룰러 오토마타에 기반한 해쉬함수의 이점은 매우 빠르고 하드웨어 구현에 적합하다는 것과 그 안전성이 셀룰러 오토마타 이론에서의 연구 결과를 이용하여 분석될 수 있다는 것이다.

2절에서는 해쉬함수에 관한 정의와 일반적 모델에 관해, 그리고 3절에서는 셀룰러 오토마타에 관한 기본 이론을 기술한다. 4절에서는 새로운 해쉬함수가 제안되고, 5절에서 그 안전성을 분석한다. 마지막 6절에서는 결론을 맺는다.

II. 해쉬함수의 정의와 일반적 모델

해쉬함수(hash function), 좀더 정확하게 암호학적 해쉬함수(cryptographic hash function)는 임의의 유한 길이 비트 스트링을 고정된 길이의 스트링으로 사상시키는 함수이다. 이 출력은 흔히 해쉬값(hash value), 메시지 다이제스트(message digest) 또는 fingerprint 등으로 불린다. 함수 h와 입력 x가 주어지면, $h(x)$를 계산하는 것은 쉬워야 한다. 일방향 해쉬함수는 다음 성질을 만족해야 한다[9].

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

암호학적으로 유용한 해쉬함수는 다음 성질을 추가로 만족해야 한다[9].

- *collision resistance* : 충돌(같은 결과로 해쉬하는 두 개의 다른 입력)을 발견하는 것이 계산상 수행 불가능하다. 즉, $h(x)=h(x')$을 만족하는 임의의 서로 다른 두 입력 쌍 x

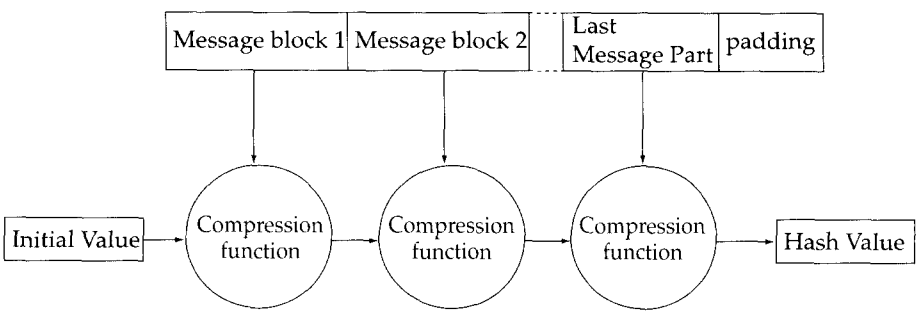


그림1. 반복적인 해쉬함수의 일반적인 구조
Fig. 1. The use of a compression function in an iterative hash function

와 x'을 발견하는 것이 수행 불가능하다.

거의 대부분의 해쉬함수의 처리 과정은 입력을 연속적인 고정된 블록들로 나누어 처리함으로서 임의 길이 입력을 해쉬하는 반복적인 처리 과정이다. 먼저 입력 X는 블록 길이의 배수가 되도록 padding되고 t개의 블록 X_i에서 X_i로 나누어진다. 해쉬함수 h는 다음처럼 기술된다.

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

여기서 f는 압축 함수(compress function)이고 g는 출력 함수, H_i는 단계 i-1과 단계 i 사이의 연쇄 변수(chaining variable)이고, IV는 초기값(Initial Value)이다. 압축 함수를 사용한 반복적인 해쉬함수의 일반적인 구조가 그림 1에 주어졌다.

해쉬값의 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 해쉬값으로 출력된다.

해쉬함수는 내부 압축 함수로 어떤 구조를 사용하느냐에 따라 3가지로 분류된다[9].

① 블록 암호(block ciphers)에 기반한 해쉬함수
② 모듈러 연산(modular arithmetic)에 기반한 해쉬함수
③ 전용해쉬함수(dedicated hash functions)

전용해쉬함수는 빠른 처리 속도를 가지고 다른 시스템 서브 요소에 무관하도록 해쉉을 위해 특별히 설계된 함수들이다. 현재까지 제안된 전용해쉬함수는 대부분 1990년에 Rivest에 의해 설계된 MD4[17]에 기반한 구조를 가진다. 현재 널리 사용되는 MD 계열 해쉬함수로 MD5[18], SHA-1[11], RIPEMD-160[3], HAVAL[20] 등이 있다.

특정 해쉬함수가 주어지면, 안전한 해쉬함수의 입증을 위해 해쉬함수를 공격하는 복잡도에 관한 하한을 증명할 수 있는 것이 바람직하지만 실제 그러한 방법은 거의 알려져 있지 않고 대부분의 경우에 적용 가능한 알려진 공격의 복잡도가 해쉬함수의 안전성으로 고려되어진다.

해쉬값이 균등한 확률 변수라고 가정하면, 다음은 잘 알려진 사실이다[9].

- n비트 해쉬함수 h에 대해, 2^n 연산으로 preimage와 second preimage를 발견하기 위한 추측 공격(guessing attack)을 기대할 수 있다.
- 메시지를 선택할 수 있는 공격자에 대해, 생일 공격(birthday attack)은 약 $2^{n/2}$ 연산으로 $hash(M) = hash(M')$인 충돌 메시지 쌍 M, M'을 발견할 수 있다.

n 비트 해쉬함수가 다음 두 가지를 만족한다면, 이상적인 안전성을 가진다.

(a) 해쉬값이 주어지면, preimage와 second preimage 발견은 약 2^n 연산을 요구한다.
(b) 충돌 발견은 약 $2^n/2$연산을 요구한다.

Merkle[10]과 Damgård[2]의 이론에 따라 입력 스트링의 끝에 그 길이를 포함하는 블록을 추가하는 것을 MD 강화(MD-streng thening)이라 한다. 다음 정리에 의해 해쉬함수 h의 안전성을 압축 함수 f와 출력 함수 g에 관련시키는 것이 가능하다.

정리 1 [7]. h를 MD-strengthening을 가진 반복적인 해쉬함수라 하자. 그러면 h에 관한 preimage와 충돌 공격은 f와 g에 관해 대응하는 공격과 같은 복잡도를 가진다.

정리 1은 해쉬함수 h에 관한 하한을 제공한다.

Davies-Meyer 압축 함수에 기반한 해쉬함수는 다음과 같다[7][16]

$$h(M_i, H_{i-1}) = E_{M_i}(H_{i-1}) \oplus H_{i-1} \quad (2)$$

여기서 $E_K()$는 키 K에 의해 제어되는 블록 암호이다.

이러한 Davies-Meyer 형태의 해쉬함수의 안전성은 기반이 되는 블록 암호 $E_K()$와 같은 안전성을 가진다. Knudsen[7] 등의 가정과 같이 다음을 가정할 수 있다.

가정 1. f를 Davies-Meyer 함수라 하고 사용된 암호학적 변환이 안전하다고 가정한다. 그러면, f에 대해 충돌 발견은 약 $2^n/2$ 연산을 요구하고 preimage 발견은 약 2^n연산을 요구한다.

위의 가정은 해쉬함수 설계에서 주요한 문제가 안전한 압축함수와 좋은 출력함수의 설계로 줄어들 수 있다는 것을 의미한다.

Ⅲ. 셀룰러 오토마타(Cellular Automata : CA)

셀룰러 오토마타(Cellular Automata : CA)는 선형적으로 연결된 L 셀의 배열과 q변수를 가진 부울 함수 $f(x)$로 구성된다. 각 셀은 0 또는 1의 값을 가진다. 셀 x_i의 값은 이산 시간 단계에서 $x_i = f(x)$, $i=1, 2, \cdots, L$로 병렬적으로 갱신된다. 파라메터 q는 보통 $q=2r+1$의 홀수 값이다. r은 $f(x)$의 반경(radius)이라 불린다. i번째 셀의 새로운 값은 i번째 셀 값과 i번째 셀의 r개의 왼쪽과 오른쪽 이웃 셀의 값을 사용하여 계산된다.

L셀이 있기 때문에 2^L개 가능한 상태 벡터가 존재한다. S_k를 시간 단계 k에서 상태 벡터라 하자. 초기 상태 벡터 S_0로부터 시작하여 CA는 시간 단계 $k=1, 2, \cdots$에서 S_1, S_2, S_3,$\cdots$로 이동한다. 상태 천이는 사이클을 형성한다. 즉 $S_k = S_k + P$이다. 여기서 P는 주기로서 초기 상태, 갱신 함수, 셀의 수의 함수이다.

예를 들어 $q=3$인 CA에 대해, 시간 t에서 i번째 셀의 상태 천이는 다음처럼 표현된다.

$$x_i(t+1) = f\{x_{i-1}(t), x_i(t), x_{i+1}(t)\} \quad (3)$$

이때 f는 CA에 관련된 조합 로직으로 불린다. 각 조합 로직은 다음 상태로 천이를 위한 갱신 규칙을 나타낸다.

셀의 다음 상태 함수가 진리표의 형태로 표현된다면, 진리표에서 출력 열의 10진 표현은 CA 규칙 번호로 불린다. 비선형 규칙, Rule 30은 Wolfram[19]에 의해 제안되었고, 다음의 조합 로직을 가진다.

$$x_i(t+1) = x_{i-1}(t) \text{ XOR } \{x_i(t) \text{ OR } x_{i+1}(t)\} \quad (4)$$

유사하게 Rule 150, Rule 102, Rule 60, Rule 204는 다음과 같다[15].

Rule 150 : $x_i(t+1) = x_{i-1}(t) \oplus x_i(t) \oplus x_{i+1}(t)$ (5)
Rule 102 : $x_i(t+1) = x_i(t) \oplus x_{i+1}(t)$ (6)
Rule 60 : $x_i(t+1) = x_{i-1}(t) \oplus x_i(t)$ (7)
Rule 204 : $x_i(t+1) = x_i(t)$ (8)

CA에서 모든 셀이 같은 규칙을 사용하면, 동형(uniform) CA라 하고, 그렇지 않으면 혼합(hybrid) CA라 한다. 경계 조건으로 양쪽 끝 셀에 로직'0'가 연결되는 *null* 과 양쪽 끝 셀이 연결되는 *periodic* 이 있다.

CA에서 매우 중요한 형태는 GF(2) 상에서 선형 CA 또는 additive CA이다. 다음 상태 천이 규칙이 XOR 또는 XNOR 연산만을 사용한다면, 그러한 CA를 additive CA라 한다. 모든 additive CA는 GF(2) 상의 천이 행렬(transition matrix)에 의해 유일하게 표현되고, 모든 천이 행렬은 특성 다항식(characteristic polynomial)을 가진다[15].

XOR 연산만을 가진 L셀 additive CA는 $L \times L$ 부울 행렬인 T에 의해 표현되는 선형 연산자에 의해 특성화된다. T의 i번째 행은 i번째 셀의 이웃 의존을 나타낸다. CA의 다음 상태는 열 벡터로 표현되는 현재 상태에 이 선형 연

산자를 적용함으로서 생성된다. 연산은 보통의 행렬 곱이지만, 관계되는 덧셈은 mod 2 덧셈이다. $x(t)$를 시간 t에서 CA의 현재 상태를 나타내는 열 벡터라 하자. 그러면 CA의 다음 상태 벡터는 다음 식에 의해 주어진다.

$$x(t+1)=T\times x(t) \qquad (9)$$

CA의 특성 다항식이 원시 다항식(primitive polynomial)이면, 최대 주기 CA(maximal-period CA)라 한다. 그러한 L셀 CA는 모두 0인 상태를 제외하고 연속적인 사이클에서 2^L의 모든 상태를 생성한다.

Rule 90과 Rule 150을 고려하면, 그들의 이웃 의존이 한 지점만 다르다는 것을 알 수 있다. 그러므로 셀 당 하나의 제어 라인을 사용함으로서, 다른 시간 단계에서 같은 셀에 Rule 90과 Rule 150 모두를 적용할 수 있다. 그로 인해 L셀 CA 구조가 2^L CA 구성을 구현하기 위해 사용될 수 있다. 같은 구조에서 다른 CA 셀 갱신 규칙을 구현하는 것은 적절한 스위치를 제어하기 위한 제어 로직과 ROM에 저장된 제어 프로그램을 사용함으로서 달성할 수 있다. 그러한 구조를 프로그램 가능한 CA(Programmable CA : PCA)라 한다.

CA는 해쉬함수 뿐만 아니라 블록 암호와 스트림 암호 모두에 사용될 수 있다. CA의 첫 번째 암호학적 응용은 Wolfram[16]에 의해 제안되었고, Nandi[15] 등은 CA에 기반한 블록과 스트림 암호 두 가지를 제안하였다. 또한 CA의 가운데 셀에 의해 생성된 비트 순열에 기반하여 CA 초기 상태를 재구성하는 방법이 Meier[8] 등에 의해 제안되었고, Koc[6] 등은 주어진 상태 벡터의 전 상태를 계산하는 inversion 알고리즘을 제안하였다. Mihaljevic은 Nandi에 의해 제안된 두 가지 키스트림 생성기의 안전성을 분석하고[11][12], 개선된 키 스트림 생성기를 제안하였다[13].

해쉬함수에 CA의 첫 번째 응용은 Damgård[2]에 의해 제안되었고, Daemen[17] 등은 Damgård 기법의 취약성을 분석하고 새로운 CA 기반 해쉬함수를 제안하였다. 또한 Hirose[5] 등은 2차원 CA에 기반한 해쉬함수를 제안하였고, Mihaljevic[14]은 압축 함수로 비선형 함수와 PCA의 결합 형태를 사용하고 출력 함수로 키 스트림 생성기를 적용한 해쉬함수를 제안하였다.

Ⅳ. 셀룰러 오토마타에 기반한 새로운 해쉬함수

이 절에서는 새로운 해쉬함수를 제안한다. 제안된 해쉬함수는 반복적인 해쉬함수에 대한 일반적인 모델을 따르고 Davies-Meyer 형태를 사용한다.

$$h(M_i, H_{i-1})=f_{M_i}(H_{i-1}) \oplus H_{i-1} \qquad (10)$$

이 형태는 압축 함수와 출력 함수가 안전하다면, 안전한 해쉬함수 구성을 함축한다. 제안된 해쉬함수는 압축 함수 f와 출력 함수 g가 CA에 기반한 형태이다. 제안된 해쉬함수는 매우 빠르고, 안전성 분석에 CA 이론을 적용할 수 있으며, 사용된 이론과 구조로 인해 preimage와 충돌 저항을 제공한다.

본 논문에서는 다음의 표기를 사용한다.

- n : 해쉬함수의 출력 길이 ($n=160$)
- l: n/l이 정수가 되도록 선택된 정수 ($l=8$)
- $\phi_k()$, $k=0, 1, \cdots 4$: 5개의 l 차원 벡터를 하나의 l차원 벡터로 사상시키는 비선형 부울 함수로, 사용된 비선형 부울 함수는 암호학적으로 강한 성질을 만족하는 것을 선택한다[20].
  - 0-1 balanced
  - high nonlinearity
  - satisfy SAC
  - pairwise linearly non-equivalent

ϕ_0(A,B,C,D,E)
=(A&E)^(B&C)^((B^C)&D)
ϕ_1(A,B,C,D,E)
=A^(B&(A^D))^(((A&D)^C)&E)
ϕ_2(A,B,C,D,E)
=A(C&D&E)^((A&C)|(B&D))
ϕ_3(A,B,C,D,E)
=B^((D&E)|(A&C))
ϕ_4(A,B,C,D,E)
=D^E^(((D&E)^A)& ¬ (B&C))

- CA() : 최대 주기를 가지는 CA
- PCA_{XY}() : 다음처럼 이진 벡터 X와 Y에 의해 제어되는 PCA

• X의 i번째 비트가 1 이고, Y의 i번째 비트가 1 이면, PCA의 i번째 셀에 rule 150을 적용
• X의 i번째 비트가 1 이고, Y의 i번째 비트가 0 이면, PCA의 i번째 셀에 rule 102를 적용
• X의 i번째 비트가 0 이고, Y의 i번째 비트가 1 이면, PCA의 i번째 셀에 rule 60을 적용
• X의 i번째 비트가 0 이고, Y의 i번째 비트가 0 이면, PCA의 i번째 셀에 rule 204를 적용

- M_i : 입력 메시지에서 $4n$비트 길이의 i번째 블록
- H_i : i번째 반복 계산 후의 n비트 연쇄변수

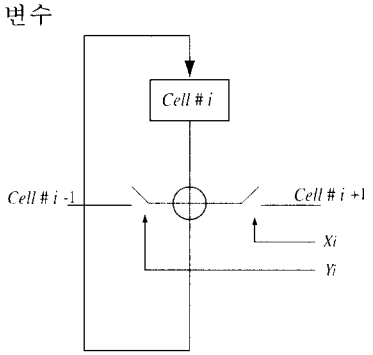


그림 2. PCAXY 의 블록도
Fig. 2 The block diagram of PCAXY

4.1 메시지 padding

제안된 해쉬함수는 가변적인 출력을 가진다. 메시지 padding 과정에서 출력 길이 L을 메시지 끝에 추가한다는 점만 제외하면 기존의 해쉬함수에서의 padding 과정과 동일하다. 2 bytes의 출력길이(최대 65536 bits의 출력) L을 메시지 padding 과정에서 8 bytes의 원래 메시지 길이 다음에 추가한다.

4.2 압축 함수 f()

padding 과정을 거친 후 입력 메시지 M을 $4n$비트의 m개 블록 M_1, M_2, …,M_m으로 분할한다. 압축함수 f는 입력으로 $4n$비트의 메시지 블록 M_i와 n 비트의 연쇄변수 H_{i-1}를 받아들여 n비트의 연쇄변수 Hi를 출력한다(n=160).

제안된 해쉬함수에서 사용하는 압축 함수는 다음과 같은 형태이다.

$$f(M_i, H_{i-1}) = \mathrm{PCA}_{XY}(Z) \oplus H_{i-1}$$

먼저 입력 메시지 블록 M_i를 l비트의 $\frac{4n}{l}$개 서브 블록, $M_{i,1}$,…,$M_{i,\frac{4n}{l}}$으로 분할한다. 마찬가지로 H_{i-1}을 l비트의 $\frac{n}{l}$개 서브 블록, $H_{i-1,1}$,…,$H_{i-1,\frac{n}{l}}$으로 분할한다.

압축 함수의 두 입력 M_i와 H_{i-1}을 사용하여 각각 n비트의 X, Y, Z를 다음처럼 계산한다.

① n비트 X를 다음처럼 계산한다.

$$X_k = \phi_{k \bmod K1}(M_{i,k}, M_{i,n/l+k}, H_{i-1,k}, H_{i-1,(k+n/2l) \bmod n/l}, M_{i,2n/l+k}) + C_{k \bmod K2} + M_{i,3n/l+k}$$
$$k = 0,1,2,\cdots,n/l$$

- ϕ_k ()는 $K1$개 비선형 부울 함수로 5개의 l비트 벡터를 하나의 l비트 벡터로 사상시킨다. ($K1$=5를 사용)
- C_k는 $K2$개의 고정된 상수이다.

② n비트 Y를 계산

$$Y = \mathrm{CA}(X)$$

③ X,Y,H_{i-1}를 PHT(Pseudo-Hadamard

Transform)에 적용

- n비트 X, Y, H_{i-1}을 X_1, X_2,…,$X_{n/8}$, Y_1,Y_2,…,$Y_{n/8}$, $H_{i-1,1}$, $H_{i-1,2}$,…,$H_{i-1,n/8}$ 으로 분할 (X_i, Y_i, H_i는 각각 8 bit)
- PHT(X_j, H_j) = ($2X_j+H_{i-1,j}$, $X_j+H_{i-1,j}$), j= 1,2,…, $\frac{n}{8\cdot 2}$
- +는 mod 256 덧셈이다.
- PHT($H_{i-1,j}$, Y_j), j= $\frac{n}{8\cdot 2}+1$, $\frac{n}{8\cdot 2}+2$,…, $\frac{n}{8}$
- PHT(X_j,X_k) j=1, 2,…, $\frac{n}{8\cdot 2}$, $k=\frac{n}{8\cdot 2}+1$, $\frac{n}{8\cdot 2}+2$,…, $\frac{n}{8}$
- PHT(Y_j, Y_k), j=1,2,…, $\frac{n}{8\cdot 2}$, $k=\frac{n}{8\cdot 2}+1$, $\frac{n}{8\cdot 2}+2$,…, $\frac{n}{8}$

④ n비트 벡터 V를 계산

$$V_k = \phi_{k \bmod K1}(X_k, M_{i,k+2n/l}, H_{i-1,k}, M_{i,n/l+k}, M_{i,k}) + M_{i,k+3n/l} + Y_k$$
$$k=0,1,2,\cdots,n/l$$

⑤ n비트 벡터 Z를 계산

$$Z=CA(V)$$

4.3 출력 함수 $g()$

출력 함수 $g()$는 다음처럼 계산된다.

① H_m을 PCA의 초기값으로 load

② 압축 함수 $h()$에서 마지막 H_m계산시의 X,Y,V,Z를 이용한다.

- n비트 X,Y,V,Z를 $X_1,X_2,\cdots,X_{n/8}$, Y_1,Y_2, $\cdots,Y_{n/8}$, $V_1,V_2,\cdots,V_{n/8}$,$Z_1,Z_2,\cdots,Z_{n/8}$으로 분할(X_i,Y_i,V_i,Z_i는 각각 8 bit)
- PHT(X_i, V_i), i=1,2,…, $\frac{n}{8\cdot 2}$
- PHT(V_i, Y_j), $i=\frac{n}{8\cdot 2}$, $\frac{n}{8\cdot 2}+1$,…, $\frac{n}{8}$
- PHT(Z_i, X_j), i=1,2,…,$\frac{n}{8\cdot 2}$, $j=\frac{n}{8\cdot 2}$,$\frac{n}{8\cdot 2}+1$,…, $\frac{n}{8}$
- PHT(Y_i, Z_j), i=1,2,…, $\frac{n}{8\cdot 2}$, $j=\frac{n}{8\cdot 2}$,$\frac{n}{8\cdot 2}+1$,…, $\frac{n}{8\cdot 2}$

③ 다음을 사용자가 지정한 출력 비트 L만큼 사이클을 수행한다. 매 사이클마다 PCA()의 상태값 중 중간비트를 출력으로 취한다.

$$X'=CA(X),\ Y'=CA(Y)$$
$$PCA_{X'Y'}(H_m)$$

4.4 해쉬함수 h()

제안된 해쉬함수 $h()$는 다음과 같은 단계를 거쳐 수행된다.

(1) 입력 : 메시지 M과 n 비트 초기값 IV

(2) 전처리 : MD 계열 해쉬함수에서 사용하는 것과 같은 방식의 MD-strengthening과 padding 수행

- 전처리된 메시지 M을 $4n$비트의 m 블록, M_1, M_2, …, M_m으로 분할한다.

(3) 반복 처리 : H_0=IV, i=1, 2, …, m에 대해 다음을 수행한다.

- 압축 함수 $f()$를 계산한다.

$$H_i=f(M_i,\ H_{i-1})$$

(4) H_m이 모두 0 인 벡터이면, 다음처럼 H_m을 다시 계산한다.

$$H_m=f(M_m,\ H_0)$$

(5) 출력 함수 : $g(H_m)$을 계산한다.

(6) 출력 : L비트 메시지 다이제스트

$$h(M)=g(H_m)$$

V. 제안된 해쉬함수의 분석

2절의 정리 1에 의해 제안된 해쉬함수의 안전성에 관한 하한은 압축 함수와 출력 함수의

특성에 의해 결정된다. 따라서 제안된 해쉬함수의 안전성은 $f()$와 $g()$의 안전성을 통해 고려된다. 두 함수의 안전성은 preimage와 second preimage 그리고 충돌 공격에 관해 분석된다.

각 메시지 블록 M_i, i=1, 2, …, m의 처리는 다음으로 구성된다.

- M_i와 H_{i-1}을 하나의 n차원 이진 벡터 X로의 비선형 사상
- CA의 현재 상태에서 다음 상태로의 사상
  : $Y=CA(X)$
- X, Y, H_{i-1}의 PHT 적용
- X, Y, M_i, H_{i-1}의 하나의 n 차원 이진 벡터 V로의 비선형 사상
- CA의 현재 상태에서 다음 상태로의 사상
  : $Z=CA(V)$
- 벡터 Z와 같은 현재 상태를 다음 상태로 PCA 사상
  · PCA 구성 규칙은 X와 Y에 의해 제어된다.
- $PCA_{XY}(Z)$와 H_{i-1}의 비트 단위 mod 2 덧셈

따라서, 다음은 압축 함수의 안전성을 함축한다.

(1) CA는 최대 주기를 가지도록 primitive 특성 다항식을 가진다. 최대 주기 CA에 의해 생성된 패턴은 암호학적 기준을 만족한다[15].
(2) 사용된 비선형 부울 함수와 PCA로 인해 매우 높은 비선형성을 가진다[14][15].
(3) CA/PCA의 어떤 출력을 이용하여 CA/PCA의 상태를 재구성하기 위해 현재까지 알려진 알고리즘은 $f()$에서 동작하지 않는다[14].
(4) 압축 함수는 암호학적 변형이다.

위의 4가지 사항은 제안된 해쉬함수의 압축함수 $f()$가 암호학적으로 안전한 함수라는 것을 의미한다. 따라서 가정 1에 의해 다음이 성립한다.

: 주어진 f() 출력에 대해 preimage 발견은 약 2n 연산을 요구하고, f()에 대한 충돌 발견은 약 2n/2 연산을 요구한다.

출력 함수 $g()$는 CA와 PCA에 의해 2 단계로 구성된 키스트림 생성기로 볼 수 있다. 이 역시 CA가 최대 주기를 가지도록 primitive 특성 다항식을 가진다. 그리고 사용된 PCA로 인해 높은 비선형을 가지며, 또한 암호학적 변형이다. 따라서 n비트 출력을 가정하면, 주어진 해쉬값에 대한 $g()$의 입력, 즉 H_m 발견은 약 2^n연산을 요구한다. 그리고 출력 함수에 대한 충돌 발견은 약 $2^{n/2}$연산을 요구한다고 기대할 수 있다.

다음으로 제안된 해쉬함수의 복잡도를 고려한다. 먼저 압축 함수 계산에서의 복잡도는 다음과 같다.

① 각 부울 함수 ϕ_i를 $\frac{n}{l\times5}$번씩 적용과 $\frac{n}{l}\times$ 2번의 mod 256 덧셈
② n비트 CA동작(=3n XOR)
③ $\frac{n}{8\times2}\times4\times2$번의 mod 256 덧셈과 $\frac{n}{8\times2}\times$ 4번의 1 bit 왼쪽 쉬프트
④ 각 부울 함수 ϕ_i를 $\frac{n}{l\times5}$번씩 적용과 $\frac{n}{l}\times2$번의 mod 256 덧셈
⑤ n비트 CA 동작(=3n XOR)
⑥ PCA_{XY} 동작(=3n XOR)
⑦ PCA_{XY}와 H_{i-1}과의 n비트 XOR

따라서 압축 함수에서는 $\{\frac{4n}{l}+\frac{n}{2}$번의 mod 256 덧셈 $+\frac{n}{4}$번의 1 비트 왼쪽 쉬프트 + 2번의 n비트 CA동작(=$2\times3n$ XOR)+1 번의 PCA_{XY}동작(=3n XOR)+ $\frac{n}{l\times5}\times15\times2$번의 AND+$\frac{n}{l\times5}\times13\times2$번의 XOR+$\frac{n}{l\times5}\times2\times2$번의 OR+$\frac{n}{l\times5}\times1\times2$번의 NOT + 1번의 n 비트의 XOR$\}$ 연산으로 처리될 수 있다. 다음으로 출력 함수의 복잡도는 다음과 같다.

① $\frac{n}{8\times2}\times4\times2$번의 mod 256 덧셈 + $\frac{n}{8\times2}\times4$번의 1 bit 왼쪽 쉬프트

② $2\times L$사이클의 CA, L사이클의 PCA_{XY} 동작 (L: 출력 길이)

m개 메시지 블록을 처리하기 위해 요구되는 복잡도는 다음과 같다(비트 단위 AND, XOR, OR, NOT 연산이 모두 동일하다고 가정하고, n=160, l=8, $L=n$).

$80(2m+1)$번의 mod 256 덧셈 + $40(m+1)$번 1비트 왼쪽 쉬프트 + $(2m+320)$ 사이클의 CA+ $(m+160)$ 사이클의 PCA_{XY}+$248m$번의 비트 단위 논리연산 + m번의 160비트 XOR

요구되는 메모리는 $4n$비트 M_i, n비트 H_i,X,Y,V,Z, n비트 임시버퍼로 총 $10n$비트의 메모리가 요구된다.

기 제안된 CA 기반 해쉬함수와 제안된 해쉬함수를 비교한다. Daemen[1]은 비선형 CA와 선형 CA를 사용하였고 Hirose[6]의 기법은 두 개의 비선형 CA 적용으로 볼 수 있다. 그러나 두 기법에서 사용된 비선형 CA는 최근에 발표된 inversion 알고리즘에 속하는 형태들이다[6]. 제안된 기법의 압축 함수는 Davies-Meyer 유형이고 비선형 함수와 PCA의 결합 형태로서 Daemen과 Hirose의 기법에 비해 높은 안전성을 가진다. 두 기법이 출력 함수를 가지고 있지 않은데 비해 제안된 기법은 CA와 PCA에 기반한 2단계 키스트림 생성기 형태의 출력 함수를 가진다. Mihaljevic[11]의 기법은 제안된 기법과 유사하게 Davies-Meyer 유형의 압축 함수에 비선형 함수와 PCA의 결합한 형태를 적용한다. 제안된 기법의 경우 비트 단위 논리연산만을 사용하는 5변수 비선형 함수를 적용하는데 비해 Mihaljevic의 경우 S-Box 형태의 비선형 함수를 적용하여 구현시에 비선형 함수를 위한 ROM을 요구하고, 연산시에 메모리 reading을 요구한다. 또한 PCA 구조에서 Mihaljevic의 경우 $PCA_X()$의 형태로 이진 벡터 X에 의존하여 규칙 150과 규칙 90을 적용하는데 반해 제안된 기법에서는 $PCA_{XY}()$의 형태로 두 개의 이진 벡터 X,Y에 의존하여 4가지 규칙 중 하나를 적용하는 조금 더 복잡한 형태를 적용하여 안전성을 증가시킨다. 또한 출력 함수로 Mihaljevic의 경우 PCA에 기반한 키 스트림 생성기를 사용하며, 제안된 기법에서도 유사한 CA와 PCA의 결합 형태 출력 함수를 사용한다. 두 출력 함수는 거의 같은 수준의 안전성을 가지는 것으로 고려된다. 하드웨어 구현시의 복잡도는 비트단위의 논리연산을 사용하는 비선형 함수를 적용함으로써 Mihaljevic의 기법에 비해 메모리 요구량을 현저히 감소시킨다.

n=160, l=8, K=3일 때 Mihaljevic의 기법의 복잡도는 다음과 같다.

(1) 압축 함수

① 40번의 ROM reading
② 20번의 ROM reading
③ 160비트 CA 동작(=480 XOR)
④ 20번의 ROM reading
⑤ 160비트 PCA_X 동작(=480 XOR)
⑥ 160번의 XOR

(2) 출력 함수

160번의 mod η덧셈 + 160번 ROM reading + 160 사이클 PCA_X 동작 + 160 비트 permutation

제안된 기법은 다음과 같은 복잡도를 가진다.

(1) 압축 함수

① 40번의 mod 256 덧셈 + 124번의 XOR
② 160비트 CA 동작(=480 XOR)
③ 80번의 mod 256 덧셈 + 40번의 1비트 shift
④ 40번의 mod 256 덧셈 + 124번의 XOR
⑤ 160비트 CA 동작(=480 XOR)
⑥ 160비트 PCA_{XY} 동작(=480 XOR)
⑦ 160번의 XOR

(2) 출력 함수

① 80번의 mod 256 덧셈 + 40번의 iqlxm shift
② 360 사이클 CA동작 + 160 사이클 PCA 동작

위의 비교에서 보면, 제안된 기법이 많은 연산량을 요구하는 것으로 보이지만, 압축 함수의 처리시에 제안된 기법은 $4n$비트의 입력을 처리하고 Mihaljevic의 기법은 $2n$ 비트의 입력을 처리한다. 따라서 같은 크기의 입력을 처리할 때 Mihaljevic의 기법은 제안된 기법보다 2배의 압축 함수 적용을 요구한다. 640비트의 입력을 처리한다고 가정하면 Mihaljevic의 기법은 (80번의 ROM reading + 2240번의 XOR)의 연산을 요구하고 제안된 기법은 (160번의 mod 256 덧셈 + 40번의 1비트 shift + 1848번의 XOR)의 연산을 요구한다. 따라서 제안된 기법은 Mihaljevic의 기법과 유사한 계산 복잡도를 가지는 것으로 볼 수 있다. 메모리 요구량을 살펴보면, $n=160$, $l=8$,$k=3$을 가정했을 때 Mihaljevic의 기법은 약 1546Kbits의 ROM과 약 800bits의 버퍼를 요구하지만 제안된 기법은 약 1600bits의 버퍼만을 요구한다. 하지만 제안된 기법은 2개의 CA와 비선형 부울 함수, PHT의 사용으로 인해 구현의 복잡도는 증가한다.

기 제안된 기법들은 모두 고정된 길이의 해쉬 출력을 가지는데 비해 제안된 기법은 가변 길이 출력으로 다양한 응용에 사용될 수 있다.

Ⅵ. 결 론

해쉬함수는 임의 길이의 비트 스트링을 입력으로 받아 고정된 짧은 길이(주로 128, 160 비트)의 비트 스트링을 출력하는 함수로, 해쉬함수는 현대 암호학에서 데이터 무결성과 인증을 제공하기 위한 도구로 매우 빈번하게 사용된다. 본 논문에서는 CA에 기반한 새로운 해쉬함수를 제안하였다. 제안된 해쉬함수의 압축 함수는 CA에 기반한 Davies-Meyer 유형이고, 출력 함수 역시 CA에 기반한 키 스트림 생성기 형태이다. CA의 사용은 제안된 해쉬함수의 효율성을 보장한다.

제안된 해쉬함수의 안전성이 압축 함수와 출력 함수의 안전성을 통하여 분석되었다. 이러한 분석은 제안된 해쉬함수가 이상적인 안전성을 가지는 것을 함축한다. 즉 주어진 n 비트 해쉬값에 대한 preimage 발견은 약 2^n연산을 요구하고, 충돌 발견은 약 $2^{n/2}$ 연산을 요구한다. 또한 m개 메시지 블록을 처리하기 위해 제안된 기법은 총{$80(2m+1)$번의 mod 256 덧셈 + $40(m+1)$번 1비트 왼쪽 쉬프트 + $(2m+320)$ 사이클의 CA+$(m+160)$ 사이클의 PCA_{XY} + $248m$번의 비트 단위 논리연산 + m번의 160비트 XOR}연산을 요구한다(L: 해쉬값의 비트 길이).

참 고 문 헌

[1] J. Daemen, R. Govaerts, J. Vandewalle, "A framework for the design of one-way hash functions including cryptanalysis of Damgård's one-way function based on cellular automaton", Advances in Cryptology-ASIACRYPTO 91, LNCS, vol. 739, 1993

[2] I.B. Damgård, "A design principle for hash functions", Advances in Cryptology-CRYPTO 89, LNCS, vol. 435, pp. 416-427, 1990

[3] 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

[4] FIPS 180-1, "Secure hash standard", Federal Information Processing Stan-

dards Publication 180-1, U.S. Department of Commerce / Nist, 1995

[5] S. Hirose, S. Yoshida, "A one-way hash function based on a two-dimensional cellular automaton", The 20th Symposium on Information Theory and Its applications(SITA'97), Japan, vol. 1, pp. 213-216, 1997

[6] C.K. Koc, A.M. Apohan, "Inversion of cellular automata iterations", IEE Proc.-Comput. Digit. Tech., vol. 144, pp. 279-284, 1997

[7] L. Knudsen, B. Preneel, "Fast and secure hashing based on codes", Advances in Cryptology-CRYPTO 97, LNCS, vol. 1294, pp. 485-498, 1997

[8] W. Meier, O. Staffelbach, "Analysis of pseudo random sequences generated by cellular automata", Advances in Cryptology-EUROCRYPT 91, LNCS, vol. 547, pp. 186-189, 1992

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

[10] R. Merkle, "One way hash functions and DES", Advances in Cryptology-CRYPTO 89, LNCS, vol. 435, pp. 428-446, 1990

[11] M. Mihaljevic, "Security examination of cellular automata based key stream generator", ISITA 96 - 1996 IEEE Int. Symp. Infor. Theory and Appl., Canada, 1996, Proc. pp. 246-249

[12] M. Mihaljevic, "Security examination of cellular automata based pseudorandom generator using an algebraic replica approach", Applied Algebra, Algorithms and Error Correcting Codes - AAECC 12, LNCS, vol. 1255, pp. 250-262, 1997

[13] M. Mihaljevic, "An improved key stream generator based on the programmable cellular automata", Information and Communication Security - ICICS 97, LNCS, vol. 1334, pp. 181-191, 1997

[14] M. Mihaljevic, Y. Zheng, H. Imai, "A Cellular Automaton Based Fast One-Way Hash Function Suitable for Hardware Implementation", Public Key Cryptography - Proceedings of PKC'98, LNCS, Vol. 1431, 1998

[15] S. Nandi, B.K. Kar, P.Pal Chaudhuri, "Theory and applications of cellular automata in cryptography", IEEE Trans. Computer, vol. 43, pp. 1346-1357, 1994

[16] B. Preneel, R. Govaerts, J. Vandewalle, "Hash functions based on block ciphers: a synthetic approach", Advances in Cryptology-CRYPTO 93, LNCS, vol. 773, pp. 368-378, 1994

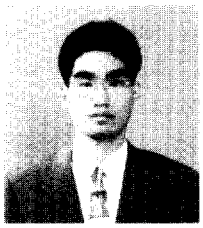
[17] R.L. Rivest, "The MD4 message-digest algorithm", Advances in Cryptology-CRYPTO 90, LNCS, vol. 537, pp. 303-311, 1991

[18] RFC 1321, "The MD5 message-digest algorithm", Internet request for comments 1321, R.L. Rivest, April 1992

[19] S. Wolfram, "Cryptography with cellular automata", Advances in Cryptology- CRYPTO 85, LNCS, vol. 218, pp. 429-432, 1985

[20] Y. Zheng, J. Pieprzyk, J. Sebery, "HAVAL - a one-way hashing algorithm with variable length of output", Advances in Cryptology-AUSCRYPT 92, LNCS, vol. 718, pp. 83-104, 1993

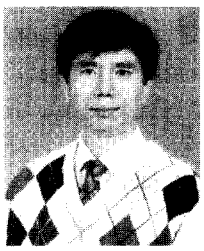
□ 著者紹介



신 상 욱

1995년 부산수산대학교(현 부경대학교) 전자계산학과 졸업(이학사)
1997년 부경대학교 전자계산학과 대학원 졸업(이학석사)
1998년 부경대학교 전자계산학과 박사과정수료
1997년 ~ 현재 부경대학교 시간강사

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



윤 재 우

1983년 전북대학교 전자공학과 졸업(공학사)
1985년 전북대학교 대학원 전자공학과 졸업(공학석사)
1997년~현재 전북대학교 대학원 전자공학과 박사과정 재학중
1989년~현재 한국전자통신연구원 선임연구원

※ 주관심 분야 : 정보보호 알고리즘, 정보보호 시스템, 전자상거래, 분산처리시스템, 암호칩 설계

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

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