Journal of Korea Multimedia Society Vol. 8, No. 5, May 2005(pp. 670-678)

이차원 셀룰라 오토마타에 기반하는 해쉬 함수

김 재 겸†

요 약

셀룰라 오토마타(CA, Cellular Automata)란 동역학계(Dynamical Systems)를 해석하는 한 방법으로 공간과 시간을 이산적으로 다루고 셀룰라 공간(Cellular Space)의 기본 단위인 각 셀(Cell)이 취할 수 있는 상태를 유한하게 처리하며, 각 셀의 상태가 국소적인 상호작용(Local Interaction)에 의해서 동시에 갱신되는 시스템이다. CA는 그 본질적인 특성이 확산(Diffusion)과 국소적인 상호 작용(Local Interaction)이므로 암호 시스템과 VLSI 환경에 적합한 것으로 알려져 있다. 본 연구에서는 새로운 구조의 이차원 셀룰라 오토마타를 제안하고 제안된 이차원 셀룰라 오토마타에 기초한 새로운 구조의 해쉬 함수를 제안하고 제안된 해쉬 함수의 통계적 검증 결과를 밝힌다. 또한 제안된 해쉬 함수는 키를 사용하는 경우와 그렇지 않은 경우에 모두 적용 가능하도록 구성되었다.

A Hash Function Based on 2D Cellular Automata

Jae-Gyeom Kim†

ABSTRACT

A Cellular Automaton(CA) is a dynamical system in which space and time are discrete, the state of each cell is finite and is updated by local interaction. Since the characteristics of CA is diffusion and local interaction, CA is used by crypto-systems and VLSI structure. In this study, we proposed a hash function based on the concept of 2-dimensional cellular automata and analyzed the proposed hash function.

Key words: Cellular Automaton(셀룰라 오토마타), Hash Function(해쉬 함수)

1. 서 론

본 연구의 목적은 안전하면서 수행속도가 빠른 전용 해쉬 함수를 개발하는 것이다.

인터넷의 발전은 21세기 e-business 시대로의 새로운 전환을 가져왔고, 인터넷의 효율적인 활용이 모든 분야와 조직에서 경쟁력을 좌우하고 있다. 인터넷의 특성인 개방화 및 분산화의 환경에서 각종 전자상거래, 전자문서, 전자우편 등에 이용되는 정보와 인터넷에 접속된 시스템과 LAN 등의 네트워크 보호를 위한 각종 보안기술에 대한 중요성이 날로 부각되고 있다.

이러한 보안기술의 핵심적인 요소 중의 하나가 안전하면서도 속도가 빠른 전용 해쉬 함수이다.

해쉬 함수란 임의의 크기의 메시지를 입력으로 받아 고정된 크기의 출력 값을 생성해 내는 함수로 해쉬 값이 입력 스트링의 압축된 대표 이미지로 작용하고, 그 스트링을 유일하게 식별할 수 있는 것으로 사용될 수 있다는 점에 의하여 사용자 인증, 전자화폐, 전자서명, 공개키 암호 알고리즘 등의 여러 분야에서 사용하고 있다[1].

해쉬 함수는 입력 파라미터의 종류에 따라 하나의 입력 파라미터-메시지-를 가지는 unkeyed 해쉬 알고리즘과 두 개의 다른 입력 파라미터-메시지와 비

※ 교신저자(Corresponding Author) : 김재겸, 주소 : 부산광역시 남구 대연동 경성대학교 수학과(608-736), 전화 : 051)620-4612, 팩스 : 051)622-7510, E-mail : jgkim@ks.ac.kr
접수일 : 2004년 4월 1일, 완료일 : 2004년 12월 30일
† 정회원, 경성대학교 수학과 교수
※ 이 논문은 2003학년도 경성대학교 학술지원연구비에 의하여 연구되었음.

밀키-를 가지는 keyed 해쉬 함수의 두 분류로 나누어 질 수 있으며, 해쉬 함수의 전체 구성에 따라 DES와 같은 블록 암호 알고리즘에 기초한 해쉬 함수와 전용 해쉬 함수로 나눌 수 있다. 블록 암호 알고리즘을 이용한 해쉬 함수는 이미 구현되어 사용되고 있는 블록 암호 알고리즘을 사용할 수 있다는 이점이 있으나, 대부분의 블록 암호알고리즘의 속도가 그리 빠르지 않을뿐더러 이를 기본함수로 이용한 해쉬 함수의 경우 블록 암호알고리즘보다도 훨씬 더 속도가 떨어지므로 현재는 대부분의 응용에서 전용 해쉬 함수가 주로 이용된다.

국내에서도 HAS160 이라는 전용 해쉬 함수를 표준으로 사용하고 있다[2].

1985년 동역학계(Dynamical Systems)의 일종인 CA(Cellular Automata)를 이용한 암호 시스템이 Wolfram에 의해서 최초로 제안된 후[3], 1994년 Nandi 등에 의해서 PCA(Programmable Cellular Automata)를 이용한 암호 시스템들이 제안된 바 있고[4], 1997년 Mihaljević에 의해서 Nandi 등의 시스템을 개선한 암호 시스템이 제안된 바 있다[5].

CA는 그 본질적인 특성이 확산(Diffusion)과 국소적인 상호 작용(Local Interaction)이므로 암호 시스템과 VLSI 환경에 적합한 것으로 알려져 있다.

CA는 셀들의 배열의 1차원 구조에 기초한 1차원 CA와 셀들의 배열의 2차원 이상의 다차원 구조에 기초한 다차원 CA로 나눌 수 있는데, 다차원 CA는 그 구조의 복잡성으로 인해 분석이 어렵고, 각 셀의 상태가 다른 셀들의 상태에 영향을 주는 속도가 1차원 CA에 비하여 훨씬 더 빠르므로 1차원 CA보다 암호 시스템에 적합하다. 그러나 현재까지의 CA의 암호 시스템에의 응용에 관한 연구는 주로 1차원 CA에 머물러 왔는데, 이는 1차원 CA의 이론적인 연구의 용이성 때문이다.

본 연구에서는 새로운 구조의 이차원 CA를 제안하고 이를 이용한 전용 해쉬 함수를 제안한다.

본 논문의 구성은 다음과 같다. 2절에서는 해쉬 함수에 관한 정의와 일반적 모델에 관해, 3절에서는 셀룰러 오토마타에 관한 기본 이론을 기술한다. 4절에서는 새로운 구조의 이차원 CA를 제안하고 5절에서는 이 구조를 이용한 전용 해쉬 함수를 제안하고 제안된 전용 해쉬 함수의 통계적 특성에 기초한 안전성을 분석하며 마지막으로 6절에서는 결론을 맺는다.

2. 해쉬 함수의 정의와 일반적 모델

해쉬 함수(hash function) 즉, 암호학적 해쉬 함수(cryptographic hash function)는 임의의 유한 길이 비트 스트링을 고정된 길이의 스트링으로 사상시키는 함수이다. 이 출력은 흔히 해쉬 값(hash value), 메시지 다이제스트(message digest) 등으로 불린다. 암호학적으로 안전한 해쉬 함수는 함수 h와 입력 x가 주어지면, 해쉬 값 h(x)를 계산하는 것은 쉬워야 하지만 함수 h와 해쉬값 h(x)로부터는 입력 x를 찾는 것이 계산상 불가능해야 한다는 일방향성을 가져야 한다.

일방향 해쉬 함수는 다음 성질을 만족해야 한다.

- preimage resistance : 어떤 미리 명시된 출력으로 해쉬하는 어떤 입력을 발견하는 것이 계산상 수행 불가능하다. 즉, 해쉬 값 y가 주어 졌을 때, h(x)=y를 만족하는 입력 x를 발견하는 것이 계산상 수행 불가능하다.

-2nd preimage resistance : 어떤 명시된 입력과 같은 출력을 가지는 어떤 두 번째 입력을 발견하는 것이 계산상 수행 불가능하다. 즉, 입력 x와 출력 h(x)가 주어졌을 때, h(x)=h(x′)을 만족하는 입력 $x \neq x'$를 발견하는 것이 계산상 수행 불가능하다.

암호학적으로 유용한 해쉬 함수는 다음 성질을 추가로 만족해야 한다.

-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, x′을 발견하는 것이 수행 불가능하다.

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 n 개의 블록으로 나누어진다.[1]

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 갱신하다. 이 과정이 모든 메시지 블록에 대해

순환적으로 반복되고, 연쇄 변수의 마지막 값이 그 메시지에 대한 해쉬 값으로 출력된다.

해쉬 함수는 내부 압축 함수로 어떤 구조를 사용하느냐에 따라 3가지로 분류된다[1].

① 블록암호에 기반한 해쉬 함수
② 모듈러 연산에 기반한 해쉬 함수
③ 전용 해쉬 함수

전용 해쉬 함수는 빠른 처리 속도를 가지고 다른 시스템 서브 요소에 무관하도록 해쉬을 위해 특별히 설계된 함수들이다. 현재까지 제안된 전용 해쉬 함수는 대부분 1990년에 Rivest에 의해 설계된 MD4에 기반한 구조를 가진다. 현재 널리 사용되는 MD 계열 해쉬 함수로 MD5, SHA-1, RIPEMD-160, HAVAL, HAS160 등이 있다.[2]

특정 해쉬 함수가 주어지면 안전한 해쉬 함수의 입증을 위해 해쉬 함수를 공격하는 복잡도에 관한 하한을 증명할 수 있는 것이 바람직하지만 실제 그러한 방법은 거의 알려져 있지 않다. 대부분의 경우에 적용 가능한 알려진 공격의 복잡도가 해쉬 함수의 안전성으로 고려되어진다.

해쉬값이 균등한 확률 변수라고 가정하면, 다음과 같은 사실들이 알려져 있다.[1]

- n 비트 해쉬 함수 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 해쉬 함수가 다음 두 가지를 만족한다면 이상적인 안전성을 가진다[1].

① 해쉬값이 주어지면, preimage와 2nd preimage 발견은 약 2^n 연산을 요구한다.
② 충돌 발견은 약 $2^n/2$연산을 요구한다.

3. 셀룰라 오토마타(Cellular Automata, CA)

셀룰라 오토마타란 동역학계를 해석하는 한 방법으로 공간과 시간을 이산적으로 다루고, 이산적인 공간인 셀룰라 공간(Cellular Space)의 기본 단위인 각 셀이 취할 수 있는 상태를 유한하게 처리하며, 각 셀의 상태가 국소적인 상호작용에 의해서 동시에 갱신되는 시스템이다.

가장 간단한 구조를 가지는 CA는 1차원 CA로써 국소적인 상호작용이 인접하는 3개의 셀들에 의해서 이루어지는 1차원 3-이웃 CA이다.

다음 기호는 1차원 CA를 특징짓는데 사용되어진다.

i : 1차원 배열에서 각 셀의 위치
t : 시간 단계
$q_i(t)$: t 번째 시간 단계에서 i번째 셀의 상태
$q_i(t+1)$: $(t+1)$ 번째 시간 단계에서 i번째 셀의 상태

3-이웃 CA에서의 각 셀의 상태의 동시 갱신에 사용되는 다음상태 전이함수는 다음과 같이 표현된다.

$$q_i(t+1) = f\left[q_{i-1}(t),\ q_i(t),\ q_{i+1}(t)\right]$$

주어진 CA의 다음상태 전이함수를 진리표 형태로 표현하였을 때, 진리표의 출력 열을 이진수로 보고 이 이진 출력 열의 십진수를 주어진 CA의 Rule이라 부른다. 즉, 시간 t 에서 CA의 이웃하는 3개의 셀들($i-1$번째 셀, i번째 셀, $i+1$번째 셀)의 가능한 8가지 모든 상태의 배열들을 3자리의 이진수로 보고 큰 순서대로 왼쪽에서 오른쪽으로 배열한 후 각각의 상태들로부터 전이 함수를 적용시켜 나온 결과 값들을 8자리의 이진수로 표현하고 이를 다시 십진수로 표기 한 것을 의미한다.

만약 CA의 모든 셀에 같은 Rule이 적용되면 그러한 CA를 Uniform CA라하고, 그렇지 않으면 Hybrid CA라 한다. 그리고 셀들에 적용되는 Rule들을 규정한 것을 Configuration이라 한다.

1차원 CA에서 가장 왼쪽(오른쪽) 끝의 셀들은 2개의 이웃밖에 가지지 못하는데, 그러한 셀들에 3번째의 이웃을 정해주는 방법을 경계조건이라 한다. 1차원 CA에서는 다음과 같은 3가지의 경계조건을 이용하는 것이 일반적이다. 즉, 가장 왼쪽(오른쪽) 끝의 셀의 왼쪽(오른쪽) 이웃이 논리 0 상태에 연결되는 CA를 NBCA(Null Boundary CA), 양끝의 셀들이 서로 이웃하는 CA를 PBCA(Periodic Boundary CA), 가장 왼쪽(오른쪽) 셀의 다음 상태가 그 자신과 오른

쪽(왼쪽) 이웃과 두 번째 오른쪽(왼쪽) 이웃에 의존하는 CA를 IBCA(Intermediate Boundary CA)라 한다.

CA에서 비교적 다루기 쉬운 부류는 GF(2)에서의 선형 CA와 Additive CA이다. Rule이 XOR 논리만을 포함하면 선형 Rule이라 하고, XNOR 논리만을 포함하면 Complemented Rule이라 한다. 선형 Rule만을 가지는 CA를 선형 CA라 한다. 반면에 XOR 논리와 XNOR 논리의 조합으로 구성되는 CA를 Additive CA라 한다.

선형 CA는 LFSM(Linear Finite State Machine)의 특별한 형태이다. 모든 LFSM은 GF(2)상에서의 전이 행렬로 나타낼 수 있고, 모든 전이 행렬은 특성 다항식을 가진다. 따라서 선형 CA는 그것에 대응되는 특성다항식을 가짐을 알 수 있다.

만약 CA의 특성 다항식이 Primitive이면 그러한 n-셀 CA는 연속되는 Cycle에서 모두 0인 상태를 제외한 2^n-1가지의 상태들을 생성하며, 이러한 CA를 Maximal Length CA라 한다.

1차원 3-이웃 CA의 통계적 조건에 근거하여 Wolfram은 CA들을 다음 4가지의 유형으로 분류하였다[3].

- 대부분의 초기 상태들이 유한번의 갱신 후에 동일한 상태로 전이되는 CA
- 각 상태가 비교적 짧은 주기를 가지는 CA
- 대부분의 초기 상태들이 비주기적이거나 의사 랜덤한 행동을 나타내는 CA
- 복잡한 국소화와 확산 구조를 가지는 CA

위의 4가지 유형 중에서 특히 세 번째에 속하는 CA들의 경우 의사 랜덤 패턴 생성기로서 가장 적절한 것으로 알려져 있다. Rule 30의 1차원 Uniform CA, Rule 45의 1차원 Uniform CA, Rule 90과 Rule 150의 1차원 Hybrid CA, 그리고 Rule 30과 Rule 45의 1차원 Hybrid CA의 경우가 세 번째에 속하는 유형들로 알려져 있다.[1]

모든 LFSR은 1차원 CA로 표현 가능하며, CA에 의해 생성된 패턴들의 랜덤성은 LFSR에 기초한 생성기들에 의해 생성된 패턴들보다 우수한 것으로 알려져 있다. 프로그램 가능한 CA를 이용할 경우 이러한 랜덤성은 더욱 향상된다. 그리고 CA는 셀들의 상태가 국소적인 상호작용에 의해서 갱신되어지는 특성으로 인하여 H/W 구현시 고속으로 구현 가능한 장점을 가진다.

프로그램 가능한 셀룰라 오토마타(PCA, Programmable Cellular Automata)는 각 셀에 적용되는 Rule이 시간 단계에 따라 변하는 CA를 뜻한다. 예를 들어 Rule 90과 Rule 150은 이웃에 대한 의존도가 한군데의 위치에서만 다르다. 따라서 그림 1에서와 같이 하나의 제어 선을 이용하여 회로를 열고 닫음으로써 각 시간 단계에서 하나의 셀에 Rule 90과 Rule 150을 선택적으로 적용할 수 있다.[1]

그림 1과 같은 제어구조를 가지는 CA는 가장 단순한 PCA 중의 한 종류이다.

그림 2는 3개의 제어 선을 이용하는 제어구조이다. 이러한 제어구조의 각 회로들의 열고 닫음의 조합을 이용하면 3-이웃 CA의 모든 종류의 선형 Rule을 각 시간 단계에 따라 선택적으로 적용할 수 있다. 이와 같은 제어구조를 다양화하면 모든 종류의 Rule을 각 시간 단계에 따라 선택적으로 적용할 수 있다.

따라서 PCA 구조를 이용하면 매우 다양한 Configuration을 구현할 수 있다.

예를 들어 그림 2의 제어구조를 각 셀에 적용할 경우, n개의 셀을 가지는 1차원 3-이웃 PCA에 적용될 수 있는 Configuration의 개수는 2^{3n}이다.

2-D CA는 1-D CA의 일반화로서 이웃하는 셀들이 두 개의 축 상에 배열되어 있다. 셀의 다음 상태는 그 자신과 인접한 네 가지의 셀(위, 왼쪽, 아래, 오른쪽)의 상태에 영향을 받는 5 이웃 셀룰라 오토마타가 일반적으로 이용되며 이 경우 다음 식으로 표시가 된다. 이때 q_{ij}는 시간 t에서 i행 j 열의 셀 상태를 나타낸다.

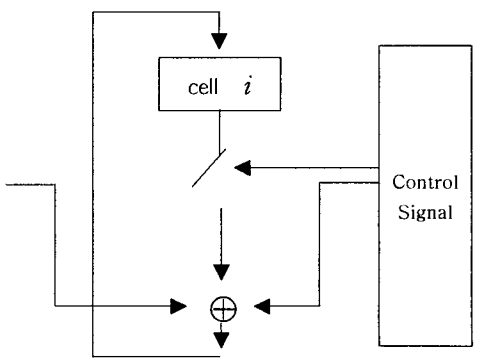

그림 1. PCA type 1

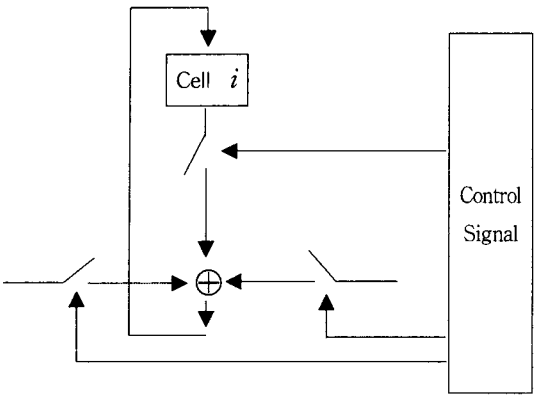

그림 2. PCA type 2

f는 5개의 변수를 가지는 Boolean 함수이므로 2^5, 즉 32개의 서로 다른 배열을 가진다. 아래의 그림 3은 다섯 이웃에 의존하는 셀을 나타내고 있다.

	위	
왼쪽	자신	오른쪽
	아래	

그림 3. 2차원 셀룰라 오토마타 기본 구조

$m \times n$개의 셀들이 m행 n열로 배열되어 있는 2-D CA의 전이행렬은 $mn \times mn$행렬로서 표현된다.

2-D CA에서의 rule은 5 비트의 수(자신, 위, 왼쪽, 아래, 오른쪽)로 표시하고 각각의 비트는 의존도를 나타낸다. 즉 만약 k의 셀의 다음 상태가 위와 왼쪽의 셀의 상태에만 의존한다면 이때의 rule은 이진수 01100 또는 12로 표시한다. 예를 들어 아래 그림과 같은 3×3 2차원 CA에서 1번과 3번 6번 9번 셀의 rule 00111이고 나머지 셀들의 rule 이 11101일 때 rule matrix R은 다음과 같이 나타낼 수 있다.

$$R=\begin{bmatrix} 7 & 29 & 7 \\ 29 & 29 & 7 \\ 29 & 29 & 7 \end{bmatrix}$$

2-D CA를 다루는 한 가지 방법은 2-D CA를 1-D CA의 확장으로 생각하여 상태전이 행렬을 이용하는 방법으로 2-D CA의 상태 전이 행렬을 구하는 방법은 다음과 같다.

우선 3×3의 2차원의 각 셀에 아래의 그림과 같이 왼쪽 위에서부터 차례로 번호를 매긴 후 i번째 행의 j번째 열의 값은 만약 j번째 셀의 현재 상태가 i번째 셀의 다음 상태 전이 함수의 입력이 되면 1로 그렇지 않으면 0으로 하여 9×9 크기의 행렬을 구성하면 된다.

1	2	3
4	5	6
7	8	9

그림 4. 2차원 CA의 셀 번호

예를 들어

$$R=\begin{bmatrix} 7 & 29 & 7 \\ 29 & 29 & 7 \\ 29 & 29 & 7 \end{bmatrix}$$

을 rule로 가지는 CA의 경우 1번 셀의 다음 상태 전이 함수의 입력으로 2번과 4번 셀의 현재 상태가 사용되므로 상태전이행렬에서 첫 번째 행벡터는 (0 1 0 1 0 0 0 0 0)이 된다.

일반적으로 $m \times n$ 2차원 CA의 전이행렬은 각각이 $n \times n$ 인 m^2개의 행렬로 분할된다. 따라서 위의 rule R을 가지는 CA의 전이행렬 T는 각각이 3×3 인 3^2개의 부분행렬로 나누어지며 전이행렬 T를 아래와 같이 각각 3×3 부분행렬로 구별하였을 경우 부분 행렬 S_1, S_2, S_3는 셀의 수평으로의 의존도에 대한 정보를 가지고 있으며, 부분행렬 L_i (i=1,2)는 i 행의 셀에 대한 아래쪽 셀의 의존도를 나타내고, U_i (i=2,3)는 i행의 셀에 대한 위쪽 셀의 의존도를 보여준다.

$$T=\begin{bmatrix} S_1 & L_1 & 0 \\ U_2 & S_2 & L_2 \\ 0 & U_3 & S_3 \end{bmatrix}$$

1-D CA와 마찬가지로 2-D CA에서도 경계조건을 고려해야 한다. 2-D CA의 경우는 1-D CA의 경우보다 더욱 다양한 경계 조건을 생각할 수 있으며 현재 고려되고 있는 2-D CA의 경계 조건들은 다음과 같다.

2-D NBCA는 2차원으로 배열된 CA의 경계 값이 모두 0인 경우이다.

IBCA는 각 열의 가장 왼쪽과 오른쪽의 경계 값에 따라서 세 가지로 나누고 위쪽 과 아래쪽의 경계 값

은 0으로 둔다.

① Pure 2-D IBCA (2-D IBCA type Ⅰ)

: 셀룰라 공간의 각 셀들을 배열로 표시하였을 경우 a_{11}와 a_{mn}에만 경계조건이 있는 경우로 a_{11}의 오른쪽 이웃은 a_{13}이고 a_{mn}의 오른쪽 이웃은 a_{mn-2}이며 나머지 셀들의 경계 값은 모두 0이다.

② Inner 2-D IBCA (2-D IBCA type Ⅱ)

: a_{i1}과 $a_{i\in}$에 ①의 경계 조건을 모두 적용시키는 것이다. 즉, a_{i1}의 왼쪽 이웃은 a_{i3}이고 $a_{i\in}$의 오른쪽 이웃은 $a_{i\in-2}$로 둔다

③ Outer 2-D IBCA (2-D IBCA type Ⅲ)

: a_{11} , a_{mn} 은 ①의 조건을 따르고 a_{i1} (i=2, ⋯,m)의 왼쪽 이웃은 a_{i-1n}으로, $a_{i\in}$ (i=1,⋯,m-1)의 오른쪽 이웃은 a_{i+11}로 두는 방법이다[.1].

4. 새로운 구조의 2차원 셀룰라 오토마타

셀룰라 공간 SP는 제안된 해쉬 함수에 사용되는 내부 함수로써 셀룰라 공간 SP의 기하적인 구성은 그림 5와 같이 36개의 사각형들로 분할되어 있는 평면도형으로, 각 셀은 각각 C_0번부터 C_{35}번까지의 셀 번호를 가지며, 각 셀은 8비트의 입력 값을 가진다.

C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}
C_{18}	C_{19}	C_{20}	C_{21}	C_{22}	C_{23}	C_{24}	C_{25}	C_{26}
C_{27}	C_{28}	C_{29}	C_{30}	C_{31}	C_{32}	C_{33}	C_{34}	C_{35}

그림 5. 셀룰라 공간 SP

동시변환 규칙은 셀룰라 공간 SP의 셀들의 값들을 시각의 변화에 따라 동시에 갱신하는 규칙들을 의미한다. 제안된 해쉬 함수에서는 선형 변환에 기초한 동시변환 규칙 L과 비선형 변환에 기초한 동시변환 규칙 NL이 사용되어 진다.

○ 연산 기호의 정의

X_i : i번째 셀의 현재 상태

X_{i-1} : i번째 셀의 왼쪽 이웃의 현재 상태

X_{i+1} : i번째 셀의 오른쪽 이웃의 현재 상태

Y_i : i번재 셀의 다음 상태

Y_{i-1} : i번째 셀의 왼쪽 이웃의 현재 상태

Y_{i+1} : i번째 셀의 오른쪽 이웃의 현재 상태

$\overline{X_i}$: i번째 셀의 현재 상태의 보수

$\oplus$: bitwise XOR

OR : bitwise OR

a<<<n : a의 현재 상태를 정수 n 만큼 왼쪽으로 로테이션

M_i : i번째 32비트 메시지 블록

A3 : 32비트 블럭 A의 오른쪽 끝을 0번으로 A의 왼쪽 끝을 31번비트라고 하였을 때, 24번 비트에서 31번 비트까지의 8비트

A2 : 32비트 블록 A의 오른쪽 끝을 0번으로 A의 왼쪽 끝을 31번비트라고 하였을 때, 16번 비트에서 23번 비트까지의 8비트

A1 : 32비트 블록 A의 오른쪽 끝을 0번으로 A의 왼쪽 끝을 31번비트라고 하였을 때, 8번 비트에서 15번 비트까지의 8비트

A0 : 32비트 블록 A의 오른쪽 끝을 0번으로 A의 왼쪽 끝을 31번비트라고 하였을 때, 0번 비트에서 7번 비트까지의 8비트

○ 동시변환 규칙 L

동시변환 규칙 L은 셀룰라 공간 SP의 셀들의 값들을 시각의 변화에 따라 동시에 갱신하는 규칙들 중의 하나로, 각 셀의 다음 상태가 자기 자신과 자신의 왼쪽 셀과 자신의 오른쪽 셀의 XOR한 값으로 갱신되는 선형규칙으로 아래의 식으로 표현된다.

$$Y_i = (X_{i-1} \oplus X_i \oplus X_{i+1})$$

이때 그림 5의 C_0, C_9, C_{18}, C_{27} 셀의 왼쪽 이웃은 각각 C_8, C_{17}, C_{26}, C_{35}번 셀로 정의하고, 그림 5의 C_8, C_{17}, C_{26}, C_{35}번 셀의 오른쪽 이웃은 각각 C_0, C_9, C_{18}, C_{27}번 셀로 정의한다.

○ 동시변환 규칙 NL

동시변환 규칙 NL은 셀룰라 공간 SP의 셀들의 값들을 시각의 변화에 따라 동시에 갱신하는 규칙들

중의 하나로, 각 셀의 다음 상태가 자기 자신과 자신의 왼쪽 셀과 자신의 오른쪽 셀의 XOR한 값과 자신과 자신의 오른쪽 셀의 보수를 XOR한 값을 다시 OR한 값으로 갱신되는 비선형규칙으로 아래의 식으로 표현 된다.

$$Y_i = ((X_{i-1} \oplus X_i \oplus X_{i+1})\, OR\, (X_i \oplus \overline{X_{i+1}}))$$

이때 그림 5의 C_0, C_9, C_{18}, C_{27} 셀의 왼쪽 이웃은 각각 C_8, C_{17}, C_{26}, C_{35}번 셀로 정의하고, 그림 5의 C_8, C_{17}, C_{26}, C_{35}번 셀의 오른쪽 이웃은 각각 C_0, C_9, C_{18}, C_{27}번 셀로 정의한다.

5. 제안된 해쉬 함수

본 연구에서 제안된 해쉬 함수는 padding 된 512(32×16)비트의 메시지 블록을 각각 32비트의 16개의 word M[i] (i=0...15)로 나눈 값을 입력으로 받아 각각 32비트인 5개의 해쉬 블록 hj(j=0...5)를 출력으로 하며 160비트의 사용자 키를 사용하여 아래와 같은 단계를 거친다.

단계 1:
A←0×67452301
B←0×efcdab89
C←0×98badcfe
D←0×10325476,
E←0×c3d2e1f0

단계 2:
AA←A, BB←B, CC←C, DD←D, EE←E
i ← 0, j ← 0

단계 3:
아래와 같은 방법으로 셀룰라 공간 SP에 입력

C_0←AA_3, C_1←$M[i]_3$, C_2←BB_3,
C_3←$K[j]_3$, $C4$←CC_3, C_5←$K[j+1]_3$,

C_6←DD_3, C_7←$M[i+1]_3$, C_8←EE_3,
C_9←AA_2, C_{10}←$M[i]_2$, C_{11}←BB_2,
C_{12}←$K[j]_2$, C_{13}←CC_2, C_{14}←$K[j+1]_2$,
C_{15}←DD_2, C_{16}←$M[i+1]_2$, C_{17}←EE_2,
C_{18}←AA_1, C_{19}←$M[i]_1$, C_{20}←BB_1,
C_{21}←$K[j]_1$, C_{22}←CC_1, C_{23}←$K[j+1]_1$,
C_{24}←DD_1, C_{25}←$M[i+1]_1$, C_{26}←EE_1,
C_{27}←AA_0, C_{28}←$M[i]_0$, C_{29}←BB_0,
C_{30}←$K[j]_0$, C_{31}←CC_0, C_{32}←$K[j+1]_0$,
C_{33}←DD_0, C_{34}←$M[i+1]_0$, C_{35}←EE_0

단계 4:
동시변환 규칙 L에 의하여 CA를 1회 갱신

단계 5:
C_3, C_{12}, C_{21}, C_{30}를 연접하여 왼쪽으로 1비트 로테이션

단계 6:
C_4, C_{13}, C_{22}, C_{31}를 연접하여 왼쪽으로 7비트 로테이션

단계 7:
C_6, C_{15}, C_{24}, C_{33}을 연접하여 왼쪽으로 13비트 로테이션

단계 8:
단계 4에서 단계7까지를 2회 반복 시행

단계 9:
동시변환 규칙 NL에 의해 CA를 1회 갱신

단계 10:
단계 5에서 단계7까지를 1회 시행

단계 11:
단계 9에서 단계 10을 총 2회 반복 시행

단계 12:
M[i]와 M[i+1]을 각각 왼쪽으로 16비트, 왼쪽으로 24비트 로테이션한 값으로 갱신한 결과를 아래와 같이 셀룰라 공간 SP에 입력

C_1 ← $M[i]_3$, C_7 ← $M[i+1]_3$,
C_{10} ← $M[i]_2$, C_{16} ← $M[i+1]_2$,
C_{19} ← $M[i]_1$, C_{25} ← $M[i+1]_1$,
C_{28} ← $M[i]_0$, C_{34} ← $M[i+1]_0$.

단계 13:
단계 4에서 단계11까지를 1회 반복 시행

단계 14:
i ←i+2 , j←j+1

단계 15:
단계4에서 단계 14까지를 i=16이 될 때까지 반복

단계 16:
AA← C_0 ‖ C_9 ‖ C_{18} ‖ C_{27}
BB← C_2 ‖ C_{11} ‖ C_{20} ‖ C_{29}
CC← C_4 ‖ C_{13} ‖ C_{21} ‖ C_{30}
DD← C_6 ‖ C_{15} ‖ C_{24} ‖ C_{33}
EE← C_8 ‖ C_{17} ‖ C_{26} ‖ C_{25}

단계 17:
AA, BB, CC, DD, EE를 해쉬 값으로 출력

단계 18:
끝

최초의 512비트 메시지 블록을 처리할 때는 A=0×9e3779b9, B=0×3c6ef373, C=0×78dde6e6, D=0×f1bbcdcc, E=0×e3779b99 으로 초기화 된 값을 사용하며, 다음 512비트 블록을 처리할 때는 앞 단계의 해쉬 출력 값을 사용한다.

본 논문에서 제안된 2차원 셀룰라 오토마타는 기존의 셀룰라 오토마타와는 다음 상태 전이 함수를 구성하는 방법에 있어서 기존의 2차원 셀룰라 오토마타들과는 차별적이다. 기존의 2차원 셀룰라 오토마타들은 다음 상태전이 함수로 로테이션을 사용하지 않는다. 또한 제안된 해쉬 함수는 기존의 셀룰라 오토마타를 이용한 해쉬 함수와는 다른 것으로 제안된 해쉬 함수는 전혀 새로운 개념의 것이라 할 수 있다[1,7]. 제안된 해쉬 함수의 수행속도는 Pentium MMX 200MHz(64M RAM, Windows 98) 환경에서 약 14.02Mbits/sec로 측정되었다. 이러한 속도는 유사한 플랫폼에서 수행된 SHA-1 및 HAS160 수준의 속도라고 볼 수 있다[2]. 또한 이 속도는 알고리즘의 최적화가 이루어지지 않은 상태에서 측정된 것인데, 알고리즘의 최적화에 의해 더욱 향상된 속도를 보일 수 있을 것으로 예측된다.

표 1. 해쉬 값의 통계테스트 검정 결과

검정 방법	기준치 (유의수준 5%)	측정치
빈도 검정	$\chi^2<3.8415$	$\chi^2=$ 1.0983
포커 검정 (m=4)	$\chi^2<24.9958$	$\chi^2=16.1594$
런 검정	$\chi^2<41.3371$	$\chi^2=19.3156$
롱런 검정	i<34 (i:최대런의길이)	i=14.6182
계열 검정	$\chi^2<5.9915$	$\chi^2=$ 2.1363
자기상관 검정	$\chi^2<5.9915$	$\chi^2=$ 3.2256

표 1은 제안된 해쉬 함수의 출력 난수열에 대한 빈도 검정, 포커 검정, 런 검정, 롱런 검정, 계열 검정, 자기상관 검정 등의 검정들을 시행한 결과를 나타낸 것인데, 모든 경우에서 통과함을 알 수 있다. 이와 같은 검정 결과들로부터, 제안된 해쉬 함수의 출력 난수열의 임의성이 우수함을 알 수 있다.

6. 결 론

본 논문에서는 새로운 구조의 2차원 셀룰라 오토마타를 제안하고, 제안된 2차원 셀룰라 오토마타를 이용한 해쉬 함수를 제안하였다. 제안된 해쉬 함수의 수행속도는 기존의 해쉬 함수와 비교하여 유사하거나 조금은 빠른 편이나 셀룰라 오토마타의 특성상 하드웨어로 구현할 경우 더욱 빠른 수행속도를 기대할 수 있다. 또한 제안된 해쉬 함수로부터 생성된 해쉬 값들의 난수적 특성을 조사하기 위하여 몇 가지 통계 테스트들을 수행하였다. 이러한 테스트의 결과 제안된 해쉬 함수로부터 생성된 해쉬 값들은 좋은 난수적 특성들을 가지고 있다는 것을 알 수 있었다.

참 고 문 헌

[1] 이경현, "비밀성 메카니즘 분석 및 평가 기법 연구," 한국정보보호센터, 1998.
[2] "해쉬함수표준 - 제2부 : 해쉬함수알고리즘 (HAS-160)", 한국정보통신기술협회 1998.11
[3] S. Wolfram, "Random sequence generation by cellular automata," *Advances in Applied Math.* 7, 123-169, 1986.
[4] S. Wolfram, "Cryptography with cellular automata," *Advances in Cryptology: Proceedings of CRYPTO '85, Lecture Notes in Computer Science* 218, 429-432, 1986.
[5] P. Chaudhuri, D. Chowdhury, S. Nandi, and S. Chattopadhyay, "Additive cellular automata theory and applications," *IEEE Computer Society Press, Los Alamitos*, California, 1997.
[6] W. Meier and O. Staffelbach, "Analysis of

pseudo random sequences generated by cellular automata," *Advances in Cryptology in Computer Science* 547, 186-199, 1992.

[7] 신상욱, 윤재우, 이경현 "셀룰러 오토마타에 기반한 안전한 해쉬함수" 한국멀티미디어논문지 1998, 12.

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

김 재 겸

1981년 고려대학교 수학과 학사
1983년 고려대학교 수학과 석사
1987년 고려대학교 수학과 박사
1989년~현재 경성대학교 수학과 전임강사, 조교수, 부교수, 교수

관심분야 : 보안 알고리즘