(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2023년03월23일
(11) 등록번호 10-2513552
(24) 등록일자 2023년03월20일

(51) 국제특허분류(Int. Cl.)
H04L 9/00 (2022.01) *G06F 7/523* (2006.01)
H04L 9/06 (2006.01)
(52) CPC특허분류
H04L 9/008 (2013.01)
G06F 7/523 (2013.01)
(21) 출원번호 10-2022-0151584
(22) 출원일자 2022년11월14일
심사청구일자 2022년11월14일
(56) 선행기술조사문헌
KR1020090006147 A*
KR1020220094052 A*
Ryan Hayward, Chia-Chu Chiang "Parallelizing fully homomorphic encryption for a cloud environment." Journal of applied research and technology, Vol. 13(2), pp. 245-252(2015.)
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
수퍼게이트 주식회사
서울특별시 강남구 테헤란로 407, 11층(삼성동, 이케이타워)
(72) 발명자
권태경
서울특별시 관악구 관악로 1 서울대학교 301동 503호 (신림동)
박민경
서울특별시 관악구 관악로 1 서울대학교 301동 551-2호 (신림동)
(뒷면에 계속)
(74) 대리인
이장훈, 박정우

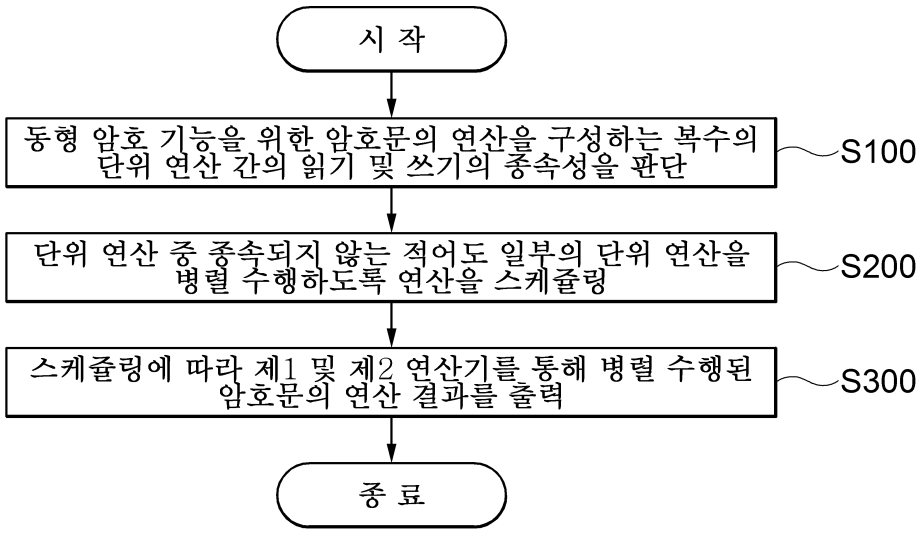
전체 청구항 수 : 총 5 항 심사관 : 강민성

(54) 발명의 명칭 **동형 암호 병렬 연산 방법 및 이를 수행하는 컴퓨팅 장치**

(57) 요 약

본 발명은 동형 암호의 연산 방법에 관한 것으로, 본 발명의 일 실시 예에 따른 컴퓨팅 장치에서 수행되는 컴퓨팅 장치에서 수행되는 병렬 동형 암호 연산 방법에 있어서, 동형 암호 기능을 위한 암호문의 연산을 구성하는 복수의 단위 연산 간의 읽기 및 쓰기의 종속성을 판단하는 단계; 상기 종속성에 따라 제1 연산기 상에서 수행되고 있는 상기 단위 연산 중 종속되지 않는 적어도 일부의 단위 연산을 제2 연산기 상에서 수행하도록 연산을 스케줄링하는 단계; 및 상기 스케줄링에 따라 제1 및 제2 연산기를 통해 병렬 수행된 암호문의 연산 결과를 출력하는 단계를 포함한다. 본 발명에 따르면, 병렬 연산을 통해 완전동형암호의 수행 속도를 높일 수 있으며 연산 리소스의 이용 효율을 높일 수 있다.

대 표 도 - 도1



(52) CPC특허분류
H04L 9/0631 (2013.01)
(72) 발명자
강민혁
서울특별시 관악구 관악로 1 서울대학교 301동 551-2호 (신림동)
천세린
서울특별시 관악구 관악로 1 서울대학교 301동 551-2호 (신림동)
이현민
서울특별시 관악구 관악로 1 서울대학교 301동 551-2호 (신림동)

명 세 서

청구범위

청구항 1

컴퓨팅 장치에서 수행되는 병렬 동형 암호 연산 방법에 있어서,

동형 암호 기능을 위한 암호문의 연산을 구성하는 복수의 단위 연산에 이용되는 암호문이 읽기 또는 쓰기인지에 따라 상기 단위 연산 간의 종속성을 판단하는 단계;

상기 종속성에 따라 제1 연산기 상에서 수행되고 있는 상기 단위 연산 중 종속되지 않는 적어도 일부의 단위 연산을 제2 연산기 상에서 수행하도록 연산을 스케줄링하는 단계;

상기 스케줄링에 따라 제1 및 제2 연산기를 통해 병렬 수행된 암호문의 연산 결과를 출력하는 단계를 포함하되,

상기 스케줄링 하는 단계는 상기 판단된 단위 연산 간의 종속성을 계층적 구조로 표현하는 그래프를 생성하되 상기 생성된 그래프 상 서로 다른 부모 노드를 갖는 단위 연산에 대하여 종속되지 않으며, 상기 연산의 변경에 따라 새로운 노드가 그래프에 추가되는 경우 재 판단된 종속성에 따라 그래프를 갱신하되,

상기 그래프는 상기 노드의 연산에 이용되는 암호문 별 읽기 또는 쓰기를 위한 부모 노드 정보 리스트를 더 포함하고, 상기 새로운 노드의 읽기 또는 쓰기에 이용되는 암호문의 부모 노드 정보를 이용하여 노드의 그래프 상의 위치가 결정되는 것을 특징으로 하는 동형 암호 연산 방법.

청구항 2

제 1 항에 있어서,

상기 스케줄링하는 단계는,

상기 서로 다른 부모 노드를 갖는 단위 연산을 각각 제1 및 제2 연산기로 연산하도록 연산 스케줄을 생성하는 것을 특징으로 하는 동형 암호 연산 방법.

청구항 3

프로세서; 및

상기 프로세서와 통신하는 메모리를 포함하고,

상기 메모리는 상기 프로세서로 하여금 동작들을 수행하게 하는 명령들을 저장하고,

상기 동작들은,

동형 암호 기능을 위한 암호문의 연산을 구성하는 복수의 단위 연산에 이용되는 암호문이 읽기 또는 쓰기인지에 따라 상기 단위 연산 간의 종속성을 판단하는 동작,

상기 종속성에 따라 제1 연산기 상에서 수행되고 있는 상기 단위 연산 중 종속되지 않는 적어도 일부의 단위 연산을 제2 연산기 상에서 수행하도록 연산을 스케줄링하는 동작,

상기 스케줄링에 따라 제1 및 제2 연산기를 통해 병렬 수행된 암호문의 연산 결과를 출력하는 동작을 포함하되,

상기 스케줄링 하는 동작은 상기 판단된 단위 연산 간의 종속성을 계층적 구조로 표현하는 그래프를 생성하되 상기 생성된 그래프 상 서로 다른 부모 노드를 갖는 단위 연산에 대하여 종속되지 않으며, 상기 연산의 변경에 따라 새로운 노드가 그래프에 추가되는 경우 재 판단된 종속성에 따라 그래프를 갱신하고,

상기 그래프는 상기 노드의 연산에 이용되는 암호문 별 읽기 또는 쓰기를 위한 부모 노드 정보 리스트를 더 포함하고, 상기 새로운 노드의 읽기 또는 쓰기에 이용되는 암호문의 부모 노드 정보를 이용하여 노드의 그래프 상의 위치가 결정되는 것을 특징으로 하는 컴퓨팅 장치.

청구항 4

제 3 항에 있어서,

상기 스케줄링하는 동작은,

상기 서로 다른 부모 노드를 갖는 단위 연산을 각각 제1 및 제2 연산기로 연산하도록 연산 스케줄을 생성하는 것을 특징으로 하는 컴퓨팅 장치.

청구항 5

제 1 항 내지 제 2 항 중 어느 한 항에 따른 컴퓨팅 장치에서 수행되는 동형 암호 연산 방법을 수행하는 프로그램이 저장된 컴퓨터 판독 가능한 기록 매체.

청구항 6

삭제

청구항 7

삭제

청구항 8

삭제

청구항 9

삭제

청구항 10

삭제

청구항 11

삭제

청구항 12

삭제

청구항 13

삭제

발명의 설명

기 술 분 야

[0001] 본 발명은 동형 암호의 연산 방법에 관한 것으로, 동형 암호의 효율적인 연산 방법에 관한 것이다.

배 경 기 술

[0002] 기본의 RSA(Rivest, Shamir, and Adleman), ECC(Elliptic Curve Cryptography), AES(Advanced Encryption Standard) 등과 같은 암호 시스템에서 암호문의 연산 처리를 위해서는 암호분을 평문으로 복호화하는 과정이 필요하며 복호화 과정에서 정보의 유출이 발생할 가능성이 존재한다.

[0003] 이를 해결하기 위해 고안된 동형암호 기술은 동형연산(Homomorphic Operation)이라는 수학적 특성을 활용하여 암호화된 데이터에 대해서도 결합 및 연산 처리를 가능하게 함으로써 복호화 과정에서의 정보 유출을 방지한다.

[0004] 즉, 동형암호는 암호문에 대해 임의의 연산을 허용함에 따라 별도의 복호화 과정 없이 암호화 된 데이터에 바로 연산을 수행할 수 있도록 해줌으로써 메시지의 안정성을 향상시킨다.

[0005] 하지만 암호화 과정에서 암호문들의 사이즈가 매우 커지며 암호문에 추가된 노이즈 값 등이 암호문 간의 연산을

반복할수록 원래의 메시지를 침범하게 되므로 이를 막기 위한 부가적인 연산들도 추가로 필요하게 된다.

[0006] 최근에는 GPU(Graphics Processing Unit), FPGA (field programmable gate array) 와 같은 하드웨어를 이용한 동형암호 가속 기술이 다양하게 제시되고 있다. 예를 들어 FPGA를 활용하여 완전동형암호의 연산 성능을 개선한 연구들이 존재하며, cuHE(CUDA Homomorphic Encryption Library), cuFHE(CUDA Fully Homomorphic Encryption Library) 등이 있지만 실제 사용되기에는 여러 제약을 지니며 여전히 연산 속도가 느리다는 한계를 보이고 있다.

발명의 내용

해결하려는 과제

[0007] 본 발명은 완전동형암호 기술에서 연산의 효율성을 높이기 위한 최적화된 연산 병렬 수행 기법을 제안하는 것을 목적으로 한다.

[0008] 본 발명은 연산들 사이의 데이터 읽기 또는 쓰기에 종속성이 없는 완전동형암호 연산들 사이의 종속성을 판단하고, 이를 기반으로 병렬 연산을 수행할 수 있도록 하는 기법을 제안하는 것을 목적으로 한다.

과제의 해결 수단

[0009] 상술한 목적을 달성하기 위한 본 발명의 일 실시 예에 따른 컴퓨팅 장치에서 수행되는 컴퓨팅 장치에서 수행되는 병렬 동형 암호 연산 방법에 있어서, 동형 암호 기능을 위한 암호문의 연산을 구성하는 복수의 단위 연산 간의 읽기 및 쓰기의 종속성을 판단하는 단계; 상기 종속성에 따라 제1 연산기 상에서 수행되고 있는 상기 단위 연산 중 종속되지 않는 적어도 일부의 단위 연산을 제2 연산기 상에서 수행하도록 연산을 스케줄링하는 단계; 및 상기 스케줄링에 따라 제1 및 제2 연산기를 통해 병렬 수행된 암호문의 연산 결과를 출력하는 단계를 포함한다.

[0010] 상기 종속성을 판단하는 단계는, 상기 단위 연산에 이용되는 암호문이 읽기 또는 쓰기인지에 따라 종속성을 판단하는 것이 바람직하다.

[0011] 상기 스케줄링 하는 단계는, 상기 판단된 단위 연산 간의 종속성을 계층적 구조로 표현하는 그래프를 생성하되, 상기 생성된 그래프 상 서로 다른 부모 노드를 갖는 단위 연산에 대하여 종속되지 않는 것이 바람직하다.

[0012] 상기 스케줄링하는 단계는, 상기 서로 다른 부모 노드를 갖는 단위 연산을 각각 제1 및 제2 연산기로 연산하도록 연산 스케줄을 생성하는 것이 바람직하다.

[0013] 상기 스케줄링하는 단계는, 상기 연산의 변경에 따라 새로운 노드가 그래프에 추가되는 경우 재 판단된 종속성에 따라 그래프를 갱신하는 것이 바람직하다.

[0014] 상기 그래프는 상기 노드의 연산에 이용되는 암호문 별 읽기 또는 쓰기를 위한 부모 노드 정보 리스트를 더 포함하고, 상기 새로운 노드의 읽기 또는 쓰기에 이용되는 암호문의 부모 노드 정보를 이용하여 노드의 그래프 상의 위치가 결정되는 것이 바람직하다.

[0015] 상기 기술적 과제를 해결하기 위한 본 실시예에 따른 병렬 동형 암호 연산 방법을 수행하는 컴퓨팅 장치는 프로세서; 및 상기 프로세서와 통신하는 메모리를 포함하고, 상기 메모리는 상기 프로세서로 하여금 동작들을 수행하게 하는 명령들을 저장하고, 상기 동작들은, 동형 암호 기능을 위한 암호문의 연산을 구성하는 복수의 단위 연산 간의 읽기 및 쓰기의 종속성을 판단하는 동작, 상기 종속성에 따라 제1 연산기 상에서 수행되고 있는 상기 단위 연산 중 종속되지 않는 적어도 일부의 단위 연산을 제2 연산기 상에서 수행하도록 연산을 스케줄링하는 동작, 및 상기 스케줄링에 따라 제1 및 제2 연산기를 통해 병렬 수행된 암호문의 연산 결과를 출력하는 동작을 포함하는 것이 바람직하다.

[0016] 상기 종속성을 판단하는 동작은, 상기 단위 연산에 이용되는 암호문이 읽기 또는 쓰기인지에 따라 종속성을 판단하는 것이 바람직하다.

[0017] 상기 스케줄링 하는 동작은, 상기 판단된 단위 연산 간의 종속성을 계층적 구조로 표현하는 그래프를 생성하되, 상기 생성된 그래프 상 서로 다른 부모 노드를 갖는 단위 연산에 대하여 종속되지 않는 것이 바람직하다.

[0018] 상기 스케줄링하는 동작은, 상기 서로 다른 부모 노드를 갖는 단위 연산을 각각 제1 및 제2 연산기로 연산하도록 연산 스케줄을 생성하는 것이 바람직하다.

[0019] 상기 스케줄링하는 동작은, 상기 연산의 변경에 따라 새로운 노드가 그래프에 추가되는 경우 재 판단된 종속성에 따라 그래프를 갱신하는 것이 바람직하다.

[0020] 상기 그래프는 상기 노드의 연산에 이용되는 암호문 별 읽기 또는 쓰기를 위한 부모 노드 정보 리스트를 더 포함하고, 상기 새로운 노드의 읽기 또는 쓰기에 이용되는 암호문의 부모 노드 정보를 이용하여 노드의 그래프 상의 위치가 결정되는 것이 바람직하다.

[0021] 한편, 상술한 목적을 달성하기 위한 본 발명의 일 실시 예에 따른 기록 매체에 저장된 프로그램은 상술한 연산 방법을 실행하기 위한 프로그램 코드를 포함할 수 있다.

발명의 효과

[0022] 본 발명에 따르면, 병렬 연산을 통해 완전동형암호의 수행 속도를 높일 수 있으며 연산 리소스의 이용 효율을 높일 수 있다.

[0023] 또한, 현재 완전동형암호의 느린 연산 속도로 인해 적용하고 있지 못하던 다양한 분야에서 완전동형암호를 활용할 수 있다.

[0024] 또한, 동형암호의 적용범위의 확장을 통해 사용자의 개인 정보가 유출되는 문제를 해결할 수 있다.

도면의 간단한 설명

[0025] 도 1은 본 발명의 일 실시예에 따른 동형 암호 연산 방법을 나타낸 흐름도이다.

도 2는 본 발명의 일 실시예에 따른 동형 암호 연산 방법의 병렬 처리를 나타낸 예시도이다.

도 3은 본 발명의 일 실시예에 따른 종속성 그래프의 생성 과정을 나타낸 예시도이다.

도 4a는 본 발명의 일 실시예에 따른 암호문 테이블을 나타낸 예시도이다.

도 4b는 본 발명의 일 실시예에 따른 암호문 테이블을 이용한 노드 추가 과정을 나타낸 예시도이다.

도 5는 본 발명의 일 실시예에 따른 종속성 그래프의 노드 삭제 과정을 나타낸 예시도이다.

도 6은 본 발명의 일 실시예에 따른 종속성 그래프의 노드 추가 과정을 나타낸 예시도이다.

도 7은 본 발명의 일 실시예에 따른 컴퓨팅 장치의 구성을 나타낸 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0026] 이하의 내용은 단지 본 발명의 원리를 예시한다. 그러므로 당업자는 비록 본 명세서에 명확히 설명되거나 도시되지 않았지만 본 발명의 원리를 구현하고 본 발명의 개념과 범위에 포함된 다양한 장치를 발명할 수 있는 것이다. 또한, 본 명세서에 열거된 모든 조건부 용어 및 실시 예들은 원칙적으로, 본 발명의 개념이 이해되도록 하기 위한 목적으로만 명백히 의도되고, 이와 같이 특별히 열거된 실시 예들 및 상태들에 제한적이지 않는 것으로 이해되어야 한다.

[0027] 상술한 목적, 특징 및 장점은 첨부된 도면과 관련한 다음의 상세한 설명을 통하여 보다 분명해질 것이며, 그에 따라 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자가 본 발명의 기술적 사상을 용이하게 실시할 수 있을 것이다.

[0028] 또한, 본 발명을 설명함에 있어서 본 발명과 관련된 공지 기술에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에 그 상세한 설명을 생략하기로 한다.

[0029] 이하에는 첨부된 도면을 참조하여 본 발명의 다양한 실시 예에 대하여 설명한다.

[0030] 도 1은 본 발명의 일 실시예에 따른 동형 암호 연산 방법을 나타내는 흐름도이다.

[0031] 도 1을 참조하면 본 실시예에 따른 동형 암호 연산 방법은 컴퓨팅 장치에서 병렬로 수행될 수 있으며 이를 위한 전제로 동형 암호 기능을 위한 암호문의 연산을 구성하는 복수의 단위 연산 간의 읽기 및 쓰기의 종속성을 판단한다(S100).

[0032] 구체적으로 본 실시예에서 동형 암호는 완전 동형 암호(Fully Homomorphic Encryption, FHE)로서 별도의 복호화 과정이 필요없이 암호문에 대해 직접 연산을 가능케한다.

[0033] 완전 동형 암호 연산 사이의 데이터 읽기/쓰기 종속성이 없는 경우에는 해당 연산들이 서로의 결과에 영향을 주지 않으므로 병렬 수행이 가능하다. 따라서 연산들을 분석하여 상호 종속성이 없는 연산들을 판별한 후, 해당 연산들을 병렬 수행하여 완전동형암호 연산의 효율성을 높인다.

[0034] 도 2를 참고하면, 본 실시예에 따른 동형 암호 연산은 복수의 곱셈 연산을 수행하는 단위 연산으로 구분될 수 있으며 각 연산에 이용되는 암호문의 읽기/쓰기에 따라 종속성이 결정될 수 있다.

[0035] 예를 들어, 도 2에서 하나의 완전 동형 암호 기능(예를 들어, 곱셈)을 수행하기 위해서는 내부적으로 여러 연산들이 수행될 수 있다.

[0036] 이때, 각 연산은 CPU와 FPGA 상에서 수행되며, 기존 완전 동형 암호 구현에서는 CPU 연산과 FPGA 연산을 동시에 수행하지 않음에 따라 CPU에서 연산이 수행되는 동안 FPGA는 휴지 상태를 유지하며, 반대로 FPGA에서 연산이 수행되는 동안 CPU는 휴지 상태를 유지하므로 추가적인 지연이 발생할 수 있다.

[0037] 따라서 본 실시예에서는 연산기로 CPU나 FPGA가 연산 중일 때 휴지 중인 연산기를 통해 병렬 수행이 가능한 연산들을 찾고 수행할 수 있도록 한다.

[0038] 구체적으로 본 실시예에서 컴퓨팅 장치는 단위 연산에 이용되는 암호문이 읽기 또는 쓰기인지에 따라 종속성을 판단한다.

[0039] 도 2에서 암호문 CT5에 CT1과 CT2 곱셈 결과를 쓰는 제1 곱셈 연산(22)과 CT6에 곱셈 결과를 쓰는 제2 곱셈 연산(23)은 서로 곱셈에 이용하기 위해 읽어들이는 암호문과 서로의 쓰기 결과와 종속성이 없으므로 병렬 수행이 가능한 것으로 판단할 수 있다.

[0040] 반면, CT7의 경우(25) CT5, CT6를 읽고 곱셈 결과를 CT7에 쓰게 되므로 제1 곱셈 연산과 제2 곱셈 연산에 종속될 수 있다. 반면 CT10(24)은 CT8과 CT9를 읽고 곱셈 결과를 쓰게 되므로 해당 연산의 경우 종속성이 없는 것으로 판단한다.

[0041] 따라서, 컴퓨팅 장치는 이상의 판단된 종속성에 따라 제1 연산기(예를 들어 CPU) 상에서 수행되고 있는 단위 연산 중 종속되지 않는 적어도 일부의 단위 연산을 제2 연산기(예를 들어 FPGA) 상에서 수행하도록 연산을 스케줄링한다(S200)

[0042] 구체적으로 컴퓨팅 장치는 상기 판단된 단위 연산 간의 종속성을 계층적 구조로 표현하는 그래프를 생성하여 생성된 그래프 상 서로 다른 부모 노드를 갖는 단위 연산에 대하여 종속되지 않는 것으로 스케줄링을 수행할 수 있다.

[0043] 따라서, 서로 다른 부모 노드를 갖는 단위 연산을 선행된 단위 연산을 수행하는 제1 연산기와, 다른 휴지 중인 제2 연산기로 연산하도록 연산 스케줄을 생성한다.

[0044] 도 3을 참고하면, 본 실시예에 따른 컴퓨팅 장치 내 프로세서(308)는 연산의 순서를 결정하는 스케줄러(Scheduler)(32)를 포함하고, 스케줄러(32)는 각 연산(곱셈, 덧셈 등)들의 종속성을 검사하여 매 시점 연산들의 관계를 나타내는 종속성 그래프(DAG, Directed Acyclic Graph) 사용하여 스케줄을 생성할 수 있다.

[0045] 본 실시예에서 그래프는 하나의 단위 연산을 표현하는 노드(34)와 단위 연산 사이의 종속성을 표현하는 에지(36)로 구성될 수 있다. 이때, 그래프에서 자식 노드는 부모 노드의 연산이 모두 먼저 수행되어야 연산 수행이 가능하다.

[0046] 즉, 그래프 상에서 상위 계층부터 연산 수행이 끝난 후 그 다음 계층의 연산을 수행할 수 있으며, 종속성 그래프의 헤드는 어떤 연산에도 종속되지 않는 연산들로 해당 시점에 즉시 실행 가능한 연산들을 가리킬 수 있다.

[0047] 상술한 바와 같이 연산에서 사용하는 암호문 CT(Ciphertext)가 읽기인지 쓰기인지에 따라 종속성이 달라지므로 이를 종속성 그래프로 생성하여 보다 용이하게 판단할 수 있도록 한다.

[0048] 이상의 도 3에 따른 종속성 그래프에서 노드 1, 2, 3은 동시에 실행 가능한 연산들로 판단될 수 있다.

[0049] 이때, 노드 1과 노드 3은 같은 암호문(CT2)에 접근하지만 해당 암호문을 읽기만 하고 변경(쓰기)은 하지 않으므로 동시에 수행 가능하다. 반면 노드 4는 노드 1이 수행된 이후에 수행될 수 있는데, 노드 1과 노드 4 모두 같은 암호문(CT1)을 변경(쓰기)때문이다. 또한, 노드 4는 암호문(CT5)을 읽기는 하지만 노드 5가 해당 암호문을 변경하는 작업(Destructor, 소멸자)을 하기 때문에 노드 5는 노드 4가 수행된 이후에 수행될 수 있다.

[0050] 즉, 본 실시예에 따른 스케줄러(32)는 종속성 그래프의 노드와 에지를 통해 동시에 수행가능한 노드에 대해서는 각각의 연산기를 통해 병렬 수행할 수 있도록 연산 스케줄을 생성한다.

[0051] 이어서 생성된 스케줄에 따라 제1 및 제2 연산기를 통해 병렬 수행된 암호문의 연산 결과를 출력한다(S300).

[0052] 나아가 본 실시예에서 스케줄러(32)는 각 노드 별 암호문의 크기, 연산의 부하 정도를 산출하고 해당 리소스와 가용 가능한 연산기의 리소스를 비교하여 실제 스케줄을 수행하는 것도 가능하며, 연산 후 노드가 삭제됨에 따라 가용 가능한 리소스를 예측함으로써 연산기와 병렬 가능한 각 노드를 대응시킬 수 있다.

[0053] 이상의 본 실시예에 따라 구현하는 완전 동형 암호 기술은 양자컴퓨터의 공격에도 안전한 수준의 민감 데이터 보호와 인공지능 등의 서비스에 개인정보의 암호화를 통해 상충되는 요구사항을 해결할 있다.

[0054] 구체적으로 본 발명에 따른 컴퓨팅 장치는 완전동형암호의 연산 효율을 높일 수 있는 병렬 수행 기법을 제시함으로써, 클라우드 서비스와 같이 민감한 정보(예를 들어, 개인 정보)를 원격 서버에서 처리해야 하는 경우 완전동형암호 기술을 적용하여 프라이버시 문제를 해결할 수 있으며, 프라이버시 기반 인증 분야 및 PIR(Private Information Retrieval) 등 다양한 분야에서 활용될 수 있도록 한다.

[0055] 나아가, 본 실시예에서는 상기 연산의 변경에 따라 새로운 노드가 그래프에 추가되는 경우 재 판단된 종속성에 따라 그래프를 갱신할 필요가 있다.

[0056] 본 실시예에 따른 컴퓨팅 장치는 동형 암호 연산의 종료 또는 추가 등의 상태에 맞게 스케줄러가 종속성 그래프를 갱신한다.

[0057] 예를 들어, 종속성 그래프에 포함된 연산의 수행이 완료되는 경우 연산에 해당하는 노드를 종속성 그래프에서 삭제할 수 있다.

[0058] 또는 새로운 연산이 호출되는 경우에는 이에 해당하는 새로운 노드를 종속성 그래프에 삽입시켜 연산들 간의 종속성을 다시 판단할 수 있다.

[0059] 특히, 새로운 노드를 종속성 그래프에 삽입하는 경우 기존 노드들 사이의 종속성을 계산하는 과정이 필요하다. 따라서 종속성 그래프에서 노드가 삭제되는 속도보다 새로운 노드가 삽입되는 속도가 빠른 경우, 그래프 삽입을 위한 종속성 계산 복잡도 및 시간이 증가하게 되며, 결국 이는 동형 암호 연산을 수행하는 전체 어플리케이션 속도에 영향을 줄 수 있다.

[0060] 또는 새로운 노드의 삽입 과정에서 병렬 연산을 위한 종속성 그래프의 갱신 과정에서 지연을 최소화하기 위한 별도의 자료구조를 이용할 수 있다.

[0061] 따라서, 본 발명에 따른 컴퓨팅 장치는 새로운 연산 노드를 종속성 그래프에 효율적으로 삽입하기 위한 추가적인 테이블을 생성한다.

[0062] 상술한 바와 같이 종속성은 각 암호문마다 결정되므로 특정 암호문마다 새로 만들어진 연산의 노드가 그래프 상의 어느 위치에 삽입되어야 하는지를 알려줄 수 있는 테이블을 미리 생성하여 이용할 수 있다.

[0063] 즉, 본 실시예에서 그래프는 상기 노드의 연산에 이용되는 암호문 별 읽기 또는 쓰기를 위한 부모 노드 정보 리스트를 더 포함하고, 새로운 노드의 읽기 또는 쓰기 이용되는 암호문의 부모 노드 정보를 이용하여 노드의 그래프 상의 위치가 결정되도록 한다.

[0064] 구체적으로 본 발명에 따른 컴퓨팅 장치는 연산이 해당 암호문에하는 연산(읽기 혹은 쓰기)에 따라 삽입되는 위치가 달라지며, 이를 위해 암호문마다 읽기/쓰기 맵을 리스트로 만들어서 관리한다.

[0065] 본 실시예에서 읽기/쓰기 맵(46)은 노드를 새로 추가해야할 때 부모 노드가 되어야 하는 노드의 리스트일 수 있다.

[0066] 도 4a 및 4b를 참고하면, CT2는 모든 연산이 읽기만 하므로 CT2의 읽기 맵은 헤드 노드를 가리킨다. CT5의 경우 노드 4가 가장 하위 레벨에서 읽기 작업을 하고 있으므로 CT5의 쓰기 맵은 노드 4를 가리킴으로써, CT5가 노드 4의 연산에 이용된 후 갱신될 수 있도록 한다.

[0067] 이때 새로 추가될 노드 5는 CT2를 읽고, CT5에 쓰기를 하므로 각각 헤드 이후 또는 노드 4 다음에 삽입되어야 하는데, 본 실시예에 따른 그래프 상에서 노드 4가 더 하위 레벨에 있으므로 노드 4 다음에 삽입될 수 있도록 위치를 결정할 수 있다.

[0068] 도 4b를 참고하면 결정된 위치로 노드 5(44)가 종속성 그래프(42)에 삽입되면 CT2의 읽기 맵은 그대로 헤드를 가리키며, CT2의 쓰기 맵은 노드 5에서 연산에 이용된 후 쓰기가 수행될 수 있도록 노드 5를 가리키도록 업데이트 된다. CT5의 읽기 맵과 쓰기 맵은 모두 노드 5를 가리키도록 업데이트 된다.

[0069] 이상 본 실시예에 따른 컴퓨팅 장치는 종속성 그래프와 읽기/쓰기맵을 통해 완전 동형 암호 연산들의 종속성을 관리하며, 종속성이 없는 연산들은 동시 수행을 할 수 있으며 기존 노드가 삭제(연산 수행 및 종료)되거나 새로운 노드가 삽입(연산 추가)되는 경우 그래프를 갱신하면서 스케줄링을 재 수행한다.

[0070] 이어서 도면을 참고하여 노드가 삭제되는 경우와 노드가 삽입되는 경우의 구체적인 동작에 대하여 설명한다.

[0071] 먼저 도 5를 참고하면 연산 수행 및 종료로 노드가 삭제되는 경우로, 그래프에서 노드 1과 노드 2가 실행되면 해당 노드들은 종속성 그래프에서 삭제될 수 있다.

[0072] 노드 4는 어떠한 노드에도 종속되지 않으므로 종속성 그래프의 헤드는 노드 4를 가리키도록 갱신된다.

[0073] 이어서, 연산의 추가로 노드가 삽입되는 경우로 새로운 연산이 호출되어 종속성 그래프에 노드가 추가되는 과정을 도 6을 참고하여 설명하면, 노드 5(72)는 두 암호문(CT2, CT4)에 접근하여, 먼저 각각 암호문을 기준으로 실행 가능한 시점을 검사한 뒤, 두 시점 중 나중 시점에 연산을 그래프에 추가할 수 있다.

[0074] 구체적으로 삽입되는 노드에서 읽기 암호문과 쓰기 암호문을 각각 구분하고, 읽기 암호문에 대하여 이전 연산에서 쓰기가 수행되는 경우 가장 늦은(하위계층) 해당 노드 이후를 읽기 맵으로 결정할 수 있다.

[0075] 반대로 쓰기 암호문은 해당 암호문에 대하여 이전 연산의 읽기와 쓰기를 수행하는 노드 중 가장 늦은 노드를 쓰기 맵으로 결정할 수 있다.

[0076] 노드 5는 CT4에 연산 후 값을 쓰기 때문에 이전 연산들에서 CT4에 접근하는 모든 연산이 실행된 이후에 실행되어야 함에 따라 노드 2 이후에 실행되어야 한다.

[0077] 노드 5는 CT2에도 접근하지만 읽기 연산이기 때문에, CT2에 쓰는 작업을 하는 연산들 이후에만 실행되면 되며, 현재는 CT2에 대한 쓰기 연산은 이전 연산에서 존재하지 않고 모든 연산이 CT2를 읽기만 하므로 헤드 이후로 실행될 수 있다.

[0078] 따라서, 헤드와 노드 2 중 노드 2가 더 나중 시점이므로 노드 2 이후에 실행되도록 종속성 그래프에 노드를 삽입한다.

[0079] 이상의 부모 노드를 결정하기 위해 본 실시예에 따른 컴퓨팅 장치는 상술한 바와 같이 별도의 리스트로 읽기/쓰기 맵 정보를 관리하며, 노드의 삽입 시 갱신될 수 있다. 따라서 갱신되는 리스트를 통해 노드의 추가에서 이용되는 암호문 들의 부모 노드를 결정하고 삽입 위치를 판단할 수 있다.

[0080] 나아가 도 7을 참조하면, 본 발명의 몇몇 실시예들에서 연산 방법은 컴퓨팅 장치의 형태로 구현될 수 있다. 컴퓨팅 장치(300)를 구성하는 각각의 모듈 중 하나 이상은 범용 컴퓨팅 프로세서 상에서 구현되며 따라서 프로세서(processor)(308), 입출력 I/O(302), 메모리 (memory)(340), 인터페이스(interface)(306) 및 버스(314, bus)를 포함할 수 있다. 프로세서(308), 입출력 장치(302), 메모리 (304) 및/또는 인터페이스(306)는 버스(314)를 통하여 서로 결합될 수 있다. 버스(314)는 데이터들이 이동되는 통로(path)에 해당한다.

[0081] 구체적으로, 프로세서(308)는 CPU(Central Processing Unit), MPU(Micro Processor Unit), MCU(Micro Controller Unit), GPU(Graphic Processing Unit), 마이크로프로세서, 디지털 신호 프로세스, 마이크로컨트롤러, 어플리케이션 프로세서(AP, application processor) 및 이들과 유사한 기능을 수행할 수 있는 논리 소자들 중에서 적어도 하나를 포함할 수 있다.

[0082] 입출력 장치(302)는 키패드(keypad), 키보드, 터치스크린 및 디스플레이 장치 중 적어도 하나를 포함할 수 있다. 메모리 장치(304)는 데이터 및/또는 프로그램 등을 저장할 수 있다.

[0083] 인터페이스(306)는 통신 네트워크로 데이터를 전송하거나 통신 네트워크로부터 데이터를 수신하는 기능을 수행할 수 있다. 인터페이스(306)는 유선 또는 무선 형태일 수 있다. 예컨대, 인터페이스(306)는 안테나 또는 유무선 트랜시버 등을 포함할 수 있다. 메모리 (304)는 프로세서(308)의 동작을 향상시키되, 개인정보의 보호를 위한 휘발성의 동작 메모리로서, 고속의 디램 및/또는 에스램 등을 더 포함할 수도 있다.

[0084] 또한, 메모리(304) 내에는 여기에 설명된 일부 또는 모든 모듈의 기능을 제공하는 프로그래밍 및 데이터 구성을 저장한다. 예를 들어, 상술한 학습 방법의 선택된 양태들을 수행하도록 하는 로직을 포함할 수 있다.

[0085] 메모리 (304)에 저장된 상술한 획득 방법을 수행하는 각 단계를 포함하는 명령어들의 집합으로 프로그램 또는 어플리케이션을 로드하고 프로세서가 각 단계를 수행할 수 있도록 한다.

[0086] 나아가, 여기에 설명되는 다양한 실시예는 예를 들어, 소프트웨어, 하드웨어 또는 이들의 조합된 것을 이용하여 컴퓨터 또는 이와 유사한 장치로 읽을 수 있는 기록매체 내에서 구현될 수 있다.

[0087] 하드웨어적인 구현에 의하면, 여기에 설명되는 실시예는 ASICs (application specific integrated circuits), DSPs (digital signal processors), DSPDs (digital signal processing devices), PLDs (programmable logic devices), FPGAs (field programmable gate arrays, 프로세서(processors), 제어기(controllers), 마이크로 컨트롤러(micro-controllers), 마이크로 프로세서(microprocessors), 기타 기능 수행을 위한 전기적인 유닛 중 적어도 하나를 이용하여 구현될 수 있다. 일부의 경우에 본 명세서에서 설명되는 실시예들이 제어 모듈 자체로 구현될 수 있다.

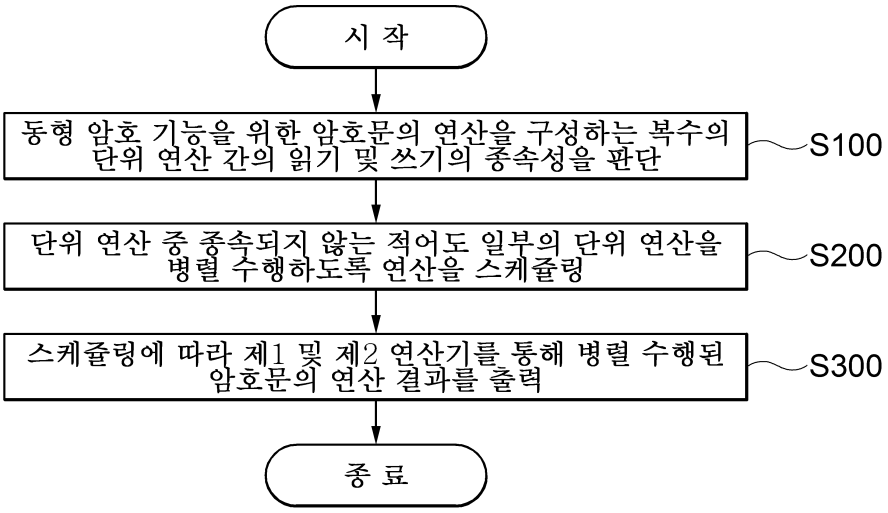
[0088] 소프트웨어적인 구현에 의하면, 본 명세서에서 설명되는 절차 및 기능과 같은 실시예들은 별도의 소프트웨어 모듈들로 구현될 수 있다. 상기 소프트웨어 모듈들 각각은 본 명세서에서 설명되는 하나 이상의 기능 및 작동을 수행할 수 있다. 적절한 프로그램 언어로 쓰여진 소프트웨어 어플리케이션으로 소프트웨어 코드가 구현될 수 있다. 상기 소프트웨어 코드는 메모리 모듈에 저장되고, 제어모듈에 의해 실행될 수 있다.

[0089] 이상의 설명은 본 발명의 기술 사상을 예시적으로 설명한 것에 불과한 것으로서, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자라면 본 발명의 본질적인 특성에서 벗어나지 않는 범위 내에서 다양한 수정, 변경 및 치환이 가능할 것이다.

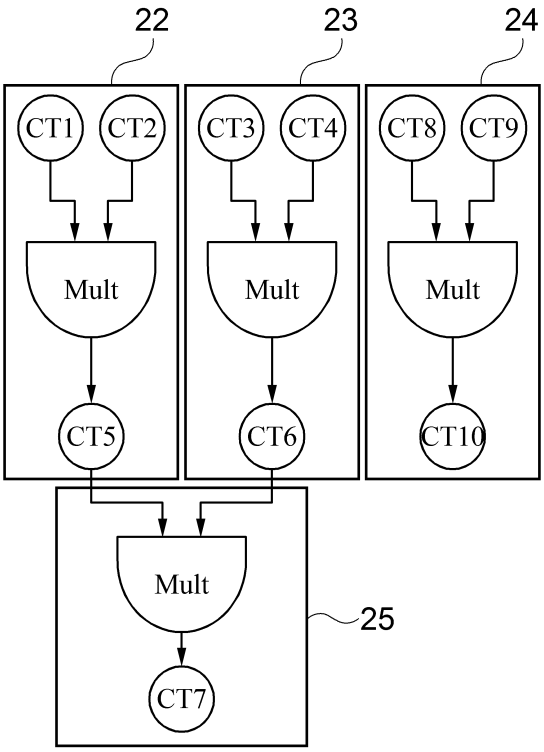
[0090] 따라서, 본 발명에 개시된 실시 예 및 첨부된 도면들은 본 발명의 기술 사상을 한정하기 위한 것이 아니라 설명하기 위한 것이고, 이러한 실시 예 및 첨부된 도면에 의하여 본 발명의 기술 사상의 범위가 한정되는 것은 아니다. 본 발명의 보호 범위는 아래의 청구 범위에 의하여 해석되어야 하며, 그와 동등한 범위 내에 있는 모든 기술 사상은 본 발명의 권리 범위에 포함되는 것으로 해석되어야 할 것이다.

도면

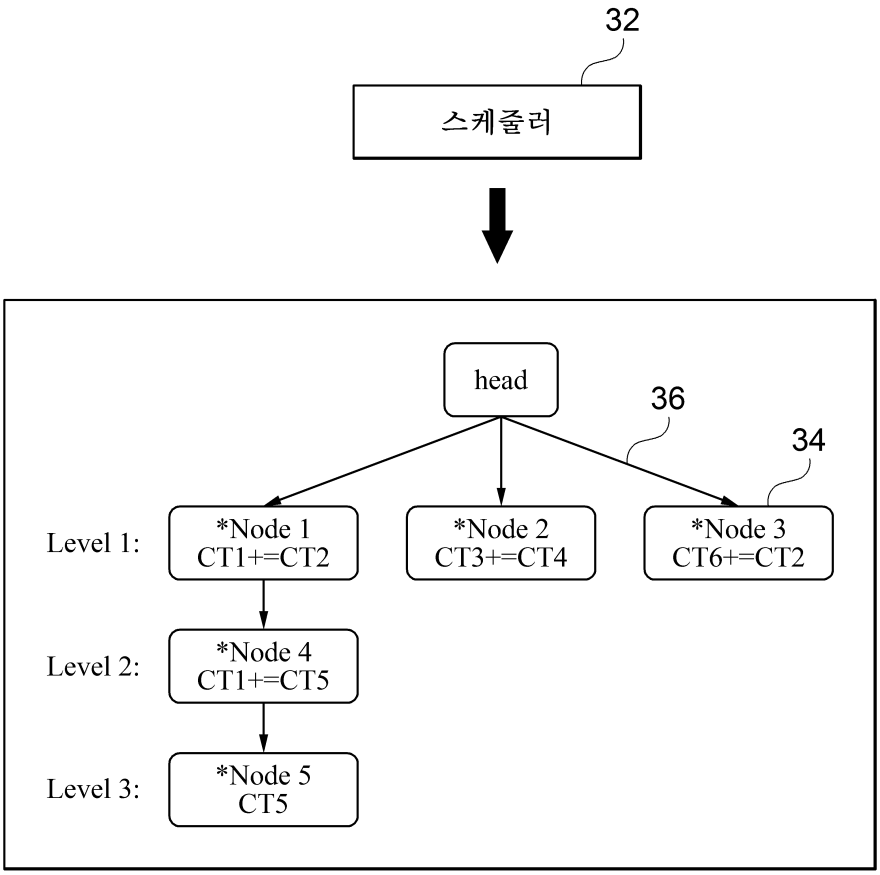
도면1



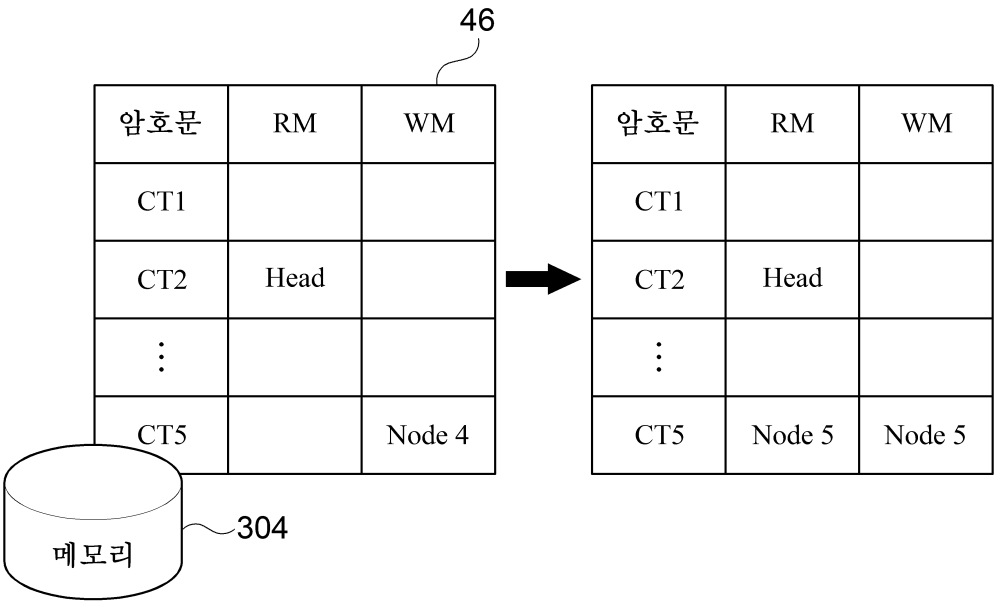
도면2


22
23
24
CT1
CT2
CT3
CT4
CT8
CT9
Mult
Mult
Mult
CT5
CT6
CT10
Mult
25
CT7

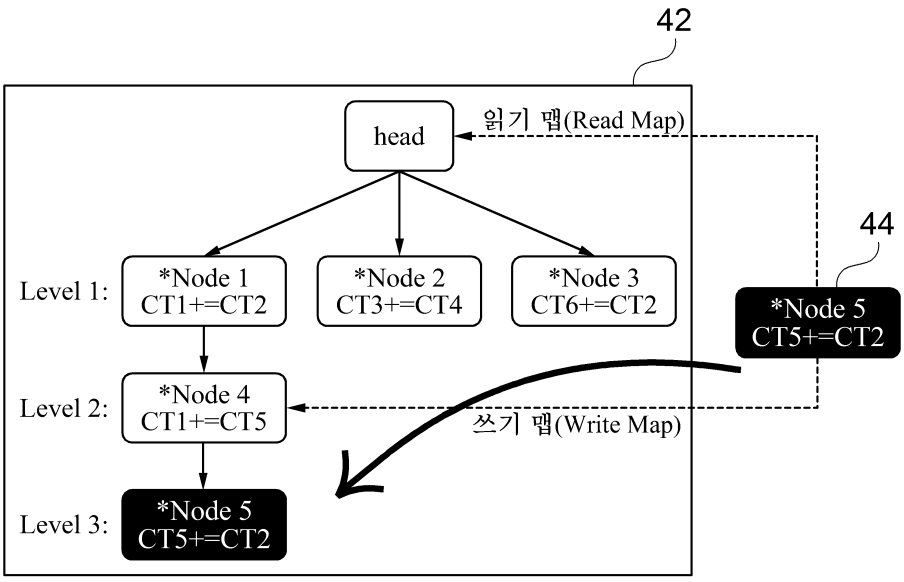
도면3


32
스케줄러
head
36
34
Level 1:
*Node 1
CT1+=CT2
*Node 2
CT3+=CT4
*Node 3
CT6+=CT2
Level 2:
*Node 4
CT1+=CT5
Level 3:
*Node 5
CT5

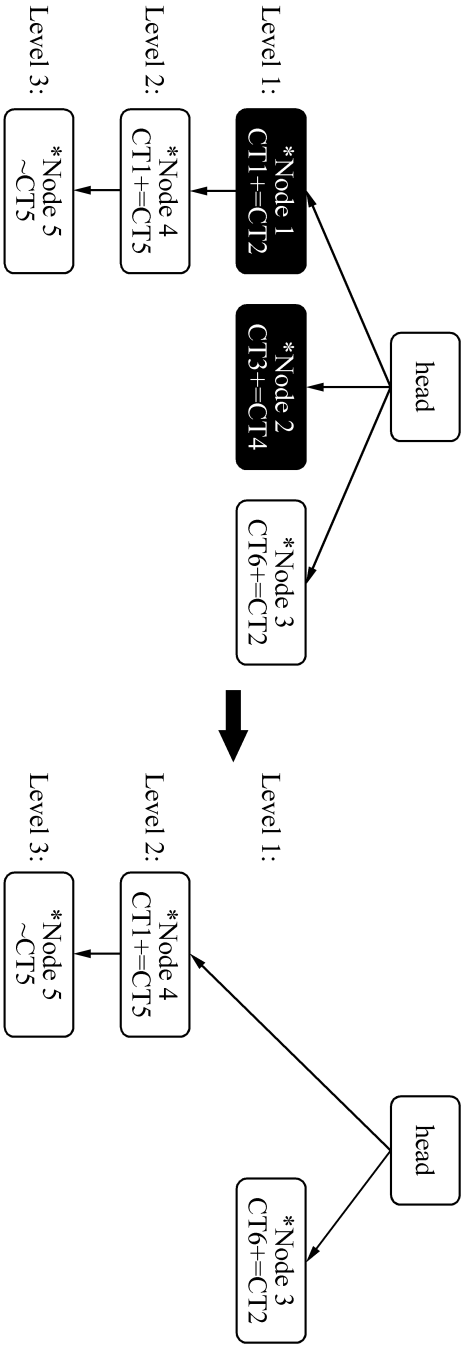
도면4a


46
암호문
RM
WM
CT1
CT2
Head
CT5
Node 4
암호문
RM
WM
CT1
CT2
Head
CT5
Node 5
Node 5
메모리
304

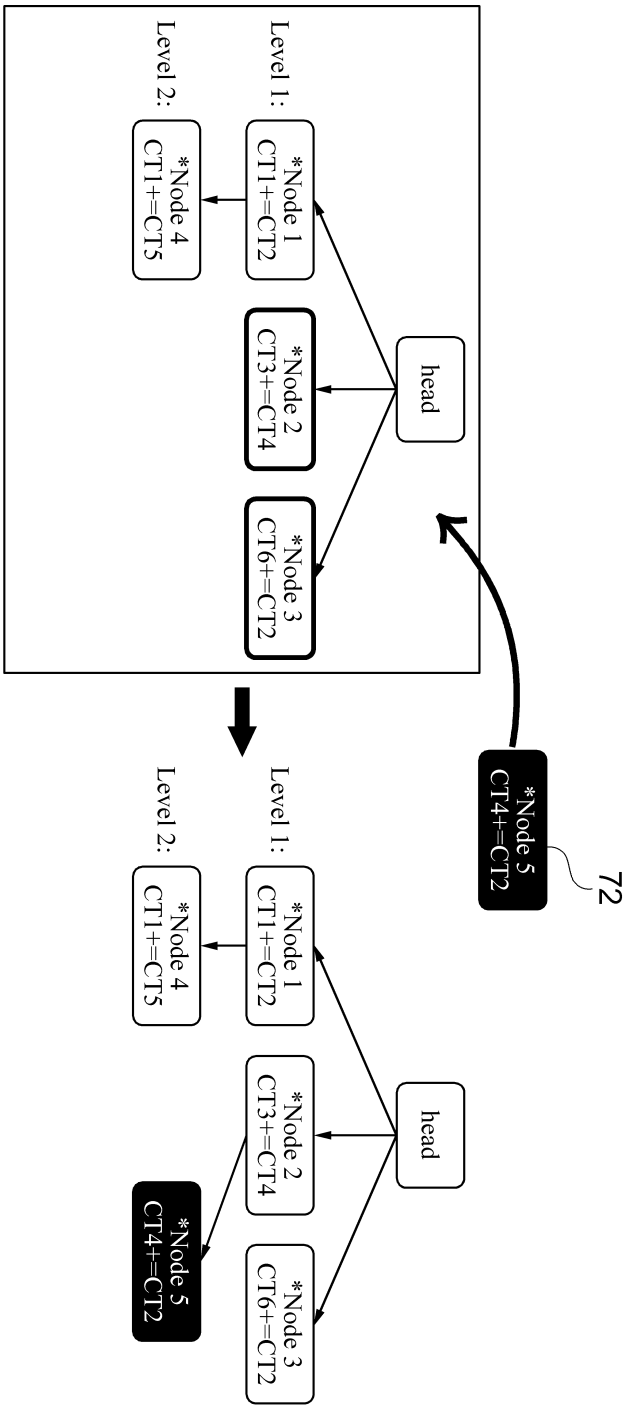
도면4b


42
head
읽기 맵(Read Map)
44
Level 1:
*Node 1
CT1+=CT2
*Node 2
CT3+=CT4
*Node 3
CT6+=CT2
*Node 5
CT5+=CT2
Level 2:
*Node 4
CT1+=CT5
쓰기 맵(Write Map)
Level 3:
*Node 5
CT5+=CT2

도면5


Level 1:
Level 2:
Level 3:
*Node 1
CT1+=CT2
*Node 4
CT1+=CT5
*Node 5
~CT5
*Node 2
CT3+=CT4
head
*Node 3
CT6+=CT2
Level 1:
Level 2:
Level 3:
*Node 4
CT1+=CT5
*Node 5
~CT5
head
*Node 3
CT6+=CT2

도면6


Level 1:
Level 2:
*Node 1
CT1+=CT2
*Node 4
CT1+=CT5
*Node 2
CT3+=CT4
*Node 3
CT6+=CT2
head
*Node 5
CT4+=CT2
72
Level 1:
Level 2:
*Node 1
CT1+=CT2
*Node 4
CT1+=CT5
*Node 2
CT3+=CT4
*Node 5
CT4+=CT2
*Node 3
CT6+=CT2
head

도면7