



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2008-0025354
(43) 공개일자 2008년03월20일

(51) Int. Cl.

G06F 17/10 (2006.01) G06F 9/06 (2006.01)

(21) 출원번호 10-2007-0094428

(22) 출원일자 2007년09월17일

심사청구일자 2007년09월17일

(30) 우선권주장

1020060089456 2006년09월15일 대한민국(KR)

(71) 출원인

고려대학교 산학협력단

서울 성북구 안암동5가1 고려대학교 내

(72) 발명자

홍득조

경기 용인시 수지구 죽전2동 동성1차아파트 102동 1303호

장동훈

서울 강서구 화곡7동 1082-21

(뒷면에 계속)

(74) 대리인

김동진, 강경찬, 박경찬

전체 청구항 수 : 총 6 항

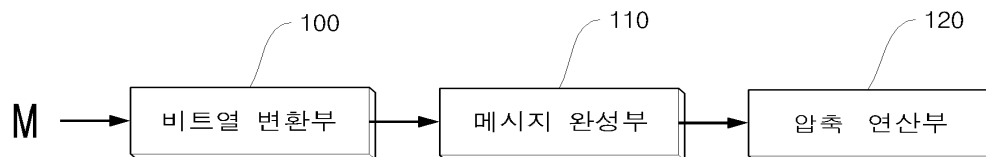
(54) 256비트 출력을 갖는 해쉬 알고리즘

(57) 요약

본 발명은 그 구현에 있어서 효율성이 크게 향상되며, 강한 안전성을 갖는 256비트 출력을 갖는 해쉬 알고리즘에 관한 것이다.

본 발명에 따르면, 임의의 길이의 메시지 비트열을 입력받아 워드열로 변환하는 단계, 상기 입력된 메시지를 기 설정된 비트로 변환하는 단계 및 상기 기설정된 비트로 변환된 메시지를 입력받아 압축을 수행하는 단계를 포함한다.

대표도 - 도1



(72) 발명자

성재철

서울 강북구 수유2동 벽산아파트 3동 1201호

이상진

서울 노원구 중계본동 136 대림벽산아파트 103동
1102호

홍석희

서울 은평구 응암2동 451-12

이제상

경기 수원시 영통구 영통동 신나무실 주공아파트
518동 604호

특허청구의 범위

청구항 1

임의의 길이의 메시지 비트열을 입력받아 워드열로 변환하는 단계;

상기 입력된 메시지를 기설정된 비트로 변환하는 단계; 및

상기 기설정된 비트로 변환된 메시지를 입력받아서 압축 연산을 수행하는 단계;를 포함하는 256비트 출력을 갖는 해쉬 알고리즘.

청구항 2

제 1항에 있어서, 상기 워드열로 변환하는 단계는;

임의의 길이 메시지의 비트열에서 32비트열을 4바이트의 문자열로 보고, 첫번째 바이트가 워드의 최하위 바이트가 되며, 첫번째 4바이트를 첫번째 워드로 두번째 4바이트를 두번째 워드로 변환하는 것을 특징으로 하는 256비트 출력을 갖는 해쉬 알고리즘.

청구항 3

제 1항에 있어서, 상기 기설정된 비트로 변환하는 단계는;

512비트로 변환하는 것을 특징으로 하는 256비트 출력을 갖는 해쉬 알고리즘.

청구항 4

제 3항에 있어서, 상기 512비트로의 변환은;

메시지의 길이가 512비트가 아닌경우에, 메시지의 길이가 448비트가 될때까지 입력된 메시지에 0을 채우고, 마지막 64비트는 메시지의 비트열의 길이로 표현

되는 것을 특징으로 하는 256비트 출력을 갖는 해쉬 알고리즘.

청구항 5

제 1항에 있어서, 상기 압축 연산을 수행하는 단계는;

일련의 메시지와 제 0 연쇄변수를 입력받아 입력된 메시지를 압축 연산을 수행하는 것을 특징으로 하는 256비트 출력을 갖는 해쉬 알고리즘.

청구항 6

제 5항에 있어서, 상기 압축 연산의 수행은 복수의 압축 함수부가 수행하며, 상기 각 압축 함수부는

제 i 연쇄변수 및 분할된 메시지를 입력받아서 브랜치 함수부에서 연산하는 단계;

이웃하는 두 브랜치 함수부에서 출력된 서브 연쇄변수들간에 덧셈 연산을 수행하는 단계;

상기 덧셈 연산에서 출력된 값의 배타적 논리합 연산을 수행하는 단계; 및

상기 배타적 논리합 연산에서 출력된 값과 상기 제 i 연쇄변수와의 덧셈연산을 수행하는 단계;를 포함하는 것을 특징으로 하는 256비트 출력을 갖는 해쉬 알고리즘.

명 세 서

발명의 상세한 설명

기술 분야

<1> 본 발명은 새로운 256비트 출력을 갖는 해쉬 알고리즘에 관한 것으로, 특히 고속 소프트웨어 환경에 적합하고 128비트 이상의 안전도를 갖는 256비트 출력을 갖는 해쉬 알고리즘에 관한 것이다.

배경 기술

- <2> 최근 미국 국립기술표준원에서는 256비트 이상의 출력크기를 갖는 SHA-256, 384, 512 알고리즘을 표준으로 제정하였다. 이는 컴퓨팅 파워의 증가로 인해 160비트 출력 크기를 갖는 알고리즘을 대체할 필요성이 대두되었기 때문이다. 그러나 국내 표준 해쉬 함수 알고리즘인 HAS-160의 경우, 160비트의 출력값을 갖기 때문에 국제 수준에서 요구되는 안전성을 제공하지 못한다. 따라서 고속 소프트웨어 환경에 적합한 256비트 출력을 갖는 해쉬 함수 알고리즘의 설계는 이 분야의 연구 초점이라 할 수 있다.

발명의 내용

해결 하고자하는 과제

- <3> 그러므로 본 발명의 목적은 256비트의 출력값을 갖고, 기존 공격에 대한 안전성을 제공하면서, 미국 표준 해쉬 함수 알고리즘인 SHA-256 이상의 효율성을 가지는 256비트 출력을 갖는 해쉬 알고리즘을 제공하는 데 있다.

과제 해결수단

- <4> 이러한 목적을 이루기 위한 본 발명의 256비트 출력을 갖는 해쉬 알고리즘은 임의의 길이의 메시지 비트열을 입력받아 워드열로 변환하는 단계, 상기 입력된 메시지를 기설정된 비트로 변환하는 단계 및 상기 기설정된 비트로 변환된 메시지를 입력받아서 압축연산을 수행하는 단계를 포함한다.
- <5> 상기 압축 연산의 수행은 복수의 압축 함수부가 수행하며, 상기 각 압축 함수부는 제 i 연쇄변수 및 분할된 메시지를 입력받아서 브랜치 함수부에서 연산하는 단계, 이웃하는 두 브랜치 함수부에서 출력된 서브 연쇄변수들 간에 덧셈 연산을 수행하는 단계, 상기 덧셈 연산에서 출력된 값의 배타적 논리합 연산을 수행하는 단계 및 상기 배타적 논리합 연산에서 출력된 값과 상기 제 i 연쇄변수와 덧셈연산을 수행하는 단계를 포함한다.

효 과

- <6> 본 발명의 256비트 출력을 갖는 해쉬 알고리즘은 그 구현에 있어서 효율성이 크게 향상되며, 강한 안전성을 갖게 된다.

발명의 실시를 위한 구체적인 내용

- <7> 이하 첨부된 도면을 참조하여 본 발명의 256비트 출력을 갖는 해쉬 알고리즘을 상세히 설명한다.
- <8> 도 1은 본 발명의 해쉬 함수 압축 장치의 구성을 나타낸 블록도이다. 도시된 바와 같이, 본 발명의 해쉬 함수 압축 장치는 비트열 변환부(100), 메시지 완성부(110) 및 압축 연산부(120)를 포함한다.
- <9> 상기 비트열 변환부(100)는 임의의 길이 메시지의 비트열을 입력받아 워드열로 변환한다. 비트열 변환부(100)는 임의의 길이 메시지의 비트열에서 32비트 비트열을 4바이트의 문자열로 보고, 첫번째 바이트가 워드의 최하위 바이트가 되며, 첫번째 4바이트를 첫번째 워드로 두번째 4바이트를 두 번째 워드로 변환하는 과정을 반복하여 입력받은 메시지의 비트열을 워드열로 변환한다.
- <10> 예를들면, 비트열 연산부는 입력되는 메시지가 10100010 0011101 11100101 01101011 11001001 10001010 10011101이라 한다. 상기 입력된 메시지를 워드로 변환하면 첫번째 4바이트는 6be539a2가 되며, 두번째 4바이트는 809d8ac9이 된다. 비트열 변환부(100)는 4바이트를 하나의 그룹으로 연산이 이루어지는데, 입력되는 메시지는 7바이트로 이루어져서, 하위 바이트가 4바이트로 형성되는 것이 아닌 3바이트로 형성된다. 비트열 변환부(100)는 하위 바이트가 4바이트로 형성되지 않은 경우에, 4바이트를 형성하기 위하여 100000...의 값으로 하위 바이트를 처리한다. 즉, 비트열 변환부(100)는 상기 입력되는 메시지의 최하위 3 바이트인 11001001 10001010 10011101에 10000000을 덧붙여서 4바이트를 형성한다. 그러므로 입력되는 메시지에 10000000이 덧붙여진 두번째 4바이트를 워드로 변환하는 경우에, 10000000의 워드로의 변환인 80이 덧붙여진다.
- <11> 메시지 완성부(110)는 입력되는 메시지를 기설정된 비트로 변환한다. 본 발명에서는 입력되는 메시지는 512비트 단위로 처리되므로, 입력되는 메시지가 512비트가 아닌 경우에 512비트로 변환시킨다. 메시지 완성부(110)는 입력되는 메시지가 512비트가 아닌 경우에, 메시지의 길이가 448비트가 될 때까지 입력된 메시지에 0을 채운다. 그리고 마지막 64비트에는 입력되는 메시지의 길이를 표현한다. 메시지의 길이도 역시 워드로 변환되어 표현되고, 마지막 64비트에 표현되는 메시지의 길이는 64비트 중에서 상위 32비트에 먼저 표현된다.

- <12> 상기 예에서와 같이 7바이트의 메시지가 입력되는 경우에, 메시지 완성부는 입력되는 메시지가 512비트가 아니므로, 메시지의 길이가 448비트가 될때까지 입력된 메시지에 0을 채운다. 즉, 입력된 메시지 뒤에 96개의 0이 채워진다. 그리고 마지막 64비트는 입력된 비트열의 길이로 표현되는데, 입력된 메시지의 길이가 56=111000이므로 이를 워드로 변환한 38이 마지막 64비트에 표현된다. 상기 38은 64비트에서 상위 32비트에 표현된다. 즉, 마지막 64비트는 00000038 00000000로 표현된다.
- <13> 압축 연산부(120)는 상기 메시지 완성부(110)에서 기설정된 비트로 변환된 메시지를 입력받아서 압축 연산을 한다.
- <14> 도 2는 도 1의 메시지가 압축 연산부에 의하여 압축되는 것을 보인 도면이다. 도시된 바와 같이, 압축 연산부(120)의 일련의 메시지(M[0] || M[1] || M[2] || ...)와 제 0 연쇄 변수(CV₀)를 입력받아 입력된 메시지를 처리한다. 압축 연산부(120)는 복수의 압축 함수부(200)로 구성되며, 상기 각 압축 함수부(200)는 512비트 단위로 메시지를 처리하므로, 각 메시지들(M[0], M[1], M[2]...)은 그 길이가 512비트이다.
- <15> 도 3은 도 2의 압축 함수부의 구성을 보인 도면이다. 도시된 바와 같이, 압축 함수부는 제 i 연쇄변수(CV_i) 및 분할된 메시지를 입력받아서 연산하는 복수의 브랜치 함수부(300, 301, 302, 303), 이웃하는 두 브랜치 함수부(300, 301, 302, 303)에서 출력된 서브 연쇄변수들간에 덧셈연산을 수행하는 복수의 제 1 덧셈 연산부(310, 311), 상기 각 제 1 덧셈 연산부(310, 311)에서 출력된 값의 배타적 논리합 연산을 수행하는 배타적 논리합 연산부(320) 및 상기 배타적 논리합 연산부(320)에서 연산된 출력과 상기 압축 함수부로 입력되는 제 i 연쇄변수(CV_i)와의 덧셈연산을 수행하는 제 2 덧셈 연산부(330)를 포함한다.
- <16> 한편, 상기 메시지 완성부에서 기설정된 비트로 변환된 메시지는 압축 함수부에 입력되어 32비트 서브 메시지 워드(m_i[0], m_i[1], m_i[2] ..., m_i[15])로 분할된다. 압축 함수부는 512비트의 메시지를 입력받게 되므로, 32비트 서브 메시지 워드는 모두 16개가 생성된다.
- <17> 도 4는 브랜치 함수부의 연산의 흐름을 보인 도면이다. 도시된 바와 같이, 복수의 각 j(1≤j≤4) 브랜치 함수부(300, 301, 302, 303)는 복수의 제 1 서브 연쇄변수(R_j⁽⁰⁾[t], 0≤t≤7)를 입력받고, 소정의 연산을 통해 제 8 서브 연쇄변수(R_j⁽⁸⁾[t])를 출력한다. 각 연쇄변수를 32비트로 구성된다. 각 제 s 서브 연쇄변수(R_j^(s)[t])는 8개로 구성된다. 제 1 서브 연쇄변수는 (R_j⁽⁰⁾[t])는 제 i 연쇄변수(CV_i)와 같다.
- <18> 제 s+1의 1번째 서브 연쇄변수(R_j^(s+1)[0])는 제 s의 5번째 서브 연쇄변수(R_j^(s)[4])와 상기 분할된 서브 메시지 중에서 제 j의 4s+1번째 서브 메시지(m_i[σ_j(4s+1)])와 덧셈연산을 수행하고, 상기 덧셈연산의 출력과 제 j의 2s+1번째 상수(δ[p_j(2s+1)])와 논리적 배타합 연산을 수행하며, 상기 논리적 배타합 연산의 출력과 제 j의 4s+3번째 서브 메시지(m_i[σ_j(4s+3)])와 덧셈연산을 수행하고, 상기 덧셈연산의 출력을 변수로 하여 제 1 함수연산(f(x))을 수행하며, 제 1 함수연산(f(x))의 출력을 8비트만큼 왼쪽으로 순환 이동연산을 행하고, 상기 순환이동연산의 출력과 제 s의 8번째 서브 연쇄변수(R_j^(s)[7])과의 덧셈연산을 수행한 값이 된다. 상기 제 1 함수연산은 하기의 수학식 1과 같다.

수학식 1

<19>
$$f(x) = x \oplus x \ll 15 \oplus x \ll 27$$

- <20> 상기 제 1 함수는 입력 변수(x)와 입력 변수를 15비트만큼 왼쪽으로 순환이동시킨 값(x^{≪15})과의 논리적 배타합 연산을 수행하고, 상기 논리적 배타합 연산의 출력과 입력 변수를 27비트만큼 왼쪽으로 순환이동시킨 값(x^{≪27})과의 논리적 배타합 연산을 수행한다.

- <21> 제 s+1의 2번째 서브 연쇄변수(R_j^(s+1)[1])는 제 s의 1번째 서브 연쇄변수(R_j^(s)[0])와 제 j의 4s번째 서브 메시지(m_i[σ_j(4s)])와 덧셈연산을 수행하고, 상기 덧셈연산의 출력과 제 j의 2s번째 상수(δ[p_j(2s+1)])와의 논리적

배타합 연산을 수행하며, 상기 논리적 배타합 연산의 출력과 제 j의 4s+2번째 서브 메시지($m_i[\sigma_j(4s+2)]$)와의 덧셈연산을 수행한 값이 된다.

<22> 제 s+1의 3번째 서브 연쇄변수($R_j^{(s+1)}[2]$)는 제 s의 1번째 서브 연쇄변수($R_j^{(s)}[0]$)와 제 j의 4s번째 서브 메시지($m_i[\sigma_j(4s)]$)와 덧셈연산을 수행하고, 상기 덧셈연산의 출력을 변수로 하여 제 1 함수연산을 수행하고, 상기 제 1 함수연산의 출력과 제 s의 2번째 서브 연쇄변수($R_j^{(s)}[1]$)와의 덧셈연산을 수행한 값이 된다.

<23> 제 s+1의 4번째 서브 연쇄변수($R_j^{(s+1)}[3]$)는 제 s의 1번째 서브 연쇄변수($R_j^{(s)}[0]$)와 제 j의 4s번째 서브 메시지($m_i[\sigma_j(4s)]$)와 덧셈연산을 수행하고, 상기 덧셈연산의 출력을 변수로 하여 제 1 함수연산을 수행하고, 상기 제 1 함수연산의 출력을 13비트만큼 왼쪽으로 순환 이동연산을 행하고, 상기 순환 이동연산의 출력과 제 s의 3번째 서브 연쇄변수($R_j^{(s)}[2]$)와 덧셈연산을 수행한다. 상기 덧셈연산의 출력값을 제 1 출력값이라 한다. 그리고 제 s의 1번째 연쇄변수($R_j^{(s)}[0]$)와 제 j의 4s번째 서브 메시지($m_i[\sigma_j(4s)]$)와의 덧셈연산을 수행하고, 상기 덧셈연산의 출력과 제 j의 2s번째 상수($\delta[p_j(2s)]$)와 논리적 배타합 연산을 수행하고, 상기 논리적 배타합 연산의 출력과 제 j의 4s+2번째 서브 메시지($m_i[\sigma_j(4s+2)]$)와 덧셈연산을 수행하고, 상기 덧셈연산의 출력을 변수로 하여 제 2 함수연산($g(x)$)을 수행한다. 상기 제 2 함수연산의 출력과 상기 제 1 출력값과의 배타적 논리합 연산을 수행한 값이 제 s+1의 4번째 서브 연쇄변수($R_j^{(s+1)}[3]$)가 된다. 상기 제 2 함수연산은 하기의 수학식 2와 같다.

수학식 2

$$g(x) = x \oplus (x \ll 7 + x \ll 25)$$

<24>

<25> 상기 제 2 함수는 입력 변수를 7비트만큼 왼쪽으로 순환이동시킨 값($x \ll 7$)과 입력 변수를 25비트만큼 왼쪽으로 순환이동시킨 값($x \ll 25$)과의 덧셈연산을 수행하고, 상기 덧셈연산의 출력과 입력 변수(x)와의 논리적 배타합 연산 수행한 값이다.

<26> 제 s+1의 5번째 서브 연쇄변수($R_j^{(s+1)}[4]$)는 제 s의 1번째 서브 연쇄변수($R_j^{(s)}[0]$)와 제 j의 4s번째 서브 메시지($m_i[\sigma_j(4s)]$)와의 덧셈연산을 수행하고, 상기 덧셈연산의 출력과 제 j의 2s번째 상수($\delta[p_j(2s)]$)와 배타적 논리합 연산을 수행하며, 상기 배타적 논리합 연산을 제 j의 4s+2번째 서브 메시지($m_i[\sigma_j(4s+2)]$)와 덧셈연산을 수행하고, 상기 덧셈연산의 출력을 변수로 하여 제 2 함수연산을 수행하고, 상기 제 2 함수연산의 출력을 17비트만큼 왼쪽으로 순환 이동연산을 행하고, 상기 순환 이동연산의 출력을 제 s의 4번째 서브 연쇄변수($R_j^{(s)}[3]$)와 논리적 배타합 연산을 수행한 값이 된다.

<27> 제 s+1의 서브 6번째 연쇄변수($R_j^{(s+1)}[5]$)는 제 s의 5번째 서브 연쇄변수($R_j^{(s)}[4]$)와 제 j의 4s+1번째 서브 메시지($m_i[\sigma_j(4s+1)]$)와 덧셈연산을 수행하고, 상기 덧셈연산의 출력을 제 j의 2s+1번째 상수($\delta[p_j(2s+1)]$)와 논리적 배타합 연산을 수행하고, 상기 논리적 배타합 연산의 출력을 제 j의 4s+3번째 서브 메시지($m_i[\sigma_j(4s+3)]$)와 덧셈연산을 수행한 값이 된다.

<28> 제 s+1의 7번째 서브 연쇄변수($R_j^{(s+1)}[6]$)는 제 s의 5번째 서브 연쇄변수($R_j^{(s)}[4]$)와 제 j의 4s+1번째 서브 메시지($m_i[\sigma_j(4s+1)]$)와 덧셈연산을 수행하고, 상기 덧셈연산의 출력을 변수로 하여 제 2 함수연산을 수행하며, 상기 제 2 함수연산의 출력과 제 s의 6번째 서브 연쇄변수($R_j^{(s)}[5]$)와의 덧셈연산을 수행한다.

<29> 제 s+1의 8번째 서브 연쇄변수($R_j^{(s+1)}[7]$)는 제 s의 5번째 서브 연쇄변수($R_j^{(s)}[4]$)와 제 j의 4s+1번째 서브 메시지($m_i[\sigma_j(4s+1)]$)와 덧셈연산을 수행하고, 상기 덧셈연산의 출력을 변수로 하여 제 2 함수연산을 수행하고, 상

기 제 2 함수연산의 출력을 3비트만큼 왼쪽으로 순환 이동연산을 행하고, 상기 순환 이동연산의 출력과 제 s의 7번째 서브 연쇄변수($R_j^{(s)}[6]$)와 덧셈연산을 수행한다. 상기 덧셈연산의 출력값을 제 2 출력값이라 한다. 그리고 제 s의 5번째 서브 연쇄변수($R_j^{(s)}[4]$)와 제 j의 4s+1번째 서브 메시지($m[\sigma_j(4s+1)]$)와의 덧셈연산을 수행하고, 상기 덧셈연산의 출력과 제 j의 2s+1번째 상수($\delta[\rho_j(2s+1)]$)와 배타적 논리합 연산을 수행하며, 상기 배타적 논리합 연산의 출력과 제 j의 4s+3번째 서브 메시지($m[\sigma_j(4s+3)]$)와 덧셈연산을 수행하고, 상기 덧셈연산의 출력을 변수로 하여 제 2 함수연산을 수행한다. 상기 제 1 함수연산의 출력과 상기 제 1 출력값과의 배타적 논리합 연산을 수행한 값이 제 s+1의 8번째 서브 연쇄변수($R_j^{(s+1)}[7]$)가 된다.

<30>

상기 브랜치 함수부(300, 301, 302, 303)에서의 제 s 서브 연쇄변수($R_j^{(s)}[t]$)와 제 s+1 서브 연쇄변수($R_j^{(s+1)}[t]$)와의 관계는 하기의 수학적 식 3과 같다.

수학적 식 3

$$\begin{aligned} R_j^{(s+1)}[0] &= R_j^{(s)}[7] \oplus f(((R_j^{(s)}[4] + M_i[\sigma_j(4s+1)]) \oplus \delta[\rho_j(2s+1)]) + M_i[\sigma_j(4s+3)])^{\ll 8}, \\ R_j^{(s+1)}[1] &= ((R_j^{(s)}[0] + M_i[\sigma_j(4s)]) \oplus \delta[\rho_j(2s)]) + M_i[\sigma_j(4s+2)], \\ R_j^{(s+1)}[2] &= R_j^{(s)}[1] + f(R_j^{(s)}[0] + M_i[\sigma_j(4s)]), \\ R_j^{(s+1)}[3] &= R_j^{(s)}[2] + f(R_j^{(s)}[0] + M_i[\sigma_j(4s)])^{\ll 13} \\ &\quad \oplus g(((R_j^{(s)}[0] + M_i[\sigma_j(4s)]) \oplus \delta[\rho_j(2s)]) + M_i[\sigma_j(4s+2)]), \\ R_j^{(s+1)}[4] &= R_j^{(s)}[3] \oplus g(((R_j^{(s)}[0] + M_i[\sigma_j(4s)]) \oplus \delta[\rho_j(2s)]) + M_i[\sigma_j(4s+2)])^{\ll 17}, \\ R_j^{(s+1)}[5] &= ((R_j^{(s)}[4] + M_i[\sigma_j(4s+1)]) \oplus \delta[\rho_j(2s+1)]) + M_i[\sigma_j(4s+3)], \\ R_j^{(s+1)}[6] &= R_j^{(s)}[5] + g(R_j^{(s)}[4] + M_i[\sigma_j(4s+1)]), \\ R_j^{(s+1)}[7] &= R_j^{(s)}[6] + f(R_j^{(s)}[4] + M_i[\sigma_j(4s+1)])^{\ll 3} \\ &\quad \oplus g(((R_j^{(s)}[4] + M_i[\sigma_j(4s+1)]) \oplus \delta[\rho_j(2s+1)]) + M_i[\sigma_j(4s+3)]) \end{aligned}$$

<31>

<32>

상기 브랜치 함수부(300, 301, 302, 303)가 서브 메시지를 입력받아 연산을 수행하는 데 있어서, 각 브랜치 함수부(300, 301, 302, 303)마다 입력되는 서브 메시지의 순서는 상이하다. 도 5는 브랜치 함수부마다 입력되는 서브 메시지의 순서를 나타낸 도면이다. 제 1 브랜치 함수부(300)에 입력되는 서브 메시지는 제 1의 0번째 서브 메시지($m[\sigma_j(0)]$)는 $m[0]$ 가 되고, 제 1의 1번째 서브 메시지는 $m[1]$ 이고, 제 1의 2번째 서브 메시지는 $m[2]$ 이고, 제 1의 3번째 서브 메시지는 $m[3]$ 이다. 그러나 제 2 브랜치 함수부(301)에 입력되는 메시지의 순서는 상기 제 1 브랜치 함수부(300)에 입력되는 메시지와 달리 제 2의 0번째 서브 메시지는 $m[12]$ 이고, 제 2의 1번째 서브 메시지는 $m[0]$ 이고, 제 2의 2번째 서브 메시지는 $m[4]$ 이고, 제 2의 3번째 서브 메시지는 $m[8]$ 이다. 제 3 브랜치 함수부(302)와 제 4 브랜치 함수부(303)의 서브 메시지의 순서 역시 도시된 바와 같이 타 브랜치 함수부의 서브 메시지의 순서와 상이하다.

<33>

그리고 상기 브랜치 함수부(300, 301, 302, 303)는 상수를 입력받게 되는데, 상수의 값은 도 6에 도시된다. 다만, 상수 역시 브랜치 함수부마다 입력되는 순서가 상이하다. 도 7은 브랜치 함수부마다 입력되는 상수의 순서를 나타낸 도면이다. 제 1 브랜치 함수부(300)는 제 1의 0번째 상수가 $\delta[0]$ 이고, 제 1의 1번째 상수가 $\delta[1]$ 이고, 제 1의 2번째 상수가 $\delta[2]$ 이고, 제 1의 3번째 상수가 $\delta[3]$ 이다. 그러나 제 2 브랜치 함수부(301)에 입력되는 상수는 제 2의 0번째 상수가 $\delta[3]$ 이고, 제 2의 1번째 상수가 $\delta[6]$ 이고, 제 2의 2번째 상수가 $\delta[9]$ 이고, 제 3의 3번째 상수가 $\delta[12]$ 이다. 제 3 브랜치 함수부(302)와 제 4 브랜치 함수부(303)의 상수의 순서 역시 도시된 바와 같이 타 브랜치 함수부의 서브 메시지의 순서와 상이하다.

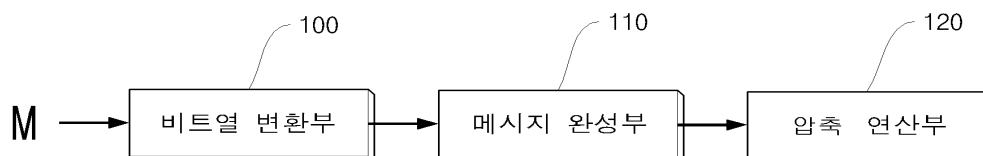
- <34> 상기 복수의 각 제 j 브랜치 함수부(300, 301, 302, 303)는 복수의 제 1 서브 연쇄변수($R_j^{(1)}[t]$)를 입력받아서 복수의 제 2 서브 연쇄변수($R_j^{(2)}[t]$)를 출력하고, 상기 출력된 복수의 제 2 서브 연쇄변수($R_j^{(2)}[t]$)를 입력으로 하여 복수의 제 3 서브 연쇄변수($R_j^{(3)}[t]$)를 출력하는 연산을 계속 반복한다. 제 j 브랜치 함수부는 상기 연산을 8번 수행한다. 즉, 제 j 브랜치 함수부가 최종적으로 출력하는 연쇄변수는 제 8 서브 연쇄변수($R_j^{(8)}[t]$)가 된다.
- <35> 상기 복수의 제 1 덧셈연산부(310, 311)는 이웃하는 브랜치 함수부(300, 301, 302, 303)에서 출력된 제 8 서브 연쇄변수($R_1^{(8)}[t]$ 와 $R_2^{(8)}[t]$, $R_3^{(8)}[t]$ 와 $R_4^{(8)}[t]$)간에 덧셈연산을 수행한다.
- <36> 배타적 논리합 연산부(320)는 상기 제 1 브랜치 함수부(300)와 제 2 브랜치 함수부(301)에서 출력된 제 8 서브 연쇄변수($R_1^{(8)}[t]$, $R_2^{(8)}[t]$)간에 덧셈연산의 출력과 제 3 브랜치 함수부(302)와 제 4 브랜치 함수부(304)에서 출력된 제 8 서브 연쇄변수($R_3^{(8)}[t]$, $R_4^{(8)}[t]$)간에 덧셈연산의 출력간에 배타적 논리합 연산을 수행한다.
- <37> 상기 배타적 논리합 연산부(320)의 출력은 입력되는 제 i 연쇄변수($CV_i[t]$)와 덧셈연산을 수행한다. 상기 덧셈연산의 출력은 제 i+1 연쇄변수($CV_{i+1}[t]$)가 된다. 즉, 압축 함수부의 출력은 제 i+1 연쇄변수($CV_{i+1}[t]$)가 된다.
- <38> 한편, 제 1 연쇄변수($CV_1[t]$)를 출력하기 위해서는 제 0 연쇄변수($CV_0[t]$)가 요구된다. 그러므로 제 0 연쇄변수의 값은 그림 8에 도시된 값을 갖는다.
- <39> 본 발명은 기재된 구체적인 예에 대해서만 상세히 설명되었지만 본 발명의 기술사상 범위 내에서 다양한 변형 및 수정이 가능함은 당업자에게 있어서 명백한 것이며, 이러한 변형 및 수정이 첨부된 특허청구범위에 속함은 당연한 것이다.

도면의 간단한 설명

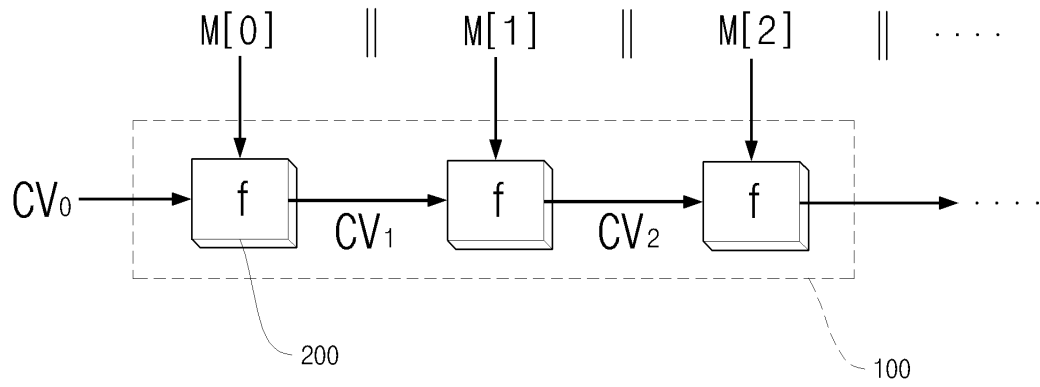
- <40> 도 1은 본 발명의 해쉬 함수 압축 장치의 구성을 나타낸 블록도.
- <41> 도 2는 도 1의 메시지가 압축 연산부에 의하여 압축되는 것을 보인 도면.
- <42> 도 3은 도 2의 압축 함수부의 구성을 보인 도면.
- <43> 도 4는 브랜치 함수부의 연산의 흐름을 보인 도면.
- <44> 도 5는 브랜치 함수부마다 입력되는 서브 메시지의 순서를 나타낸 도면.
- <45> 도 6은 브랜치 함수부로 입력되는 상수의 값을 나타낸 도면.
- <46> 도 7은 브랜치 함수부마다 입력되는 상수의 순서를 나타낸 도면.
- <47> 도 8은 제 0 연쇄변수의 값을 나타낸 도면.

도면

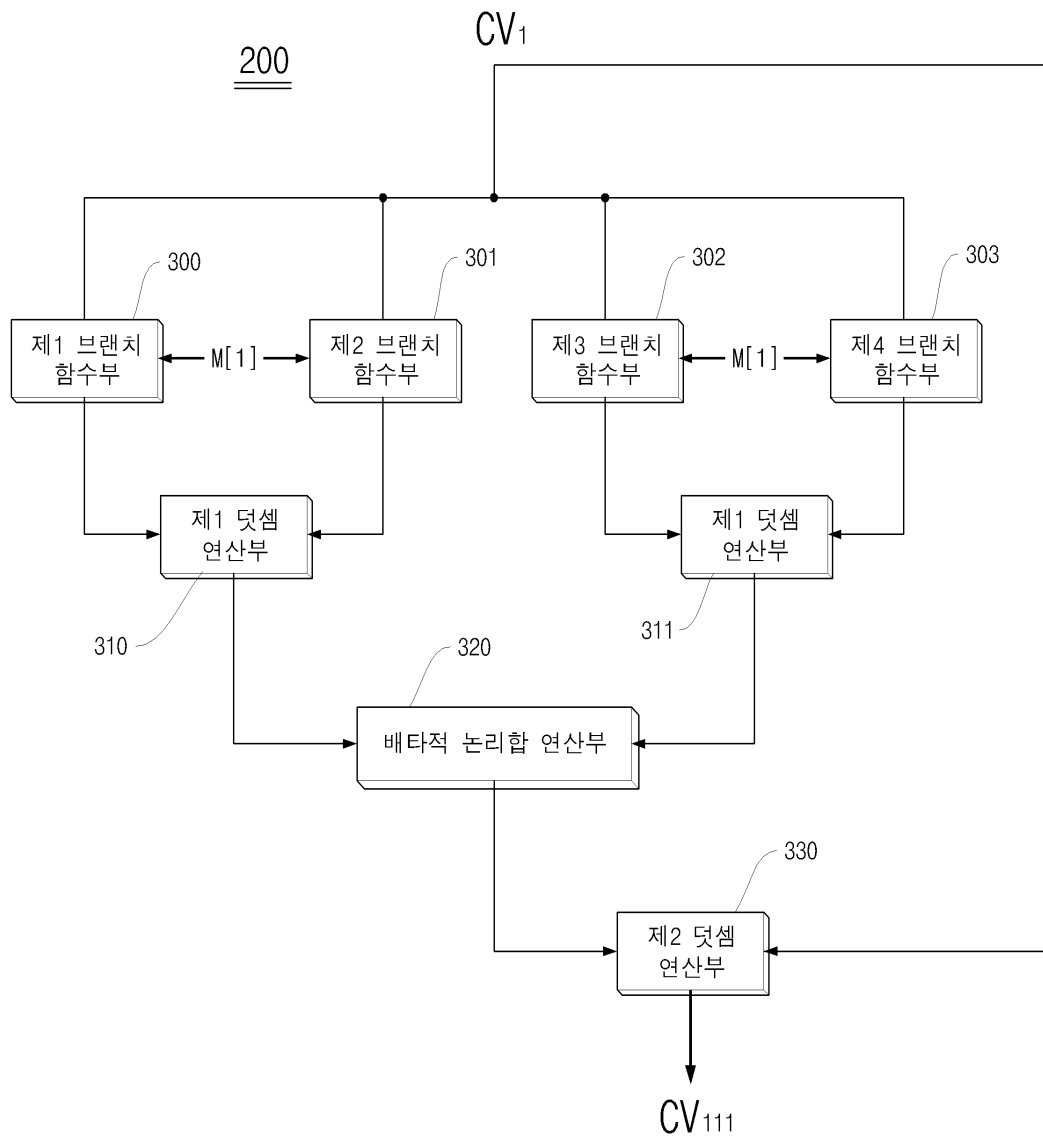
도면1



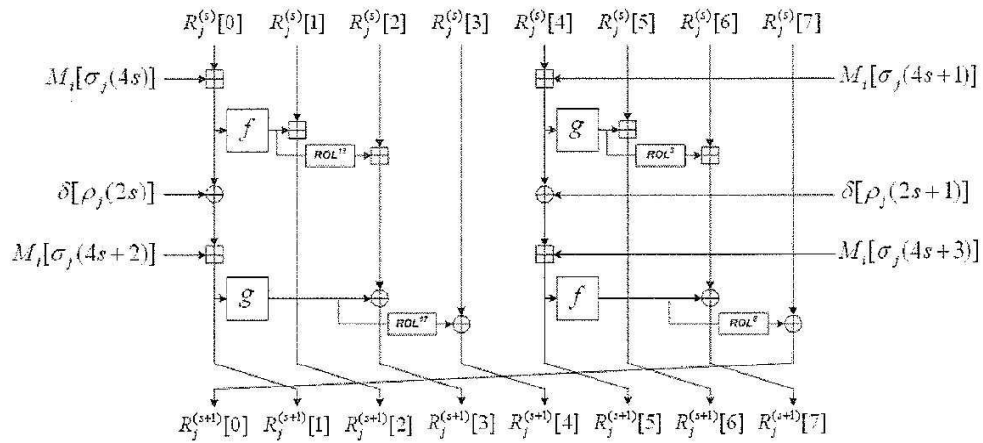
도면2



도면3



도면4



도면5

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\sigma_1(x)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$\sigma_1(x)$	9	12	15	2	5	8	11	14	1	4	7	10	13	0	3	6
x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\sigma_2(x)$	12	0	4	8	13	1	5	9	14	2	6	10	15	3	7	11
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$\sigma_2(x)$	8	15	6	13	4	11	2	9	0	7	14	5	12	3	10	1
x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\sigma_3(x)$	3	8	13	2	7	12	1	6	11	0	5	10	15	4	9	14
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$\sigma_3(x)$	6	1	12	7	2	13	8	3	14	9	4	15	10	5	0	11
x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\sigma_4(x)$	13	10	7	4	1	14	11	8	5	2	15	12	9	6	3	0
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$\sigma_4(x)$	14	7	0	9	2	11	4	13	6	15	8	1	10	3	12	5

도면6

$\delta[0] = 0x428a2f98$	$\delta[1] = 0x71374491$
$\delta[2] = 0xb5c0fbcf$	$\delta[3] = 0xe9b5dba5$
$\delta[4] = 0x3956c25b$	$\delta[5] = 0x59f111f1$
$\delta[6] = 0x923f82a4$	$\delta[7] = 0xab1c5ed5$
$\delta[8] = 0xa807aa98$	$\delta[9] = 0x12835b01$
$\delta[10] = 0x243185be$	$\delta[11] = 0x550c7dc3$
$\delta[12] = 0x72be5d74$	$\delta[13] = 0x80deb1fe$
$\delta[14] = 0x9bdc06a7$	$\delta[15] = 0xc19bf174$

도면7

s	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho_1(s)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho_2(s)$	3	6	9	12	15	2	5	8	11	14	1	4	7	10	13	0
$\rho_3(s)$	12	5	14	7	0	9	2	11	4	13	6	15	8	1	10	3
$\rho_4(s)$	7	2	13	8	3	14	9	4	15	10	5	0	11	1	6	12

도면8

$CV_0[0] = 0x6a09e667$	$CV_0[1] = 0xb67ae85$
$CV_0[2] = 0x3c6ef372$	$CV_0[3] = 0xa54ff53a$
$CV_0[4] = 0x510e527f$	$CV_0[5] = 0x9b05688c$
$CV_0[6] = 0x1f83d9ab$	$CV_0[7] = 0x5be0ca19$