



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2017-0091902
(43) 공개일자 2017년08월10일

(51) 국제특허분류(Int. Cl.)
G06Q 20/02 (2012.01) G06F 21/62 (2013.01)
H04L 9/06 (2006.01)
(52) CPC특허분류
G06Q 20/02 (2013.01)
G06F 21/6209 (2013.01)
(21) 출원번호 10-2016-0012760
(22) 출원일자 2016년02월02일
심사청구일자 2016년02월02일

(71) 출원인
주식회사 코인플러그
경기도 성남시 분당구 성남대로 331번길 8, 801호
(정자동, 킨스타워)
(72) 발명자
어준선
경기도 성남시 분당구 느티로 22, B동 1710호(정자동)
홍재우
서울특별시 은평구 연서로 149, 1203호
송주한
경기도 성남시 분당구 정자일로 177 인텔리지II, A동 2309호
(74) 대리인
특허법인 수

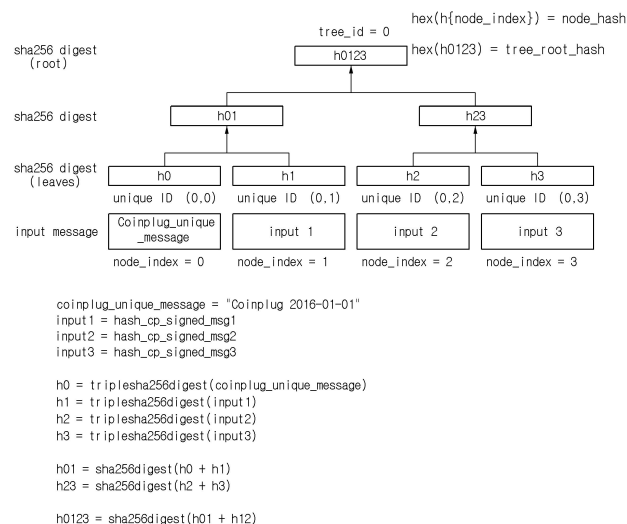
전체 청구항 수 : 총 38 항

(54) 발명의 명칭 파일에 대한 노터리 서비스를 제공하고 상기 노터리 서비스를 사용하여 기록된 파일에 대한 검증을 수행하는 방법 및 서버

(57) 요약

본 발명의 일 태양에 따르면, 파일에 대한 노터리(notary) 서비스를 제공하는 방법에 있어서, (a) 특정 파일에 대한 노터리 서비스 요청이 획득되면, 서버가, hash 함수를 사용하여 상기 특정 파일의 message digest를 생성하거나 생성하도록 지원하는 단계, 및 (b) 소정의 조건이 만족되면, 상기 서버가, 특정 해쉬값 - 상기 특정 해쉬값은, 상기 message digest에 대하여 특정 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 매칭되는 적어도 하나의 이웃 해쉬값을 연관함으로써 생성되는 대표 해쉬값 또는 상기 대표 해쉬값을 가공한 값을 데이터베이스에 등록하거나 등록하도록 지원하는 단계를 포함하는 방법이 제공된다.

대표도 - 도3



(52) CPC특허분류

G06F 21/6218 (2013.01)

H04L 9/0637 (2013.01)

H04L 9/0643 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호 R6910-15-1029

부처명 미래창조과학부

연구관리전문기관 정보통신기술진흥센터

연구사업명 K-Global ICT 유망기술개발지원 (시장친화형 SW R&D지원(MOS))

연구과제명 블록체인 (blockchain) 을 이용한 멀티시그니처 (multi-signature) 전자지갑 기술 및 금융
보안 다중인증 기술 개발

기 여 율 1/1

주관기관 (주)코인플러그

연구기간 2015.09.01 ~ 2017.08.31

명세서

청구범위

청구항 1

파일에 대한 노터리(notary) 서비스를 제공하는 방법에 있어서,

(a) 특정 파일에 대한 노터리 서비스 요청이 획득되면, 서버가, hash 함수를 사용하여 상기 특정 파일의 message digest를 생성하거나 생성하도록 지원하는 단계, 및

(b) 소정의 조건이 만족되면, 상기 서버가, 특정 해쉬값 - 상기 특정 해쉬값은, 상기 message digest에 대하여 특정 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써 생성되는 대표 해쉬값 또는 상기 대표 해쉬값을 가공한 값을 데이터베이스에 등록하거나 등록하도록 지원하는 단계

를 포함하는 방법.

청구항 2

제1항에 있어서,

상기 (b) 단계에서,

상기 소정의 조건은,

(i) 복수의 노터리 서비스 요청이 이루어져 message digest가 소정의 개수만큼 획득되는 조건, (ii) 소정 시간이 경과하는 조건, (iii) 블록 체인에서 블록이 생성되는 조건, (iv) 노터리 서비스 요청을 한 사용자 특성에 대한 조건 중 적어도 하나를 포함하는 것을 특징으로 하는 방법.

청구항 3

제1항에 있어서,

상기 (b) 단계에서,

상기 서버는, 상기 특정 해쉬값이 리프 노드에 할당된 머클 트리(merkle tree)를 생성하거나 생성하도록 지원하고,

상기 소정의 조건이 만족되면, 상기 특정 해쉬값과 매칭되는 적어도 하나의 다른 리프 노드에 할당된 해쉬값을 연산하여 생성되는 상기 대표 해쉬값 또는 상기 대표 해쉬값을 가공한 값을 상기 데이터베이스에 등록하거나 등록하도록 지원하는 것을 특징으로 하는 방법.

청구항 4

제3항에 있어서,

상기 머클 트리가 체인 형태로 연결된 적어도 하나의 머클 트리 중 첫번째 머클 트리인 경우, 상기 머클 트리의 첫번째 리프 노드에는 텍스트, 숫자, 또는 기호로 이루어진 소정의 메시지 데이터의 해쉬값 또는 이를 가공한 값이 할당되는 것을 특징으로 하는 방법.

청구항 5

제3항에 있어서,

상기 소정의 조건이 만족되면,

(x1) 상기 서버는, (i) 상기 특정 해쉬값과 (ii) 상기 특정 해쉬값이 할당된 노드의 형제 노드에 할당된 해쉬값을 연산하거나 연산하도록 지원하고, 상기 연산값에 대한 해쉬값을 상기 노드의 부모 노드에 할당하거나 할당하도록 지원하며,

(x2) 상기 부모 노드가 상기 머클 트리의 루트 노드이면, 상기 부모 노드에 할당된 해쉬값을 상기 대표 해쉬값으로서 상기 데이터베이스에 등록하거나 등록하도록 지원하고,

(x3) 상기 부모 노드가 상기 머클 트리의 루트 노드가 아니면, 상기 부모 노드에 할당된 해쉬값을 상기 특정 해쉬값으로 하여 상기 (x1) 내지 (x3)를 반복하여 수행하는 것을 특징으로 하는 방법.

청구항 6

제5항에 있어서,

상기 (x1)에서

상기 소정의 조건이 만족되었음에도 상기 특정 해쉬값이 할당된 노드의 형제 노드에 해쉬값이 할당되어 있지 않은 경우, 상기 서버는, 상기 형제 노드에 소정의 해쉬값을 할당하거나 할당하도록 지원하여 상기 (x1) 내지 (x3)를 수행하는 것을 특징으로 하는 방법.

청구항 7

제1항에 있어서,

상기 서버가 상기 특정 해쉬값과 상기 적어도 하나의 이웃 해쉬값을 소정의 제1 데이터 구조로 저장하고, 이후 상기 제1 데이터 구조와 동일한 형태의 제2 데이터구조를 저장하여 관리하는 경우, 상기 제1 데이터 구조와 상기 제2 데이터 구조는 체인 형태로 연결되는 것을 특징으로 하는 방법.

청구항 8

제7항에 있어서,

상기 제1 데이터 구조 및 상기 제2 데이터 구조가 머클 트리인 경우, 상기 제1 데이터 구조의 루트값 또는 상기 루트값의 해쉬값이 상기 제2 데이터 구조의 첫번째 리프 노드에 할당되는 것을 특징으로 하는 방법.

청구항 9

제1항에 있어서,

상기 (a) 단계에서, 아무런 노터리 서비스 요청이 없고,

상기 (b) 단계에서, 상기 소정의 조건이 만족되면, 상기 서버는, 소정의 메시지 데이터가 첫번째 리프 노드와 두번째 리프 노드에 할당된 머클 트리를 생성하거나 생성하도록 지원하고, 상기 머클 트리의 루트값 또는 이를 가공한 값을 상기 데이터베이스에 등록하거나 등록하도록 지원하는 것을 특징으로 하는 방법.

청구항 10

제1항에 있어서,

상기 특정 파일에 대한 노터리 서비스는,

상기 특정 파일에 대한 공증 서비스 또는 상기 특정 파일에 대한 파기 서비스인 것을 특징으로 하는 방법.

청구항 11

제1항에 있어서,

상기 데이터베이스는 가상 화폐의 블록체인 또는 상기 서버에 의해 관리되는 데이터베이스인 것을 특징으로 하는 방법.

청구항 12

노터리(notary) 서비스를 사용하여 기록된 파일에 대한 검증을 수행하는 방법에 있어서,

(a) 기준에 서버가 특정 파일에 대한 노터리 서비스 요청을 받았을 때, 제1 특정 해쉬값 - 상기 제1 특정 해쉬값은, 상기 특정 파일의 message digest에 대하여 특정 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 소정의 조건 하에서 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써

생성되는 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값을 데이터베이스에 등록하고 있는 상태에서, 상기 서버가, 상기 특정 파일과 관련된 검증 요청을 획득하는 단계,

(b) 상기 검증 요청에 포함된 입력 데이터를 사용하여 생성된 제2 대표 해쉬값 또는 상기 제2 대표 해쉬값을 가공한 값이 상기 데이터베이스에 등록된 상기 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값과 대응되면, 상기 서버가, 상기 특정 파일과 관련된 검증이 이루어진 것으로 판단하거나 판단하도록 지원하는 단계를 포함하는 방법.

청구항 13

제12항에 있어서,

상기 (a) 단계는,

(a1) 상기 특정 파일과 관련된 검증 요청이 획득되면, 상기 서버가, 상기 특정 파일과 관련된 소정의 transaction ID를 참조하는 단계, 및

(a2) 상기 서버가, 상기 transaction ID를 사용하여 상기 데이터베이스로부터 OP 메시지를 획득하는 단계를 포함하고,

상기 (b) 단계에서,

상기 검증 요청에 포함된 입력 데이터를 사용하여 생성된 상기 제2 대표 해쉬값 또는 상기 제2 대표 해쉬값을 가공한 값이 상기 OP 메시지에 포함된 상기 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값과 대응되면, 상기 서버가, 상기 특정 파일과 관련된 검증이 이루어진 것으로 판단하거나 판단하도록 지원하는 것을 특징으로 하는 방법.

청구항 14

제13항에 있어서,

상기 (a1) 단계에서,

상기 특정 파일과 관련된 검증 요청이 획득되면, 상기 서버는, 상기 입력 데이터와 관련된 머클 트리 정보 및 리프 노드 정보를 식별하고, 상기 식별된 머클 트리 정보에 대응되는 상기 소정의 transaction ID를 참조하는 것을 특징으로 하는 방법.

청구항 15

제12항에 있어서,

상기 (a) 단계에서,

상기 입력 데이터는, (i) 상기 특정 파일, (ii) 상기 특정 파일에 대한 message digest, 또는 (iii) 상기 특정 파일에 대한 공증 당시 발급된 ID 중 어느 하나를 포함하는 것을 특징으로 하는 방법.

청구항 16

제12항에 있어서,

상기 제2 대표 해쉬값은, 제2 특정 해쉬값 - 상기 제2 특정 해쉬값은, 상기 검증 요청에 포함된 입력 데이터에 관한 상기 특정 파일의 message digest에 대하여 특정 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써 생성되는 것을 특징으로 하는 방법.

청구항 17

제16항에 있어서,

상기 제2 대표 해쉬값은,

상기 제2 특정 해쉬값이 특정 리프 노드에 할당된 머클 트리(merkle tree)에서 상기 제2 특정 해쉬값 및 상기 제2 특정 해쉬값과 매칭되는 적어도 하나의 다른 리프 노드에 할당된 해쉬값을 연산하여 생성되는 것을 특징으로

로 하는 방법.

청구항 18

제17항에 있어서,

(x1) 상기 서버는, (i) 상기 제2 특정 해쉬값과 (ii) 상기 제2 특정 해쉬값이 할당된 노드의 형제 노드에 할당된 해쉬값을 연산하거나 연산하도록 지원하고, 상기 연산값에 대한 해쉬값을 상기 노드의 부모 노드에 할당하거나 할당하도록 지원하며,

(x2) 상기 부모 노드가 상기 머클 트리의 루트 노드이면, 상기 부모 노드에 할당된 해쉬값을 상기 제2 대표 해쉬값으로서 상기 OP 메시지에 포함된 값과 비교하거나 비교하도록 지원하고,

(x3) 상기 부모 노드가 상기 머클 트리의 루트 노드가 아니면, 상기 부모 노드에 할당된 해쉬값을 상기 제2 특정 해쉬값으로 하여 상기 (x1) 내지 (x3)를 반복하여 수행하는 것을 특징으로 하는 방법.

청구항 19

제12항에 있어서,

상기 데이터베이스는 가상 화폐의 블록체인 또는 상기 서버에 의해 관리되는 데이터베이스인 것을 특징으로 하는 방법.

청구항 20

파일에 대한 노터리(notary) 서비스를 제공하는 서버에 있어서,

특정 파일에 대한 노터리 서비스 요청을 획득하는 통신부,

(i) 상기 특정 파일에 대한 노터리 서비스 요청이 획득되면, hash 함수를 사용하여 상기 특정 파일의 message digest를 생성하거나 생성하도록 지원하고, (ii) 소정의 조건이 만족되면, 특정 해쉬값 - 상기 특정 해쉬값은, 상기 message digest에 대하여 특정 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써 생성되는 대표 해쉬값 또는 상기 대표 해쉬값을 가공한 값을 데이터베이스에 등록하거나 등록하도록 지원하는 프로세서

를 포함하는 서버.

청구항 21

제20항에 있어서,

상기 소정의 조건은,

(i) 복수의 노터리 서비스 요청이 이루어져 message digest가 소정의 개수만큼 획득되는 조건, (ii) 소정 시간이 경과하는 조건, (iii) 블록 체인에서 블록이 생성되는 조건, (iv) 노터리 서비스 요청을 한 사용자 특성에 대한 조건 중 적어도 하나를 포함하는 것을 특징으로 하는 서버.

청구항 22

제20항에 있어서,

상기 프로세서는,

상기 특정 해쉬값이 리프 노드에 할당된 머클 트리(merkle tree)를 생성하거나 생성하도록 지원하고,

상기 소정의 조건이 만족되면, 상기 특정 해쉬값과 매칭되는 적어도 하나의 다른 리프 노드에 할당된 해쉬값을 연산하여 생성되는 상기 대표 해쉬값 또는 상기 대표 해쉬값을 가공한 값을 상기 데이터베이스에 등록하거나 등록하도록 지원하는 것을 특징으로 하는 서버.

청구항 23

제22항에 있어서,

상기 머클 트리가 체인 형태로 연결된 적어도 하나의 머클 트리 중 첫번째 머클 트리인 경우, 상기 머클 트리의

첫번째 리프 노드에는 텍스트, 숫자, 또는 기호로 이루어진 소정의 메시지 데이터가 할당되는 것을 특징으로 하는 서버.

청구항 24

제22항에 있어서,

상기 소정의 조건이 만족되면,

(x1) 상기 프로세서는, (i) 상기 특정 해쉬값과 (ii) 상기 특정 해쉬값이 할당된 노드의 형제 노드에 할당된 해쉬값을 연산하거나 연산하도록 지원하고, 상기 연산값에 대한 해쉬값을 상기 노드의 부모 노드에 할당하거나 할당하도록 지원하며,

(x2) 상기 부모 노드가 상기 머클 트리의 루트 노드이면, 상기 부모 노드에 할당된 해쉬값을 상기 대표 해쉬값으로서 상기 데이터베이스에 등록하거나 등록하도록 지원하고,

(x3) 상기 부모 노드가 상기 머클 트리의 루트 노드가 아니면, 상기 부모 노드에 할당된 해쉬값을 상기 특정 해쉬값으로 하여 상기 (x1) 내지 (x3)를 반복하여 수행하는 것을 특징으로 하는 서버.

청구항 25

제24항에 있어서,

상기 (x1)에서

상기 소정의 조건이 만족되었음에도 상기 특정 해쉬값이 할당된 노드의 형제 노드에 해쉬값이 할당되어 있지 않은 경우, 상기 프로세서는, 상기 형제 노드에 소정의 해쉬값을 할당하거나 할당하도록 지원하여 상기 (x1) 내지 (x3)를 수행하는 것을 특징으로 하는 서버.

청구항 26

제20항에 있어서,

상기 서버가 상기 특정 해쉬값과 상기 적어도 하나의 이웃 해쉬값을 소정의 제1 데이터 구조로 저장하고, 이후 상기 제1 데이터 구조와 동일한 형태의 제2 데이터구조를 저장하여 관리하는 경우, 상기 제1 데이터 구조와 상기 제2 데이터 구조는 체인 형태로 연결되는 것을 특징으로 하는 서버.

청구항 27

제26항에 있어서,

상기 제1 데이터 구조 및 상기 제2 데이터 구조가 머클 트리인 경우, 상기 제1 데이터 구조의 루트값 또는 상기 루트값의 해쉬값이 상기 제2 데이터 구조의 첫번째 리프 노드에 할당되는 것을 특징으로 하는 서버.

청구항 28

제20항에 있어서,

상기 (i)에서, 아무런 노터리 서비스 요청이 없고,

상기 (ii)에서, 상기 소정의 조건이 만족되면, 상기 프로세서는, 소정의 메시지 데이터가 첫번째 리프 노드와 두번째 리프 노드에 할당된 머클 트리를 생성하거나 생성하도록 지원하고, 상기 머클 트리의 루트값 또는 이를 가공한 값을 상기 데이터베이스에 등록하거나 등록하도록 지원하는 것을 특징으로 하는 서버.

청구항 29

제20항에 있어서,

상기 특정 파일에 대한 노터리 서비스는,

상기 특정 파일에 대한 공증 서비스 또는 상기 특정 파일에 대한 파기 서비스인 것을 특징으로 하는 서버.

청구항 30

제20항에 있어서,

상기 데이터베이스는 가상 화폐의 블록체인 또는 상기 서버에 의해 관리되는 데이터베이스인 것을 특징으로 하는 서버.

청구항 31

노터리(notary) 서비스를 사용하여 기록된 파일에 대한 검증을 수행하는 서버에 있어서,

특정 파일과 관련된 검증 요청을 획득하는 통신부, 및

기준에 상기 서버가 상기 특정 파일에 대한 노터리 서비스 요청을 받았을 때, 제1 특정 해쉬값 - 상기 제1 특정 해쉬값은, 상기 특정 파일의 message digest에 대하여 특정 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 소정의 조건 하에서 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써 생성되는 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값을 데이터베이스에 등록하고 있는 상태에서, 상기 특정 파일과 관련된 검증 요청이 획득되고, 상기 검증 요청에 포함된 입력 데이터를 사용하여 생성된 제2 대표 해쉬값 또는 상기 제2 대표 해쉬값을 가공한 값이 상기 데이터베이스에 등록된 상기 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값과 대응되면, 상기 특정 파일과 관련된 검증이 이루어진 것으로 판단하거나 판단하도록 지원하는 프로세서를 포함하는 서버.

청구항 32

제31항에 있어서,

상기 프로세서는,

상기 특정 파일과 관련된 검증 요청이 획득되면, 상기 특정 파일과 관련된 소정의 transaction ID를 참조하고, 상기 transaction ID를 사용하여 상기 데이터베이스로부터 OP 메시지를 획득하거나 획득하도록 지원하고,

상기 검증 요청에 포함된 입력 데이터를 사용하여 생성된 상기 제2 대표 해쉬값 또는 상기 제2 대표 해쉬값을 가공한 값이 상기 OP 메시지에 포함된 상기 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값과 대응되면, 상기 특정 파일과 관련된 검증이 이루어진 것으로 판단하거나 판단하도록 지원하는 것을 특징으로 하는 서버.

청구항 33

제32항에 있어서,

상기 특정 파일과 관련된 검증 요청이 획득되면, 상기 프로세서는, 상기 입력 데이터와 관련된 머클 트리 정보 및 리프 노드 정보를 식별하고, 상기 식별된 머클 트리 정보에 대응되는 상기 소정의 transaction ID를 참조하는 것을 특징으로 하는 서버.

청구항 34

제31항에 있어서,

상기 입력 데이터는, (i) 상기 특정 파일, (ii) 상기 특정 파일에 대한 message digest, 또는 (iii) 상기 특정 파일에 대한 검증 당시 발급된 ID 중 어느 하나를 포함하는 것을 특징으로 하는 서버.

청구항 35

제31항에 있어서,

상기 제2 대표 해쉬값은, 제2 특정 해쉬값 - 상기 제2 특정 해쉬값은, 상기 검증 요청에 포함된 입력 데이터에 관한 상기 특정 파일의 message digest에 대하여 특정 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써 생성되는 것을 특징으로 하는 서버.

청구항 36

제35항에 있어서,

상기 제2 대표 해쉬값은,

상기 제2 특정 해쉬값이 특정 리프 노드에 할당된 머클 트리(merkle tree)에서 상기 제2 특정 해쉬값 및 상기 제2 특정 해쉬값과 매칭되는 적어도 하나의 다른 리프 노드에 할당된 해쉬값을 연산하여 생성되는 것을 특징으로 하는 서버.

청구항 37

제36항에 있어서,

(x1) 상기 프로세서는, (i) 상기 제2 특정 해쉬값과 (ii) 상기 제2 특정 해쉬값이 할당된 노드의 형제 노드에 할당된 해쉬값을 연산하거나 연산하도록 지원하고, 상기 연산값에 대한 해쉬값을 상기 노드의 부모 노드에 할당하거나 할당하도록 지원하며,

(x2) 상기 부모 노드가 상기 머클 트리의 루트 노드이면, 상기 부모 노드에 할당된 해쉬값을 상기 제2 대표 해쉬값으로서 상기 OP 메시지에 포함된 값과 비교하거나 비교하도록 지원하고,

(x3) 상기 부모 노드가 상기 머클 트리의 루트 노드가 아니면, 상기 부모 노드에 할당된 해쉬값을 상기 제2 특정 해쉬값으로 하여 상기 (x1) 내지 (x3)를 반복하여 수행하는 것을 특징으로 하는 서버.

청구항 38

제31항에 있어서,

상기 데이터베이스는 가상 화폐의 블록체인 또는 상기 서버에 의해 관리되는 데이터베이스인 것을 특징으로 하는 서버.

발명의 설명

기술 분야

[0001] 본 발명은, 파일에 대한 노터리 서비스를 제공하고 상기 노터리 서비스를 사용하여 기록된 파일에 대한 검증을 수행하는 방법 및 서버에 관한 것으로, 좀더 상세하게는, (i) 특정 파일에 대한 노터리 서비스 요청이 획득되면, hash 함수를 사용하여 상기 특정 파일의 message digest를 생성하고, 소정의 조건이 만족되면, 특정 해쉬값 - 상기 특정 해쉬값은, 상기 message digest에 대하여 특정 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써 생성되는 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값을 데이터베이스에 등록하는 “노터리 프로세스”를 수행하고, (ii) 상기 특정 파일과 관련된 검증 요청이 획득되면, 상기 검증 요청에 포함된 입력 데이터를 사용하여 생성된 제2 대표 해쉬값 또는 상기 제2 대표 해쉬값을 가공한 값이 상기 데이터베이스에 등록된 상기 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값과 대응되면, 상기 특정 파일과 관련된 검증이 이루어진 것으로 판단하는 “검증 프로세스”를 수행하는, 파일에 대한 노터리 서비스를 제공하고 상기 노터리 서비스를 사용하여 기록된 파일에 대한 검증을 수행하는 방법 및 서버에 관한 것이다.

배경 기술

[0002] 공증(公證)이란, 특정한 사실 또는 법률관계의 존재여부나 내용을 공적으로 증명하는 행위를 말한다. 일반적으로 특정 법률관계에 대하여 공증 촉탁이 있는 경우, 공증인은 일정한 방식에 따라서 증서를 작성하고 기명 날인하여 공증 사무소에 보관한다.

[0003] 그런데, 종래의 공증 방법은 촉탁인이 공증할 서류를 들고 직접 공증 사무소를 방문하여 공증을 받고, 공증 사무소는 이에 대한 증서를 서류 보관소에 보관하는 방식으로 이루어졌으므로 공증을 받는데 시간이 오래 걸리고 공간적인 제약이 있으며 공증 서류의 분실 내지 위/변조 가능성이 있는 문제점이 있었다.

[0004] 이에 발명자는, 가상 화폐의 블록체인에 공증 정보를 기록하여 위/변조가 불가능하고 공간적 제약이 없이 실시간으로 공증을 수행할 수 있는 파일 공증 시스템을 개발하고 이에 대한 특허출원(한국특허출원 제2015-0188978호)을 진행한 바 있다.

- [0005] 다만, 상기 블록체인을 이용한 파일 공증 시스템은 매 파일마다 OP 메시지를 생성하여 블록체인에 기록하므로 transaction이 많아졌을 때, 서비스 제공의 bottle neck 현상이 발생할 수 있고, 매번 transaction fee를 지불해야 하므로 서비스 운영에 많은 비용이 들 수 있는 단점이 있다.
- [0006] 따라서, 위/변조가 불가능하고 실시간 공증이 가능한 상기 파일 공증 시스템의 장점을 살리면서도 이를 보완하여 빠른 서비스 속도를 보장하고 비용도 절감할 수 있는 기술 방안이 요청된다.

발명의 내용

해결하려는 과제

- [0007] 본 발명은 상술한 문제점들을 모두 해결하는 것을 그 목적으로 한다.
- [0008] 또한, 본 발명은 가상 화폐의 블록체인에 공증 정보를 기록하여 실시간으로 공증 및 검증을 수행할 수 있는 파일 공증 서비스를 제공함에 있어, 빠른 서비스 속도를 보장하고 transaction 비용을 절감할 수 있는 것을 다른 목적으로 한다.
- [0009] 또한, 본 발명은 모든 노터리 서비스 정보를 블록체인에 올리는 대신에 이를 이용하여 머클 트리를 구성하여 상기 구성된 머클 트리의 루트값만을 블록체인에 등록함으로써 공증 서비스의 속도를 향상시키고 transaction 비용을 절감할 수 있는 것을 또 다른 목적으로 한다.
- [0010] 또한, 본 발명은, 파일에 대한 검증 요청시 상기와 같이 생성된 머클 트리를 이용하여 검증을 수행함으로써 데이터베이스의 integrity를 좀더 보장할 수 있는 것을 또 다른 목적으로 한다.

과제의 해결 수단

- [0011] 상기 목적을 달성하기 위한 본 발명의 대표적인 구성은 다음과 같다.
- [0012] 본 발명의 일 태양에 따르면, 파일에 대한 노터리(notary) 서비스를 제공하는 방법에 있어서, (a) 특정 파일에 대한 노터리 서비스 요청이 획득되면, 서버가, hash 함수를 사용하여 상기 특정 파일의 message digest를 생성하거나 생성하도록 지원하는 단계, 및 (b) 소정의 조건이 만족되면, 상기 서버가, 특정 해쉬값 - 상기 특정 해쉬값은, 상기 message digest에 대하여 특정 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써 생성되는 대표 해쉬값 또는 상기 대표 해쉬값을 가공한 값을 데이터베이스에 등록하거나 등록하도록 지원하는 단계를 포함하는 방법이 제공된다.
- [0013] 본 발명의 다른 태양에 따르면, 노터리(notary) 서비스를 사용하여 기록된 파일에 대한 검증을 수행하는 방법에 있어서, (a) 기존에 서버가 특정 파일에 대한 노터리 서비스 요청을 받았을 때, 제1 특정 해쉬값 - 상기 제1 특정 해쉬값은, 상기 특정 파일의 message digest에 대하여 특정 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 소정의 조건 하에서 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써 생성되는 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값을 데이터베이스에 등록하고 있는 상태에서, 상기 서버가, 상기 특정 파일과 관련된 검증 요청을 획득하는 단계, (b) 상기 검증 요청에 포함된 입력 데이터를 사용하여 생성된 제2 대표 해쉬값 또는 상기 제2 대표 해쉬값을 가공한 값이 상기 데이터베이스에 등록된 상기 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값과 대응되면, 상기 서버가, 상기 특정 파일과 관련된 검증이 이루어진 것으로 판단하거나 판단하도록 지원하는 단계를 포함하는 방법이 제공된다.
- [0014] 본 발명의 또 다른 태양에 따르면, 파일에 대한 노터리(notary) 서비스를 제공하는 서버에 있어서, 특정 파일에 대한 노터리 서비스 요청을 획득하는 통신부, (i) 상기 특정 파일에 대한 노터리 서비스 요청이 획득되면, hash 함수를 사용하여 상기 특정 파일의 message digest를 생성하거나 생성하도록 지원하고, (ii) 소정의 조건이 만족되면, 특정 해쉬값 - 상기 특정 해쉬값은, 상기 message digest에 대하여 특정 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써 생성되는 대표 해쉬값 또는 상기 대표 해쉬값을 가공한 값을 데이터베이스에 등록하거나 등록하도록 지원하는 프로세서를 포함하는 서버가 제공된다.
- [0015] 본 발명의 또 다른 태양에 따르면, 노터리(notary) 서비스를 사용하여 기록된 파일에 대한 검증을 수행하는 서버에 있어서, 특정 파일과 관련된 검증 요청을 획득하는 통신부, 및 기존에 상기 서버가 상기 특정 파일에 대한 노터리 서비스 요청을 받았을 때, 제1 특정 해쉬값 - 상기 제1 특정 해쉬값은, 상기 특정 파일의 message digest에 대하여 특정 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 -

과 소정의 조건 하에서 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써 생성되는 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값을 데이터베이스에 등록하고 있는 상태에서, 상기 특정 파일과 관련된 검증 요청이 획득되고, 상기 검증 요청에 포함된 입력 데이터를 사용하여 생성된 제2 대표 해쉬값 또는 상기 제2 대표 해쉬값을 가공한 값이 상기 데이터베이스에 등록된 상기 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값과 대응되면, 상기 특정 파일과 관련된 검증이 이루어진 것으로 판단하거나 판단하도록 지원하는 프로세서를 포함하는 서버가 제공된다.

발명의 효과

- [0016] 본 발명에 의하면, 다음과 같은 효과가 있다.
- [0017] 본 발명은 가상 화폐의 블록체인에 공증 정보를 기록하여 실시간으로 공증 및 검증을 수행할 수 있는 파일 공증 시스템을 제공함에 있어, 빠른 서비스 속도를 보장하고 transaction 비용을 절감할 수 있는 효과가 있다.
- [0018] 또한, 본 발명은 모든 노터리 서비스 정보를 블록체인에 올리는 대신에 이를 이용하여 머클 트리를 구성하여 상기 구성된 머클 트리의 루트값만을 블록체인에 등록함으로써 공증 서비스의 속도를 향상시키고 transaction 비용을 절감할 수 있는 효과가 있다.
- [0019] 또한, 본 발명은, 파일에 대한 검증 요청시 상기와 같이 생성된 머클 트리를 이용하여 검증을 수행함으로써 데이터베이스의 integrity를 좀더 보장할 수 있는 효과가 있다.

도면의 간단한 설명

- [0020] 도 1은 본 발명의 일 실시예에 따른 파일에 대한 노터리 서비스를 제공하고 상기 노터리 서비스를 사용하여 기록된 파일에 대한 검증을 수행하는 서버의 구성을 도시한 블록도이다.
- 도 2는 단일 사용자의 요청에 따라 파일에 대한 노터리 서비스를 수행하는 과정을 도시한 시퀀스 다이어그램이다.
- 도 3 및 4는 본 발명의 일 실시예에 따라 생성된 머클 트리의 예를 도시한 도면이다.
- 도 5는 단일 사용자의 요청에 따라 파일에 대한 검증 서비스를 수행하는 과정을 도시한 시퀀스 다이어그램이다.
- 도 6은 다중 사용자의 노터리 서비스 요청에 대해 병렬적으로 서비스를 수행하는 parallel signing 과정을 나타낸 시퀀스 다이어그램이다.
- 도 7은 다중 사용자의 노터리 서비스 요청에 대해 순차적으로 서비스를 수행하는 serial signing 과정을 나타낸 시퀀스 다이어그램이다.
- 도 8은 다중 사용자의 서명으로 공증이 이루어진 경우 공증된 파일에 대해 검증을 수행하는 과정을 도시한 시퀀스 다이어그램이다.

발명을 실시하기 위한 구체적인 내용

- [0021] 후술하는 본 발명에 대한 상세한 설명은, 본 발명이 실시될 수 있는 특정 실시예를 예시로서 도시하는 첨부 도면을 참조한다. 이들 실시예는 당업자가 본 발명을 실시할 수 있기에 충분하도록 상세히 설명된다. 본 발명의 다양한 실시예는 서로 다르지만 상호 배타적일 필요는 없음이 이해되어야 한다. 예를 들어, 여기에 기재되어 있는 특정 형상, 구조 및 특성은 일 실시예에 관련하여 본 발명의 정신 및 범위를 벗어나지 않으면서 다른 실시예로 구현될 수 있다. 또한, 각각의 개시된 실시예 내의 개별 구성요소의 위치 또는 배치는 본 발명의 정신 및 범위를 벗어나지 않으면서 변경될 수 있음이 이해되어야 한다. 따라서, 후술하는 상세한 설명은 한정적인 의미로서 취하려는 것이 아니며, 본 발명의 범위는, 적절하게 설명된다면, 그 청구항들이 주장하는 것과 균등한 모든 범위와 더불어 첨부된 청구항에 의해서만 한정된다. 도면에서 유사한 참조부호는 여러 측면에 걸쳐서 동일하거나 유사한 기능을 지칭한다.
- [0022] 도 1은 본 발명의 일 실시예에 따른 파일에 대한 노터리 서비스를 제공하고 상기 노터리 서비스를 사용하여 기록된 파일에 대한 검증을 수행하는 서버의 구성을 도시한 블록도이고, 도 2는 단일 사용자의 요청에 따라 파일에 대한 노터리 서비스를 수행하는 과정을 도시한 시퀀스 다이어그램이다.
- [0023] 각 실시예에 대한 설명에 앞서 본 명세서에서 사용되는 표현식을 다음과 같이 정의한다. ':'의 왼쪽은 표현식

을 나타내며 오른쪽은 표현식의 정의를 나타낸다.

- [0025] <표현식 정의>
- [0026] $PrivX$: Private key of X
- [0027] $PubX$: Public key of X
- [0028] $AESX$: AES secret key of X
- [0029] $Enc_{PrivX}(Y)$: Output of encryption of Y using $PrivX$
- [0030] $Dec_{PubX}(Y)$: Output of decryption of Y using $PubX$
- [0031] $Enc_{AESX}(Y)$: Output of AES encryption of Y using $AESX$
- [0032] $Dec_{AESX}(Y)$: Output of AES decryption of Y using $AESX$
- [0033] $Hash(Y)$: Hash value of Y
- [0035] 우선, 단일 사용자의 요청에 대해 파일에 대해 노터리 서비스를 수행하는 과정을 도 1 및 2를 참조로 하여 설명한다.
- [0036] 도 1을 참조로 하면, 본 발명의 일 실시예에 따른 파일에 대한 노터리 서비스를 수행하는 서버(100)는 통신부(110), 프로세서(120)를 포함한다. 각 실시예에서 설명되는 서버는 상이하게 구성될 수도 있을 것이나, 설명의 편의상 동일한 서버(100)를 통해 서비스가 이루어지는 것으로 설명한다.
- [0037] 통신부(110)는 특정 사용자(또는 사용자 단말), 데이터베이스(200)에 데이터를 송수신하는 구성이다. 우선, 통신부(110)는 특정 파일에 대한 노터리 서비스 요청을 획득할 수 있다(S210). 도 2에 도시된 것처럼 특정 파일에 대한 노터리 서비스 요청이 획득되는 경우 특정 파일(File)과 노터리 서비스 요청을 한 특정 사용자의 public key(PubU)가 획득될 수 있다. 통신부(110)는, 이에 대한 응답을 할 수 있는데(S220), 요청(request)과 대응되는 응답(response)는 하나의 TCP 세션 동안 이루어질 수 있다.
- [0038] 프로세서(120)는, 상기 특정 파일에 대한 노터리 서비스 요청이 획득되면, hash 함수를 사용하여 상기 특정 파일의 message digest(MD=Hash(File))를 생성하거나 생성하도록 지원할 수 있다(S215). 여기서 hash 함수의 종류는 한정되지 않는다. 예를 들어, hash 함수는 triple sha256일 수 있다. 또한, 본 명세서에서 hash 함수는 일반적으로는 원본 파일을 변형시키기 위한 함수를 의미하지만, 극단적인 경우로서, 특정 파일에 대해 아무런 처리를 하지 않고 단순히 저장하는 함수를 포함할 수도 있다. 이 경우, hash 함수가 적용되어 생성된 message digest는 특정 파일과 동일할 수도 있을 것이다.
- [0039] 상기 S220 단계에서 상기 특정 파일에 대한 노터리 서비스 요청에 대한 응답으로 상기 생성된 message digest가 특정 파일에 대한 노터리 서비스 요청을 한 특정 사용자에게 전달될 수 있다. 상기 특정 사용자가 사용자 단말(미도시)을 통해 특정 파일에 대한 노터리 서비스 요청을 한 경우라면, 사용자 단말로 상기 생성된 message digest가 전송될 수 있다.
- [0040] 다음으로 통신부(110)는 상기 특정 사용자로부터 상기 특정 사용자의 private key(PrivU)로 인코딩된 message digest(EncPrivU(MD))를 획득할 수 있다(S225). 이 경우, S220 단계의 응답 후 TCP 세션이 끊어질 수 있으므로 특정 사용자로부터 message digest와 특정 사용자의 public key(PubU)도 수신될 수 있다. 참고로, 본 명세서에서는 private key나 public key를 이용하여 MD를 인코딩하는 표현식(EncPriv(MD), EncPub(MD))을 기준으로 설명을 하나 이는 SigPriv(File), SigPub(File)로 대체될 수 있다. 또한, message digest가 두 번 이상 생성되는 경우, SigPriv(MD), SigPub(MD)로 표현될 수도 있다.
- [0041] 또한, 프로세서(120)는 특정 사용자의 private key로 인코딩된 message digest가 획득되고, 상기 특정 사용자의 private key로 인코딩된 message digest를 상기 특정 사용자의 public key로 디코딩한 정보(A, DecPubU(EncPrivU(MD)))와 상기 생성된 message digest(B, MD)가 일치하면(S230-2), 상기 특정 사용자의 private key 및 서버(100)의 private key(PrivC)로 인코딩된 message digest의 해쉬값을 생성하거나 생성하도록

록 지원할 수 있고(S240), 생성된 해쉬값을 데이터베이스(200)에 등록하거나 등록하도록 지원할 수 있다(S250). 이처럼 해쉬값의 형태로 데이터베이스(200)에 등록하는 이유는 데이터베이스(200)의 저장 용량 제한 등으로 인해 데이터의 사이즈를 줄이기 위함이다. 여기서, 데이터베이스(200)는 가상 화폐의 블록체인일 수 있다. 예를 들어, 비트코인 블록체인일 수 있다.

[0042] 참고로, 도 2에서 상기 A와 B가 일치하는지를 나타내는 표현식은 'DecPubU(EncPrivU(MD)) == MD'로 나타냈으나, 이러한 표현식에 한정되는 것은 아니다. 동일한 관계를 'True == VerPubU(SigPrivU(File))'로 대체하여 표현할 수 있다. 또한, message digest가 두 번 이상 생성되는 경우, 'True == VerPubU(SigPrivU(MD))'로 표현될 수도 있다.

[0043] 또한, 상기 A와 상기 B가 일치하는 경우, 프로세서(120)는, 상기 특정 사용자의 private key, 상기 서버(100)의 private key 순으로 인코딩된 message digest의 해쉬값(Hash[EncPrivC(EncPrivU(MD))])을 상기 데이터베이스(200)에 등록하거나 등록하도록 지원할 수 있다. 즉, 특정 사용자(촉탁인)이 먼저 자신의 private key로 서명을 하고 공증인(서버)이 서버(100)의 private key로 서명을 한 후 그 해쉬값을 데이터베이스(200)에 등록할 수 있는 것이다. 그러나, 서명의 순서는 이와 다르게 구현될 수도 있다. 특히, 특정 사용자와 서버(100) 간의 세션이 유지되는 경우라면 서버(100)가 먼저 서명을 하고 특정 사용자가 다음에 서명을 하도록 구현될 수도 있을 것이다.

[0044] 상기 특정 사용자의 private key 및 상기 서버(100)의 private key로 인코딩된 message digest의 해쉬값이 데이터베이스(200)에 등록되는 경우, 통신부(110)는 데이터베이스(200)로부터 상기 등록된 해쉬값의 상기 데이터베이스(200) 상의 위치 정보를 나타내는 transaction ID(TxID)를 획득할 수 있다(S255).

[0045] 한편, 상기 A와 상기 B가 일치하지 않으면(S230-1), 프로세서(120)는 상기 특정 사용자가 상기 특정 파일과 관련된 사람(예를 들어, 당사자)이 아니라는 이유로 특정 사용자에게 등록 실패 응답을 전달할 수 있다(S235).

[0046] 또한, 프로세서(120)는 상기 A와 상기 B가 일치하면, 상기 특정 파일을 저장하거나 저장하도록 지원할 수 있다(S245). 이때 특정 파일은 AES(Advanced Encryption Standard)와 같은 대칭키 암호 알고리즘으로 암호화될 수 있다. 이 경우 비밀키는 사용자의 public key로부터 가공된 특정한 비밀키일 수 있으나 이에 한정되는 것은 아니다.

[0047] 프로세서(120)는 특정 사용자의 public key(PubU), 상기 특정 파일의 message digest(MD), 상기 특정 사용자의 private key로 인코딩된 message digest(PrivU(MD)), 상기 획득된 transaction ID(TxID)를 저장하거나 저장하도록 지원할 수 있다. 또한, 이 밖에도 상기 특정 사용자의 private key, 상기 서버(100)의 private key 순으로 인코딩된 message digest(EncPrivC(EncPrivU(MD))) 및 상기 인코딩된 message digest의 해쉬값(Hash[EncPrivC(EncPrivU(MD))])을 더 저장하거나 저장하도록 지원할 수도 있다.

[0048] 위와 같이 공증 정보가 데이터베이스(200)에 등록되면, 통신부(110)는 특정 사용자에게 등록이 성공했음을 나타내는 응답을 전달할 수 있다(S260).

[0049] 그런데, 전술한 실시예의 경우, 매 파일마다 OP 메시지를 생성하여 블록체인에 기록하므로 transaction 요청이 많아졌을 때, 서비스 제공의 bottle neck 현상이 발생될 수 있고, 매번 transaction fee를 지불해야 하므로 서비스 운영에 많은 비용이 들 수 있는 단점이 있다.

[0050] 이하에서는 위/변조가 불가능하고 실시간 공증이 가능한 상기 공증 시스템의 장점을 살리면서도 이를 보완하여 빠른 서비스 속도를 보장하고 비용도 절감할 수 있는 공증 시스템을 설명한다. 이러한 공증 시스템을 전술한 실시예와 달리 서버(100)에 의해 관리되는 별도의 데이터 구조를 이용한다.

[0051] 전술한 프로세서(120)는 통신부(110)에 의해 특정 파일에 대한 노터리 서비스 요청이 획득되면, hash 함수를 사용하여 상기 특정 파일의 message digest를 생성하거나 생성하도록 지원할 수 있다. 이에 대해서는 전술한 실시예에서 자세히 설명하였으므로 중복 설명은 생략한다. 여기서, 특정 파일에 대한 노터리 서비스는 전술한 실시예와 달리 상기 특정 파일에 대한 공증 서비스 또는 상기 특정 파일에 대한 파기(revocation) 서비스일 수 있다.

[0052] 그리고, 프로세서(120)는, 소정의 조건이 만족되면, 특정 해쉬값 - 상기 특정 해쉬값은, 상기 message digest에 대하여 특정 사용자의 private key 및 상기 서버(100)의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써 생성되는 대표 해쉬값 또는 상기 대표 해쉬값을 가공한 값을 데이터베이스에 등록하거나 등록하도록 지원할 수 있다. 즉, 프로세서(120)는 전술한 실시예의 S240 단계와 같

이 상기 특정 사용자의 private key(PrivU)와 상기 서버(100)의 private key(PrivC)로 인코딩한 결과에 대한 해쉬값을 데이터베이스에 등록하는 대신, 소정의 조건이 만족되면, 상기 특정 사용자의 private key(PrivU)와 상기 서버(100)의 private key(PrivC)로 인코딩한 결과에 대한 해쉬값과 적어도 하나의 이웃 해쉬값을 연산하여 대표 해쉬값을 생성하거나 생성하도록 지원하고, 생성된 대표 해쉬값 또는 이를 가공한 값을 데이터베이스에 등록하거나 등록하도록 지원할 수 있다. 여기서, 데이터베이스는 전술한 데이터베이스(200)와 같이 상기 서버(100)가 관리하지 않는 데이터베이스(예를 들어, 가상 화폐의 블록체인)일 수도 있으나 이와 달리 상기 서버(100)에 의해 관리되는 데이터베이스일 수도 있다. 본 명세서에서 서버(100)가 관리하지 않는 데이터베이스(200)를 중심으로 설명을 하나, 본 발명의 기술 사상은 서버(100)에 의해 관리되는 데이터베이스의 경우도 적용될 수 있다.

[0053] 특정 해쉬값과 적어도 하나의 이웃 해쉬값의 연산은 다양한 함수에 의해 수행될 수 있다. 특정 해쉬값을 input 이라고 나타내고, 적어도 하나의 이웃 해쉬값을 $x_1, x_2, \dots, x_n$ 이라고 나타낼 때, 대표 해쉬값 t 는 다음 수식과 같이 나타낼 수 있다.

[0055] <수학식>

[0056]
$$t = \text{hash}(\text{function}(\text{input}, x_1, x_2, \dots, x_n))$$

[0058] 이때 서버(100)는 상기 특정 해쉬값과 상기 적어도 하나의 이웃 해쉬값을 소정의 데이터 구조로 저장하여 관리할 수 있다. 여기서, 데이터 구조는 다양할 수 있는데, 일 예로 머클 트리(merkel tree) 구조가 될 수도 있다. 이 경우, 상기 특정 해쉬값과 적어도 하나의 이웃 해쉬값의 연산은 머클 트리(merkel tree)를 통해 이루어질 수 있다.

[0059] 즉, 프로세서(120)는 상기 특정 해쉬값이 특정 리프 노드에 할당된 머클 트리(merkle tree)를 생성하거나 생성하도록 지원할 수 있고, 상기 소정의 조건이 만족되면, 상기 특정 해쉬값과 매칭되는 적어도 하나의 다른 리프 노드에 할당된 해쉬값을 연산하여 데이터베이스(200)에 등록하거나 등록하도록 지원할 수 있다.

[0060] 좀더 구체적으로 설명하면, (x1) 프로세서(120)는, (i) 상기 특정 해쉬값과 (ii) 상기 특정 해쉬값이 할당된 노드의 형제 노드에 할당된 해쉬값을 연산하거나 연산하도록 지원하고, 상기 연산값에 대한 해쉬값을 상기 노드의 부모 노드에 할당하거나 할당하도록 지원할 수 있다. (x2) 만일, 상기 부모 노드가 상기 머클 트리의 루트 노드이면, 상기 부모 노드에 할당된 해쉬값이 상기 대표 해쉬값이 된다. (x3) 반면, 상기 부모 노드가 상기 머클 트리의 루트 노드가 아니면, 프로세서(120)는, 상기 부모 노드에 할당된 해쉬값을 상기 특정 해쉬값으로 하여 상기 (x1) 내지 (x3)를 반복하여 수행한다.

[0061] 그리고, 프로세서(120)는 최종적으로 머클 트리의 루트 노드에 할당된 해쉬값을 대표 해쉬값으로서 데이터베이스(200)에 등록하거나 등록하도록 지원한다. 이때, 대표 해쉬값을 가공한 값이 등록될 수도 있다. 예를 들어, 대표 해쉬값에 hex 연산이 수행된 결과값이 등록될 수 있다.

[0062] 한편, 상기 서버(100)가 상기 특정 해쉬값과 상기 적어도 하나의 이웃 해쉬값을 소정의 제1 데이터 구조로 저장하고, 이후 상기 제1 데이터 구조와 동일한 형태의 제2 데이터구조를 저장하여 관리하는 경우, 상기 제1 데이터 구조와 상기 제2 데이터 구조는 체인 형태로 연결될 수 있다.

[0063] 특히, 상술한 예에서와 같이 상기 제1 데이터 구조 및 상기 제2 데이터 구조가 머클 트리인 경우, 상기 제1 데이터 구조의 루트값 또는 상기 루트값의 해쉬값이 상기 제2 데이터 구조의 첫번째 리프 노드에 할당될 수 있다.

[0064] 또한, 제2 데이터 구조를 생성할 때는 제1 데이터 구조에 대한 검증이 이루어짐으로써 데이터 integrity가 좀더 보장될 수 있다. 제2 데이터 구조의 검증에 대해서는 후술하기로 한다.

[0065] 또한, 체인 형태로 연결된 적어도 하나의 머클 트리 중 첫번째 머클 트리의 경우, 상기 첫번째 머클 트리의 첫번째 리프 노드에는 텍스트, 숫자, 또는 기호로 이루어진 소정의 메시지 데이터의 해쉬값 또는 이를 가공한 값이 할당될 수 있다. 예를 들어, 머클 트리 생성시 서버(100)에 의해 최초로 부여된 입력 메시지의 해쉬값이 할당될 수 있다.

[0066] 도 3 및 4는 본 발명의 일 실시예에 따라 생성된 머클 트리의 예를 도시한 도면이다.

- [0067] 도 3에서는 리프 노드의 개수가 $4(2^2)$ 개인 머클 트리가 도시된다. 도시된 머클 트리는 첫번째 머클 트리이기 때문에($tree_id=0$), 첫번째 리프 노드인 h1 노드에는 소정의 메시지 데이터 “Coinplug 2016-01-01”의 해쉬값($triplesha256digest(coinplug_unique_message)$)이 할당되었음을 알 수 있다. 특정 파일에 대한 노터리 서비스 요청이 있는 경우, 프로세서(120)는 현재 구성 중인 머클 트리의 가장 마지막 리프 노드의 다음 리프 노드를 생성하여 특정 해쉬값 또는 특정 해쉬값을 가공한 값을 할당하거나 할당하도록 지원한다. 예를 들어, 도 3의 머클 트리에서 이전 단계에서 두번째 리프 노드인 h1 노드까지 값 할당이 완료된 경우, 다음 리프 노드인 h2 노드를 생성하여 특정 해쉬값 또는 특정 해쉬값을 가공한 값($triplesha256digest(input2)$)을 할당할 수 있다. 또한, 프로세서(120)는 (i) 특정 해쉬값과 (ii) 상기 특정 해쉬값이 할당된 세번째 리프 노드인 h2 노드의 형제 노드인 h3 노드에 할당된 해쉬값을 연산하거나 연산하도록 지원할 수 있다. 상기 연산값에 대한 해쉬값은 h2 노드와 h3 노드의 부모 노드(h23 노드)에 할당된다. 부모 노드(h23 노드)가 머클 트리의 루트 노드가 아니므로 프로세서(120)는 상기 h23 노드에 할당된 해쉬값을 상기 특정 해쉬값으로 하여 상기 과정을 반복하여 수행할 수 있다. 즉, h23 노드에 할당된 해쉬값을 특정 해쉬값으로 하고, h23 노드에 할당된 해쉬값과 h01 노드에 할당된 해쉬값을 연산하여 h23 노드와 h01 노드의 부모 노드(h0123 노드)에 할당할 수 있다. 이때, h0123 노드가 머클 트리의 루트 노드이므로 프로세서(120)는, h0123 노드에 할당된 해쉬값을 가공한 값($hex(h\{node_index\})$)을 데이터베이스(200)에 등록하거나 등록하도록 지원할 수 있다.
- [0068] 한편, 전술한 소정의 조건이란, (i) 복수의 노터리 서비스 요청이 이루어져 message digest가 소정의 개수만큼 획득되는 조건, (ii) 소정 시간이 경과하는 조건, (iii) 블록 체인에서 블록이 생성되는 조건, (iv) 노터리 서비스 요청을 한 사용자 특성에 대한 조건 중 적어도 하나를 포함할 수 있다.
- [0069] 예를 들어, 복수의 노터리 서비스 요청이 이루어져 message digest가 머클 트리의 리프 노드 수만큼 획득되면, 각각의 message digest에 대하여 각 사용자의 private key 및 서버(100)의 private key로 인코딩한 결과에 대한 해쉬값이 전술한 머클 트리의 입력값(리프 노드에 할당된 값)이 될 수 있다.
- [0070] 또한, 프로세서(120)는 소정 시간 단위로 전술한 머클 트리의 루트값을 생성할 수 있다(상기 (ii) 조건). 이 경우 프로세서(120)는 소정의 시간이 경과되면 그때까지의 입력값을 이용하여 머클 트리를 생성하고 머클 트리의 루트값을 데이터베이스(200)에 등록하거나 등록하도록 지원할 수 있다.
- [0071] 그런데, 이 경우에는 소정 시간이 경과하였음에도 머클 트리의 특정 해쉬값이 할당된 노드의 형제 노드에 값이 할당되지 않을 수 있다. 이처럼 소정의 조건이 만족되었음에도 상기 특정 해쉬값이 할당된 노드의 형제 노드에 해쉬값이 할당되어 있지 않은 경우, 프로세서(120)는, 상기 형제 노드에 소정의 해쉬값을 할당하거나 할당하도록 지원하여 전술한 방식으로 머클 트리의 루트값이 산출되도록 할 수 있다. 예를 들어, 프로세서(120)는 상기 특정 해쉬값을 복제하여 상기 형제 노드에 할당하거나 할당하도록 지원할 수 있다.
- [0072] 노터리 서비스 요청을 한 사용자 특성이란, 노터리 서비스 요청을 한 사용자가 지급한 비용 정보, 노터리 서비스가 이루어지는 시간대 정보, 노터리 서비스가 이루어지는 지역 정보, 노터리 서비스 요청을 한 사용자 타입 정보 중 적어도 일부가 될 수 있다. 다만, 여기서 기재한 것에 한정할 것은 아니고, 통상적으로 인정되는 차등적 서비스가 제공될 수 있는 다양한 조건 정보를 포함한다.
- [0073] 한편, 새로운 머클 트리 생성이 시작되고, 아무런 노터리 서비스 요청이 없는 상태에서 상기 소정의 조건이 만족되면, 상기 프로세서(120)는, 소정의 메시지 데이터가 첫번째 리프 노드와 두번째 리프 노드에 할당된 머클 트리를 생성하거나 생성하도록 지원하고, 상기 머클 트리의 루트값 또는 이를 가공한 값을 데이터베이스(200)에 등록하거나 등록하도록 지원할 수 있다. 이 경우에는 리프 노드 2개짜리 머클 트리가 생성될 수 있는 것이다.
- [0074] 한편, 전술한 것처럼 서버(100)가 상기 특정 해쉬값과 상기 적어도 하나의 이웃 해쉬값을 소정의 제1 데이터 구조로 저장하고, 이후 상기 제1 데이터 구조와 동일한 형태의 제2 데이터구조를 저장하여 관리하는 경우, 상기 제1 데이터 구조와 상기 제2 데이터 구조는 체인 형태로 연결될 수 있다. 특히, 상기 제1 데이터 구조 및 상기 제2 데이터 구조가 머클 트리인 경우, 상기 제1 데이터 구조의 루트값 또는 상기 루트값의 해쉬값이 상기 제2 데이터 구조의 첫번째 리프 노드에 할당될 수 있다.
- [0075] 도 4는 본 발명의 일 실시예에 따라 상기 제2 데이터 구조로서 생성된 머클 트리를 도시한 도면이다.
- [0076] 도 4를 참조하면, 도 3의 머클 트리($tree_id=0$)의 루트값($hex(h0123)$)이 새로운 머클 트리의 첫번째 리프 노드(h4 노드)에 할당되었음을 알 수 있다($triplesha256digest(input4)$). 본 발명은 이와 같이 트랜잭션 발생시 생성되는 복수의 데이터 구조를 연결함으로써 중간에 데이터가 변조가 발생하는 경우라도 쉽게 트래킹이 가능하여

데이터 integrity를 향상시키는 장점을 가진다.

- [0077] 다음으로, 단일 사용자의 요청에 따라 파일에 대해 검증을 수행하는 과정을 설명한다.
- [0078] 본 명세서에서 설명하는 검증은 당사자 또는 제3자가 요청할 수 있으며, 서버(100)가 관리하는 별도의 데이터 구조를 이용하는 경우와 그렇지 않은 경우로 나눌 수 있다. 용어로는 audit, verification이 모두 사용될 수 있다. 우선, 서버(100)가 관리하는 별도의 데이터 구조를 사용하지 않는 경우를 먼저 설명한다.
- [0079] 파일의 검증은 전술한 서버(100)의 정보를 참조할 수 있는 별도의 장치에 의해 수행될 수도 있고 전술한 서버(100)에 의해 수행될 수도 있다. 이하에서는 편의상 전술한 서버(100)에 의해 파일의 검증이 이루어지는 것을 전제로 설명을 한다.
- [0080] 도 5는 단일 사용자의 요청에 따라 파일에 대한 검증 서비스를 수행하는 과정을 도시한 시퀀스 다이어그램이다.
- [0081] 도 5를 참조하면, 전술한 예와 유사하게 통신부(110)는 특정 파일에 대한 검증 요청을 획득할 수 있다(S510). 도 5에 도시된 것처럼 특정 파일에 대한 검증 요청이 획득되는 경우 특정 파일(File)과 검증 요청을 한 특정 사용자의 public key(PubU)가 획득될 수 있다. 통신부(110)는, 이에 대한 응답을 할 수 있는데, 요청(request)과 대응되는 응답(response)는 하나의 TCP 세션 동안 이루어질 수 있다.
- [0082] 프로세서(120)는 전술한 바와 같이 기존에 서버(100)가 특정 파일에 대한 노터리 서비스 요청을 받았을 때, 상기 특정 파일과 관련된 사람이 특정 사용자임이 감지되어, 상기 특정 사용자의 private key 및 서버(100)의 private key로 인코딩된 message digest의 해쉬값을 데이터베이스(200)에 저장하고 transaction ID를 받아서 관리하고 있는 상태에서, 상기 특정 파일에 대한 검증 요청이 획득되면, 상기 특정 파일의 message digest(MD) 및 상기 특정 사용자의 public key 중 적어도 하나에 대응되는 transaction ID를 참조할 수 있다. 이 경우 transaction ID(TxID)는 서버(100) 또는 서버(100)가 참조할 수 있는 장치에 저장되어 있을 수 있다.
- [0083] 상기 S510 단계에서 통신부(110)가 상기 특정 파일에 대한 검증 요청을 받는 경우에 있어서, 상기 특정 파일에 대한 검증을 위해 상기 특정 파일이 획득되면, 프로세서(120)는 hash 함수를 사용하여 상기 특정 파일의 message digest(MD=Hash(File))를 생성하거나 생성하도록 지원할 수 있다(S515). 이 경우 프로세서(120)는 이렇게 생성된 특정 파일의 message digest 및 상기 특정 사용자의 public key 중 적어도 하나에 대응되는 transaction ID를 참조한다.
- [0084] 상기 특정 파일에 대한 검증 요청이 있음에도 상기 특정 파일의 message digest에 대응되는 정보(transaction ID)가 발견되지 않으면(S520-1), 통신부(110)는 검증 실패 메시지를 특정 사용자에게 전송할 수 있다(S525).
- [0085] 또한, 상기 특정 파일의 message digest에 대응되는 정보(transaction ID)가 발견되면(S520-2) 통신부(110)는 상기 transaction ID(TxID)를 사용하여 데이터베이스(200)에 질의하여(S525) 상기 데이터베이스(200)로부터 상기 특정 사용자의 private key(PrivU) 및 서버(100)의 private key(PrivC)로 인코딩된 message digest의 해쉬값을 포함하는 OP 메시지를 획득할 수 있다(S530).
- [0086] 그리고, 프로세서(120)는 상기 특정 파일의 message digest를 서버(100)의 private key 및 특정 사용자의 private key로 인코딩한 값의 해쉬값과 상기 OP 메시지에 포함된 상기 특정 사용자의 private key 및 상기 서버(100)의 private key로 인코딩된 message digest의 해쉬값이 동일하면, 상기 특정 파일이 검증된 것으로 판단하거나 판단하도록 지원할 수 있다(S535).
- [0087] 이때, 상기 특정 파일의 message digest를 서버(100)의 private key 및 특정 사용자의 private key로 인코딩한 값은 상기 특정 파일의 message digest를 상기 특정 사용자의 private key 및 상기 서버(100)의 private key로 순차로 인코딩한 값(Hash[EncPrivC(EncPrivU(MD))])일 수 있다.
- [0088] 다만, 프로세서(120)는, 상기 특정 파일의 message digest를 상기 서버(100)의 private key 및 상기 특정 사용자의 private key로 인코딩한 값(X)의 해쉬값과 상기 OP 메시지에 포함된 상기 특정 사용자의 private key 및 상기 서버(100)의 private key로 인코딩된 message digest의 해쉬값이 동일한 경우라도, 상기 X를 서버(100)의 public key 및 상기 특정 사용자의 public key를 사용하여 디코딩한 결과값(DecPubU(DecPubC(EncPrivC(EncPrivU(MD))))이 상기 특정 파일의 message digest(MD)와 동일한 경우에 상기 특정 파일이 검증된 것으로 판단하거나 판단하도록 추가적으로 지원할 수도 있다(S540). 즉, 두 단계의 검증을 더 수행하여 파일의 검증을 완료할 수도 있다. 이는 사용자의 private key, public key 등이 변동되었을 가능성을 감안하기 위함이다.

- [0089] 상기 세 단계의 검증은 아래의 표현식으로 나타낼 수 있다.
- [0091] <표현식 1>
- [0092] $OP_MESSAGE == Hash[Enc_{PrivC}(Enc_{PrivU}(MD))]$
- [0093] $Enc_{PrivU}(MD) == Dec_{PubC}[Enc_{PrivC}(Enc_{PrivU}(MD))]$
- [0094] $MD (=Hash[File]) == Dec_{PubU}[Enc_{PrivU}(MD)]$
- [0096] 파일의 검증이 성공한 경우, 통신부(110)는 검증 성공 메시지를 특정 사용자에게 전달할 수 있다(S545).
- [0097] 또한, 프로세서(120)는 상기 특정 파일이 검증된 경우, 상기 특정 파일의 다운로드를 지원할 수 있다. 이 경우 특정 사용자는 사용자 단말 등을 통해 특정 파일을 다운로드 받을 수 있다.
- [0098] 다음으로, 서버(100)가 관리하는 별도의 데이터 구조를 이용하여 검증을 수행하는 경우를 설명한다.
- [0099] 우선, 상술한 실시예와 같이 기존에 서버(100)가 특정 파일에 대한 노터리 서비스 요청을 받았을 때, 제1 특정 해쉬값(설명의 편의를 위하여 전술한 특정 해쉬값을 여기서는 제1 특정 해쉬값이라 지칭한다.) - 상기 제1 특정 해쉬값은, 상기 특정 파일의 message digest에 대하여 특정 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 소정의 조건 하에서 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써 생성되는 제1 대표 해쉬값(설명의 편의를 위하여 전술한 대표 해쉬값을 여기서는 제1 대표 해쉬값이라 지칭한다.) 또는 상기 제1 대표 해쉬값을 가공한 값을 데이터베이스(200)에 등록하고 있는 상태에서, 상기 특정 파일과 관련된 검증 요청이 획득되고, 상기 검증 요청에 포함된 입력 데이터를 사용하여 생성된 제2 대표 해쉬값 또는 상기 제2 대표 해쉬값을 가공한 값이 상기 데이터베이스(200)에 등록된 상기 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값과 대응되면, 프로세서(120)는 상기 특정 파일과 관련된 검증이 이루어진 것으로 판단하거나 판단하도록 지원할 수 있다.
- [0100] 여기서, 상기 제2 대표 해쉬값은, 제2 특정 해쉬값 - 상기 제2 특정 해쉬값은, 상기 검증 요청에 포함된 입력 데이터에 관한 상기 특정 파일의 message digest에 대하여 특정 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써 생성될 수 있다.
- [0101] 이때, 상기 검증 요청에 포함된 입력 데이터는, (i) 상기 특정 파일, (ii) 상기 특정 파일에 대한 message digest, 또는 (iii) 상기 특정 파일에 대한 노터리 서비스 당시 발급된 ID 중 어느 하나를 포함할 수 있다.
- [0102] 입력 데이터에 특정 파일이 포함되는 경우, 전술한 방법으로 특정 파일에 대한 message digest가 생성되고 이를 기초로 하여 상기 제2 특정 해쉬값이 산출될 수 있다. 그리고, 상기 제2 대표 해쉬값은 상기 제2 특정 해쉬값과 매칭되는 적어도 하나의 이웃 해쉬값을 연산하여 생성될 수 있다.
- [0103] 입력 데이터에 노터리 서비스 당시 발급된 ID가 포함되는 경우 기존에 생성된 머클 트리에서 상기 ID에 대응되는 리프 노드에 할당된 값이 상기 제2 특정 해쉬값으로 설정될 수 있다. 도 3 및 4에 도시된 예에서, 상기 ID는 unique ID일 수 있다.
- [0104] 제2 특정 해쉬값에 대해서는 전술한 제1 특정 해쉬값에 대한 설명이 동일하게 적용될 수 있다. 즉, 제2 특정 해쉬값과 적어도 하나의 이웃 해쉬값의 연산은 다양한 함수에 의해 수행될 수 있다. 이에 대해서는 전술하였으므로 설명을 생략한다.
- [0105] 이 경우, 상기 제2 특정 해쉬값과 적어도 하나의 이웃 해쉬값의 연산은 머클 트리(merkle tree)를 통해 이루어질 수 있다. 상기 특정 파일과 관련된 검증 요청이 획득되면, 프로세서(120)는, 상기 입력 데이터와 관련된 머클 트리 정보 및 리프 노드 정보를 식별할 수 있다.
- [0106] 그리고, 상기 제2 대표 해쉬값은, 상기 제2 특정 해쉬값이 특정 리프 노드에 할당된 머클 트리(merkle tree)에서 상기 제2 특정 해쉬값 및 상기 제2 특정 해쉬값과 매칭되는 적어도 하나의 다른 리프 노드에 할당된 해쉬값을 연산하여 생성될 수 있다.

- [0107] 서버(100)가 관리하지 않는 데이터베이스(예를 들어, 가상 화폐의 블록체인)를 사용하여 서비스가 이루어지는 경우, 프로세서(120)는, 상기 식별된 머클 트리 정보에 대응되는 상기 소정의 transaction ID를 참조할 수 있다.
- [0108] 이 경우, 프로세서(120)는, 상기 특정 파일과 관련된 검증 요청이 획득되면, 상기 특정 파일과 관련된 소정의 transaction ID를 참조하고, 상기 transaction ID를 사용하여 상기 데이터베이스(200)로부터 OP 메시지를 획득하거나 획득하도록 지원할 수 있다. 또한, 상기 검증 요청에 포함된 입력 데이터를 사용하여 생성된 상기 제2 대표 해쉬값 또는 상기 제2 대표 해쉬값을 가공한 값이 상기 OP 메시지에 포함된 상기 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값과 대응되면, 상기 특정 파일과 관련된 검증이 이루어진 것으로 판단하거나 판단하도록 지원할 수 있다.
- [0109] 한편, 상기 소정의 transaction ID는 입력 데이터에 포함되어 제공될 수도 있다. 즉, 사용자는 검증요청시 전술한 (i) 상기 특정 파일, (ii) 상기 특정 파일에 대한 message digest, 또는 (iii) 상기 특정 파일에 대한 노터리 서비스 당시 발급된 ID 중 어느 하나와 함께 transaction ID를 서버(100)에 전달할 수도 있다.
- [0110] 검증 연산과정에서 프로세서(120)는 새로운 머클 트리를 생성하거나 생성하도록 지원할 수 있는데, 여기에는 전술한 노터리 서비스 수행 과정에서 생성된 머클 트리가 이용될 수 있다. 즉, 프로세서(120)는 (x1) (i) 상기 제2 특정 해쉬값과 (ii) 기존에 생성된 머클 트리에서 상기 제2 특정 해쉬값이 할당된 노드의 형제 노드에 할당된 해쉬값을 연산하거나 연산하도록 지원하고, 상기 연산값에 대한 해쉬값을 새로운 머클 트리에서 상기 노드의 부모 노드에 할당하거나 할당하도록 지원할 수 있다. 또한, (x2) 상기 부모 노드가 상기 새로운 머클 트리의 루트 노드이면, 상기 부모 노드에 할당된 해쉬값을 상기 제2 대표 해쉬값으로서 상기 OP 메시지에 포함된 값과 비교하거나 비교하도록 지원하고, (x3) 상기 부모 노드가 상기 새로운 머클 트리의 루트 노드가 아니면, 상기 부모 노드에 할당된 해쉬값을 상기 제2 특정 해쉬값으로 하여 상기 (x1) 내지 (x3)를 반복하여 수행할 수 있다.
- [0111] 프로세서(120)는, 최종적으로 생성된 머클 트리의 루트값 또는 이를 가공한 값이 상기 OP 메시지에 포함된 상기 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값과 대응되면, 상기 특정 파일과 관련된 검증이 이루어진 것으로 판단하거나 판단하도록 지원할 수 있다.
- [0112] 이처럼 본 발명은 노터리 서비스 단계에서 $n^m (=N)$ 개의 리프 노드를 가지는 머클 트리가 생성이 되면, 머클 트리의 높이($\log_n N=m$)만큼의 연산만을 수행함으로써 파일에 대한 검증을 수행할 수 있어 빠르게 transaction의 integrity 확인이 가능하게 된다.
- [0113] 한편, 프로세서(120)는 입력 데이터를 획득하는 경우, 입력 데이터와 관련된 시간 정보를 참조로 하여 상기 입력 데이터와 관련된 머클 트리 정보 및 리프 노드 정보 중 적어도 하나를 식별할 수도 있다.
- [0114] 이하에서는 도 3에 도시된 구체적인 예를 들어 검증 과정을 설명한다.
- [0115] 다시 도 3을 참조하면, input 2에 대한 검증 요청이 있는 경우, 프로세서(120)는 입력 데이터 정보를 참조로 하여 기존에 생성된 머클 트리과 리프 노드 정보를 식별하고, 식별된 머클 트리 정보(tree_id = 0)에 대응되는 소정의 transaction ID를 데이터베이스(200)에 전송하여 데이터베이스(200)로부터 OP 메시지를 획득한다. 그리고, 프로세서(120)는 input 2를 이용하여 새로운 머클 트리를 생성하거나 생성하도록 지원할 수 있다. input 2의 해쉬값(triplesha256digest(input2))이 h2 노드에 할당되어 있으므로 프로세서(120)는 h2 노드에 할당된 해쉬값과 기존에 생성된 머클 트리에서 h2 노드의 형제 노드인 h3 노드에 할당된 해쉬값을 연산하거나 연산하도록 지원할 수 있다. 상기 연산값에 대한 해쉬값은 h2 노드와 h3 노드의 부모 노드인 h23 노드에 할당된다. h23 노드가 새로운 머클 트리의 루트 노드가 아니므로 프로세서(120)는 상기 h23 노드에 할당된 해쉬값을 상기 제2 특정 해쉬값으로 하여 상기 과정을 반복하여 수행할 수 있다. 즉, h23 노드에 할당된 해쉬값을 제2 특정 해쉬값으로 하고, h23 노드에 할당된 해쉬값과 기존에 생성된 머클 트리에서 h01 노드에 할당된 해쉬값을 연산하여 h23 노드와 h01 노드의 부모 노드인 h0123 노드에 할당할 수 있다. 이때, h0123 노드가 머클 트리의 루트 노드이므로 프로세서(120)는, h0123 노드에 할당된 해쉬값을 가공한 값(hex(h{node_index}))을 상기 OP 메시지에 포함된 값과 비교하거나 비교하도록 지원할 수 있다.
- [0116] 다음으로, 다중 사용자의 요청에 따라 파일에 대해 노터리 서비스를 수행하는 과정을 설명한다. 다만, 중복되는 설명은 생략한다. 우선, 서버(100)에 의해 관리되는 별도의 데이터 구조를 이용하지 않는 공증 시스템을 설명한다.
- [0117] 도 6은 다중 사용자의 노터리 서비스 요청에 대해 병렬적으로 서비스를 수행하는 parallel signing 과정을 나타

낸 것이고 도 7은 순차적으로 서비스를 수행하는 serial signing 과정을 나타내고 있다.

- [0118] 먼저, 도 6을 참조하면 프로세서(120)는, 특정 파일에 대한 다중 사용자의 노터리 서비스 요청이 획득되면 (S610, S620), hash 함수를 사용하여 상기 특정 파일의 message digest(MD=Hash(File))를 생성하거나 생성하도록 지원할 수 있다(S612, S622). 그리고, 제1 사용자의 private key(PrivU1)로 인코딩된 message digest 및 제2 사용자의 private key(PrivU2)로 인코딩된 message digest가 획득되고(S640, S650), (i) 상기 제1 사용자의 private key로 인코딩된 message digest를 상기 제1 사용자의 public key(PubU1)로 디코딩한 정보(A', DecPubU1(EncPrivU1(MD))), (ii) 상기 제2 사용자의 private key로 인코딩된 message digest를 상기 제2 사용자의 public key(PubU2)로 디코딩한 정보(B', DecPubU2(EncPrivU2(MD))) 및 상기 생성된 message digest(C', MD)가 일치하면(S640-3), 상기 제1 사용자의 private key, 상기 제2 사용자의 private key 및 상기 서버(100)의 private key를 사용하여 인코딩된 message digest의 해쉬값을 데이터베이스(200)에 등록하거나 등록하도록 지원할 수 있다(S675).
- [0119] 그리고, 통신부(110)는 상기 등록된 해쉬값의 상기 데이터베이스(200) 상의 위치 정보를 나타내는 transaction ID를 획득할 수 있다(S680).
- [0120] 또한, 상기 A', 상기 B' 및 상기 C'가 일치하면, 프로세서(120)는, 상기 제1 사용자의 private key로 인코딩된 message digest와 상기 제2 사용자의 private key로 인코딩된 message digest를 합한 값을 상기 서버(100)의 private key로 인코딩한 값(EncPrivC(EncPrivU1(MD)+EncPrivU2(MD)))을 데이터베이스(200)에 등록하거나 등록하도록 지원할 수 있다.
- [0121] 다음으로 도 7을 참조하면, 프로세서(120)는, 상기 특정 파일에 대한 노터리 서비스 요청이 획득되면(S710, S735), hash 함수를 사용하여 상기 특정 파일의 message digest(MD=Hash(File))를 생성하거나 생성하도록 지원할 수 있다(S712, S737). 이 경우, 도 7에 도시된 것처럼 제1 사용자의 public key(PubU1), 제2 사용자의 public key(PubU2), 특정 파일(File)이 함께 획득될 수 있다.
- [0122] 프로세서(120)는 (i) 제1 사용자의 private key(PrivU1)로 인코딩된 message digest(EncPrivU1(MD))가 획득되고(S720), 상기 제1 사용자의 private key로 인코딩된 message digest를 상기 제1 사용자의 public key로 디코딩한 정보(A', DecPubU1(EncPrivU1(MD)))와 상기 생성된 message digest(B', MD)가 일치하고, (ii) 상기 제1 사용자의 private key 및 제2 사용자의 private key(PrivU2)로 인코딩된 message digest(EncPrivU2(EncPrivU1(MD)))가 획득되고(S745), 상기 제1 사용자의 private key 및 상기 제2 사용자의 private key로 인코딩된 message digest를 상기 제2 사용자의 public key로 디코딩한 정보(C', DecPubU2(EncPrivU2(EncPrivU1(MD))))와 상기 생성된 message digest를 상기 제1 사용자의 private key로 인코딩한 정보(D', EncPrivU1(MD))가 일치하면(S760), 상기 제1 사용자의 private key, 상기 제2 사용자의 private key 및 상기 서버(100)의 private key를 사용하여 인코딩된 message digest의 해쉬값을 데이터베이스(200)에 등록하거나 등록하도록 지원한다(S775).
- [0123] 또한, 상기 A'와 상기 B'가 일치하고, 상기 C'와 상기 D'가 일치하면, 프로세서(120)는, 상기 제1 사용자의 private key, 상기 제2 사용자의 private key 및 서버(100)의 private key로 순차로 인코딩된 message digest(S0=EncPrivC(EncPrivU2(EncPrivU1(MD))))를 데이터베이스(200)에 등록하거나 등록하도록 지원할 수 있다.
- [0124] 이때, 통신부(110)는, 상기 등록된 해쉬값의 상기 데이터베이스(200) 상의 위치 정보를 나타내는 transaction ID(TxID)를 획득할 수 있다(S780)
- [0125] 다음으로, 서버(100)에 의해 관리되는 별도의 데이터 구조를 이용하는 공증 시스템에서 다중 사용자의 노터리 서비스 요청을 처리하는 실시예를 설명한다.
- [0126] 프로세서(120)는 (i) 특정 파일에 대한 노터리 서비스 요청이 획득되면, hash 함수를 사용하여 상기 특정 파일의 message digest를 생성하거나 생성하도록 지원하고, (ii) 소정의 조건이 만족되면, 특정 해쉬값 - 상기 특정 해쉬값은, 상기 message digest에 대하여 제1 사용자의 private key, 제2 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써 생성되는 대표 해쉬값 또는 상기 대표 해쉬값을 가공한 값을 데이터베이스(200)에 등록하거나 등록하도록 지원할 수 있다.
- [0127] 이때, 상기 특정 해쉬값은 상기 message digest를 상기 제1 사용자의 private key로 인코딩한 값과 상기 message digest를 상기 제2 사용자의 private key로 인코딩한 값을 합한 값을 서버(100)의 private key로 인코

딩한 값($\text{EncPrivC}(\text{EncPrivU1}(\text{MD})+\text{EncPrivU2}(\text{MD}))$)에 대한 해쉬값일 수 있다(parallel signing 과정).

- [0128] 이와 달리, 상기 특정 해쉬값은 상기 message digest를 상기 제1 사용자의 private key, 상기 제2 사용자의 private key 및 상기 서버의 private key로 순차로 인코딩한 값($\text{EncPrivC}(\text{EncPrivU2}(\text{EncPrivU1}(\text{MD})))$)에 대한 해쉬값일 수 있다(serial signing 과정).
- [0129] 다음으로, 다중 사용자의 요청에 따라 공증된 파일에 대해 검증을 수행하는 과정을 설명한다. 우선, 서버(100)에 의해 관리되는 별도의 데이터 구조를 이용하지 않는 경우를 도 8을 참조로 하여 설명한다. 다만, 전술한 내용과 중복되는 설명은 생략한다.
- [0130] 도 8은 다중 사용자의 서명으로 공증이 이루어진 경우 공증된 파일에 대해 검증을 수행하는 과정을 도시한 시퀀스 다이어그램이다.
- [0131] 참고로, S825~S850 단계는 사용자의 검증 요청에 대해 순차적으로 검증을 수행하는 serial signing 과정을 나타낸 것이고 S860~S885는 병렬적으로 검증을 수행하는 parallel signing 과정을 나타내고 있다. 각 과정은 별도로 이루어질 수 있고, 함께 수행될 수도 있다.
- [0132] 도 8을 참조하면, 프로세서(120)는, 전술한 바와 같이 서버(100)가 특정 파일(File)에 대한 노터리 서비스 요청을 받았을 때, 상기 특정 파일과 관련된 사람이 제1 사용자 및 제2 사용자임이 감지되어, 상기 제1 사용자의 private key(PrivU1), 상기 제2 사용자의 private key(PrivU2) 및 서버(100)의 private key(PrivC)를 사용하여 인코딩된 message digest의 해쉬값을 데이터베이스(200)에 저장하고 transaction ID(TxID)를 받아서 관리하고 있는 상태에서, 상기 특정 파일에 대한 검증 요청이 획득되면(S810), 상기 특정 파일의 message digest($\text{MD}=\text{Hash}(\text{File})$) 및 검증 요청을 한 특정 사용자의 public key 중 적어도 하나에 대응되는 transaction ID를 참조할 수 있다.
- [0133] 또한, 통신부(110)는, transaction ID를 사용하여 데이터베이스(200)로부터 상기 제1 사용자의 private key, 상기 제2 사용자의 private key 및 상기 서버(100)의 private key를 사용하여 인코딩된 message digest의 해쉬값을 포함하는 OP 메시지를 획득할 수 있다(S835, S870).
- [0134] 또한, 프로세서(120)는, 상기 특정 파일의 message digest를 서버(100)의 private key, 상기 제1 사용자의 private key 및 상기 제2 사용자의 private key를 사용하여 인코딩한 값의 해쉬값과 상기 OP 메시지에 포함된 상기 제1 사용자의 private key, 상기 제2 사용자의 private key 및 상기 서버(100)의 private key를 사용하여 인코딩된 message digest의 해쉬값이 동일하면, 상기 특정 파일이 검증된 것으로 판단하거나 판단하도록 지원할 수 있다(S840, S875).
- [0135] parallel signing 과정의 경우, 프로세서(120)는, 상기 제1 사용자의 private key로 인코딩된 message digest와 상기 제2 사용자의 private key로 인코딩된 message digest를 합한 값을 상기 서버(100)의 private key로 인코딩한 값의 해쉬값($\text{Hash}[\text{EncPrivC}(\text{EncPrivU1}(\text{MD})+\text{EncPrivU2}(\text{MD}))]$)과 상기 OP 메시지에 포함된 상기 제1 사용자의 private key, 상기 제2 사용자의 private key 및 상기 서버(100)의 private key를 사용하여 인코딩된 message digest의 해쉬값의 동일 여부에 따라 상기 특정 파일이 검증된 것으로 판단하거나 판단하도록 지원할 수 있다.
- [0136] 다만, 프로세서(120)는, 상기 제1 사용자의 private key로 인코딩된 message digest와 상기 제2 사용자의 private key로 인코딩된 message digest를 합한 값을 상기 서버(100)의 private key로 인코딩한 값(Y)의 해쉬값($\text{Hash}[\text{EncPrivC}(\text{EncPrivU1}(\text{MD})+\text{EncPrivU2}(\text{MD}))]$)과 상기 OP 메시지에 포함된 상기 제1 사용자의 private key, 상기 제2 사용자의 private key 및 상기 서버(100)의 private key를 사용하여 인코딩된 message digest의 해쉬값의 동일한 경우라도, (i) 상기 Y를 서버(100)의 public key로 디코딩하여 나온, message digest를 상기 제1 사용자의 private key로 인코딩한 값을, 상기 제1 사용자의 public key로 디코딩한 값이 상기 특정 파일의 message digest(MD)와 동일하고, (ii) 상기 Y를 서버(100)의 public key로 디코딩하여 나온, message digest를 상기 제2 사용자의 private key로 인코딩한 값을, 상기 제2 사용자의 private key로 디코딩한 값이 상기 특정 파일의 message digest(MD)와 동일한 경우에 상기 특정 파일이 검증된 것으로 판단하거나 판단하도록 지원할 수도 있다. 즉, 세 단계의 검증을 더 수행하여 파일의 검증을 완료할 수도 있다.
- [0137] 상기 네 단계의 검증은 아래의 표현식으로 나타낼 수 있다.

[0139] <표현식 2>

[0140] $OP_MESSAGE == Hash[EncPrivC(EncPrivU1(MD)+EncPrivU2(MD))]$

[0141] $EncPrivU1(MD)+EncPrivU2(MD) == DecPubC[EncPrivC(EncPrivU1(MD)+EncPrivU2(MD))]$

[0142] $MD (= Hash[File]) == DecPubU1[EncPrivU1(MD)]$

[0143] $MD (= Hash[File]) == DecPubU2[EncPrivU2(MD)]$

[0145] 반면, serial signing 과정의 경우, 프로세서(120)는, 상기 제1 사용자의 private key, 상기 제2 사용자의 private key 및 상기 서버(100)의 private key로 순차로 인코딩된 message digest의 해쉬값($Hash[EncPrivC(EncPrivU2(EncPrivU1(MD)))]$)과 상기 OP 메시지에 포함된 상기 제1 사용자의 private key, 상기 제2 사용자의 private key 및 서버(100)의 private key 를 사용하여 인코딩된 message digest의 해쉬값의 동일 여부에 따라 상기 특정 파일이 검증된 것으로 판단하거나 판단하도록 지원할 수 있다.

[0146] 다만, 프로세서(120)는, 상기 제1 사용자의 private key, 상기 제2 사용자의 private key 및 상기 서버(100)의 private key로 순차로 인코딩된 message digest(Z)의 해쉬값($Hash[EncPrivC(EncPrivU2(EncPrivU1(MD)))]$)과 상기 OP 메시지에 포함된 상기 제1 사용자의 private key, 상기 제2 사용자의 private key 및 상기 서버(100)의 private key 를 사용하여 인코딩된 message digest의 해쉬값의 동일한 경우라도, 상기 Z를 서버(100)의 public key, 상기 제2 사용자의 public key 및 상기 제1 사용자의 public key 를 사용하여 디코딩한 결과값($DecPubU1(DecPubU2(DecPubC(EncPrivC(EncPrivU2(EncPrivU1(MD)))))$)이 상기 특정 파일의 message digest(MD)와 동일한 경우에 상기 특정 파일이 검증된 것으로 판단하거나 판단하도록 지원할 수도 있다. 즉, 세 단계의 검증을 더 수행하여 파일의 검증을 완료할 수도 있다.

[0147] 상기 네 단계의 검증은 아래의 표현식으로 나타낼 수 있다.

[0149] <표현식 3>

[0150] $OP_MESSAGE == Hash[EncPrivC(EncPrivU2(EncPrivU1(MD)))]$

[0151] $EncPrivU2(EncPrivU1(MD)) == DecPubC[EncPrivC(EncPrivU2(EncPrivU1(MD)))]$

[0152] $EncPrivU1(MD) == DecPubU2[EncPrivU2(EncPrivU1(MD))]$

[0153] $MD (= Hash[File]) == DecPubU1[EncPrivU1(MD)]$

[0155] 다음으로, 서버(100)에 의해 관리되는 별도의 데이터 구조를 이용하는 공증 시스템에서 검증 요청을 처리하는 실시예를 설명한다.

[0156] 프로세서(120)는 기존에 서버(100)가 상기 특정 파일에 대한 공증 요청을 받았을 때, 제1 특정 해쉬값 - 상기 제1 특정 해쉬값은, 상기 특정 파일의 message digest에 대하여 제1 사용자의 private key, 제2 사용자의 private key 및 상기 서버의 private key로 인코딩한 결과에 대한 해쉬값임 - 과 소정의 조건 하에서 매칭되는 적어도 하나의 이웃 해쉬값을 연산함으로써 생성되는 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값을 데이터베이스에 등록하고 있는 상태에서, 상기 특정 파일과 관련된 검증 요청이 획득되고, 상기 검증 요청에 포함된 입력 데이터를 사용하여 생성된 제2 대표 해쉬값 또는 상기 제2 대표 해쉬값을 가공한 값이 상기 데이터베이스에 등록된 상기 제1 대표 해쉬값 또는 상기 제1 대표 해쉬값을 가공한 값과 대응되면, 상기 특정 파일과 관련된 검증이 이루어진 것으로 판단하거나 판단하도록 지원할 수 있다.

[0157] 이때, 상기 제1 특정 해쉬값은 상기 message digest를 상기 제1 사용자의 private key로 인코딩한 값과 상기 message digest를 상기 제2 사용자의 private key로 인코딩한 값을 합한 값을 상기 서버의 private key로 인코딩한 값($EncPrivC(EncPrivU1(MD)+EncPrivU2(MD))$)에 대한 해쉬값일 수 있다(parallel signaling 과정).

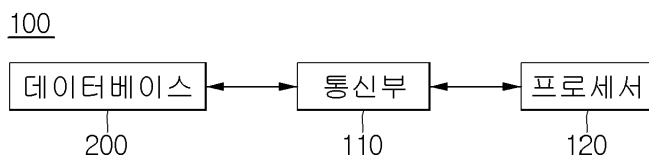
- [0158] 이와 달리, 상기 제1 특정 해쉬값은 상기 message digest을 상기 제1 사용자의 private key, 상기 제2 사용자의 private key 및 상기 서버의 private key로 순차로 인코딩한 값(EncPrivC(EncPrivU2(EncPrivU1(MD))))에 대한 해쉬값일 수 있다(serial signaling 과정).
- [0159] 한편, 프로세서(120)는 통신부(110) 및 다른 구성요소 간의 데이터의 흐름을 제어하는 기능을 수행한다. 즉, 프로세서(120)는 서버(100)의 각 구성요소 간의 데이터의 흐름을 제어함으로써, 통신부(110) 및 다른 구성요소에서 각각 고유 기능을 수행하도록 제어한다.
- [0160] 프로세서(120)는 MPU(Micro Processing Unit) 또는 CPU(Central Processing Unit), 캐쉬 메모리(Cache Memory), 데이터 버스(Data Bus) 등의 하드웨어 구성을 포함할 수 있다. 또한, 운영체제, 특정 목적을 수행하는 어플리케이션의 소프트웨어 구성을 더 포함할 수도 있다.
- [0161] 또한, 이상 설명된 본 발명에 따른 실시예들은 다양한 컴퓨터 구성요소를 통하여 수행될 수 있는 프로그램 명령어의 형태로 구현되어 컴퓨터 판독 가능한 기록 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능한 기록 매체는 프로그램 명령어, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 컴퓨터 판독 가능한 기록 매체에 기록되는 프로그램 명령어는 본 발명을 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 분야의 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능한 기록 매체의 예에는, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM, DVD와 같은 광기록 매체, 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 ROM, RAM, 플래시 메모리 등과 같은 프로그램 명령어를 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령어의 예에는, 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드도 포함된다. 상기 하드웨어 장치는 본 발명에 따른 처리를 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.
- [0162] 이상에서 본 발명이 구체적인 구성요소 등과 같은 특정 사항들과 한정된 실시예 및 도면에 의해 설명되었으나, 이는 본 발명의 보다 전반적인 이해를 돕기 위해서 제공된 것일 뿐, 본 발명이 상기 실시예들에 한정되는 것은 아니며, 본 발명이 속하는 기술분야에서 통상적인 지식을 가진 자라면 이러한 기재로부터 다양한 수정 및 변형을 꾀할 수 있다.
- [0163] 따라서, 본 발명의 사상은 상기 설명된 실시예에 국한되어 정해져서는 아니 되며, 후술하는 특허청구범위뿐만 아니라 이 특허청구범위와 균등하게 또는 등가적으로 변형된 모든 것들은 본 발명의 사상의 범주에 속한다고 할 것이다.

부호의 설명

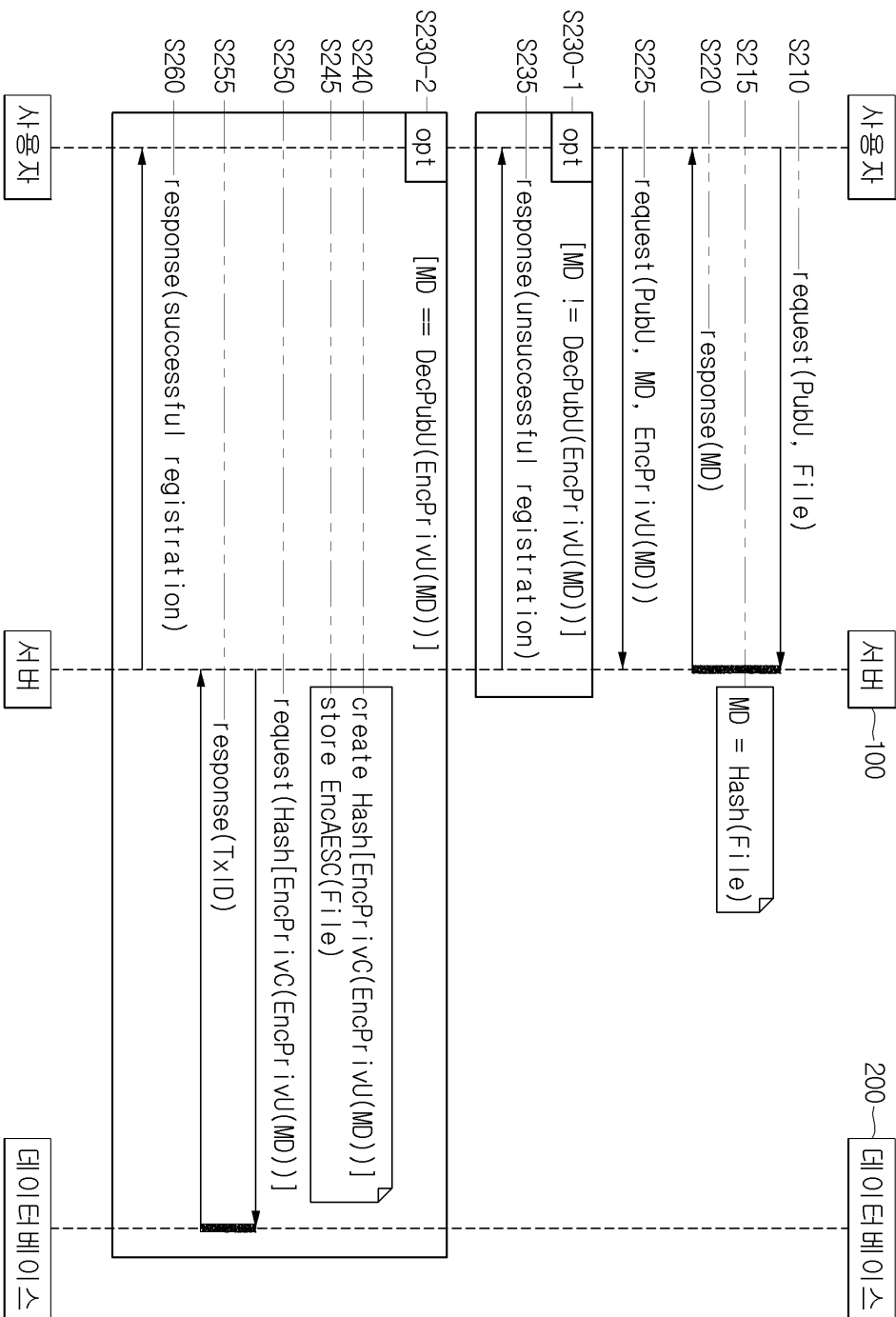
- [0164] 100 : 서버
110 : 통신부
120 : 프로세서
200 : 데이터베이스

도면

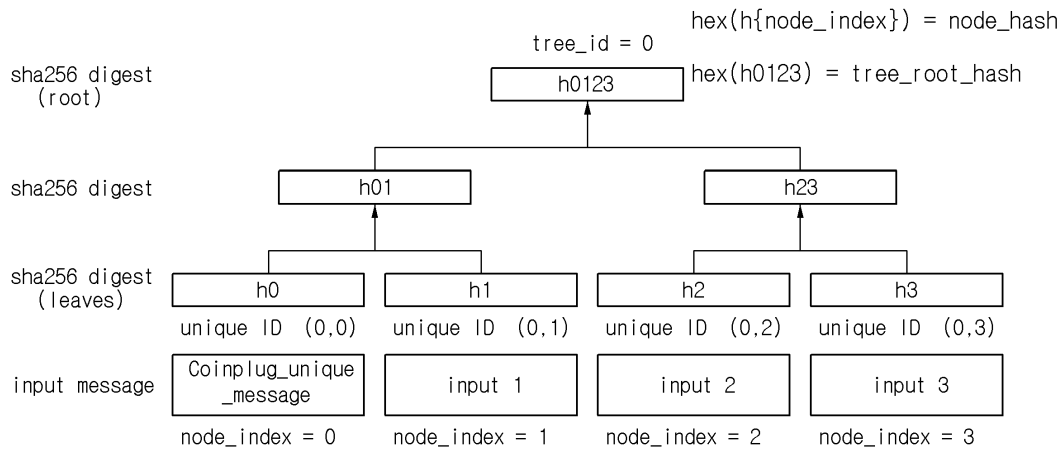
도면1



도면2



도면3



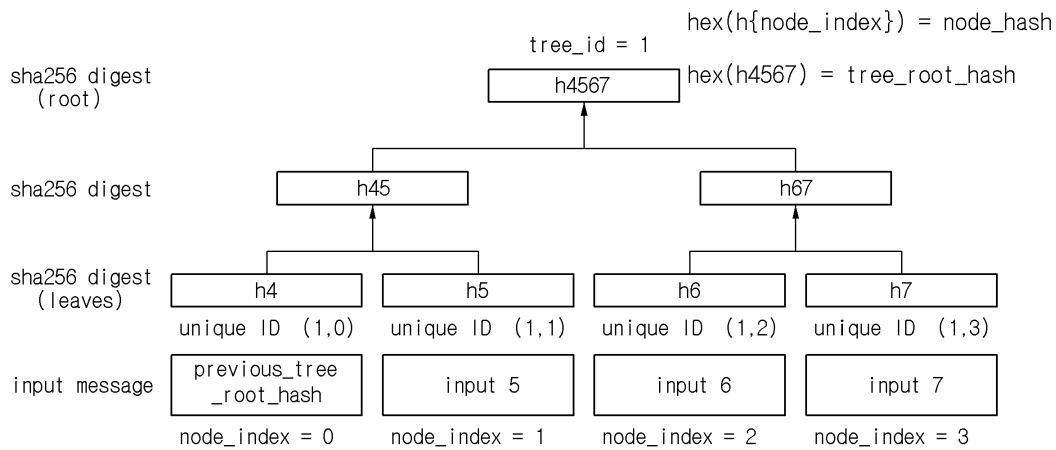
```
coinplug_unique_message = "Coinplug 2016-01-01"
input1 = hash_cp_signed_msg1
input2 = hash_cp_signed_msg2
input3 = hash_cp_signed_msg3
```

```
h0 = triplesha256digest(coinplug_unique_message)
h1 = triplesha256digest(input1)
h2 = triplesha256digest(input2)
h3 = triplesha256digest(input3)
```

```
h01 = sha256digest(h0 + h1)
h23 = sha256digest(h2 + h3)
```

```
h0123 = sha256digest(h01 + h23)
```

도면4



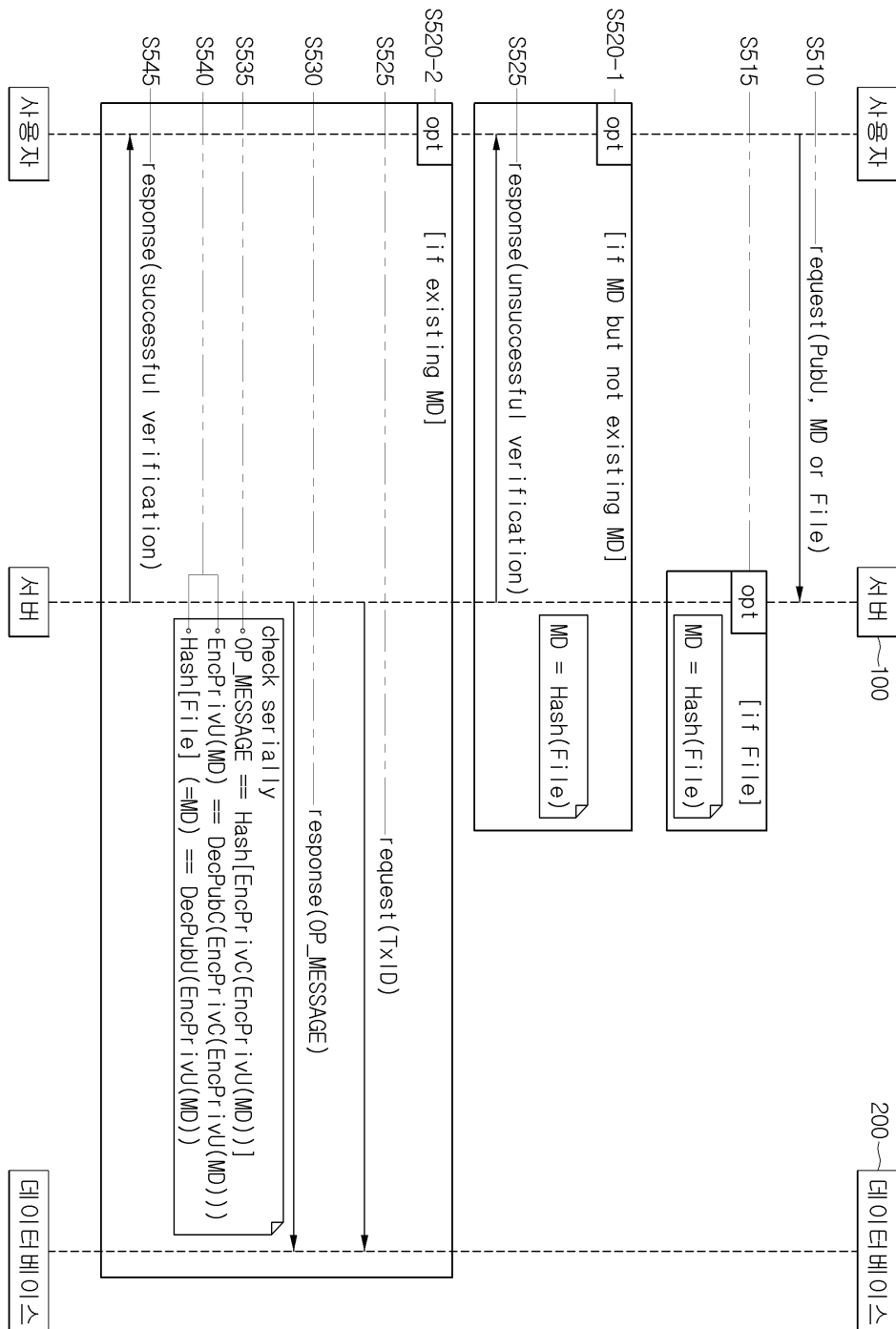
input4 = previous_tree_root_hash
 input5 = hash_cp_signed_msg5
 input6 = hash_cp_signed_msg6
 input7 = hash_cp_signed_msg7

h4 = triplsha256digest(input4)
 h5 = triplsha256digest(input5)
 h6 = triplsha256digest(input6)
 h7 = triplsha256digest(input7)

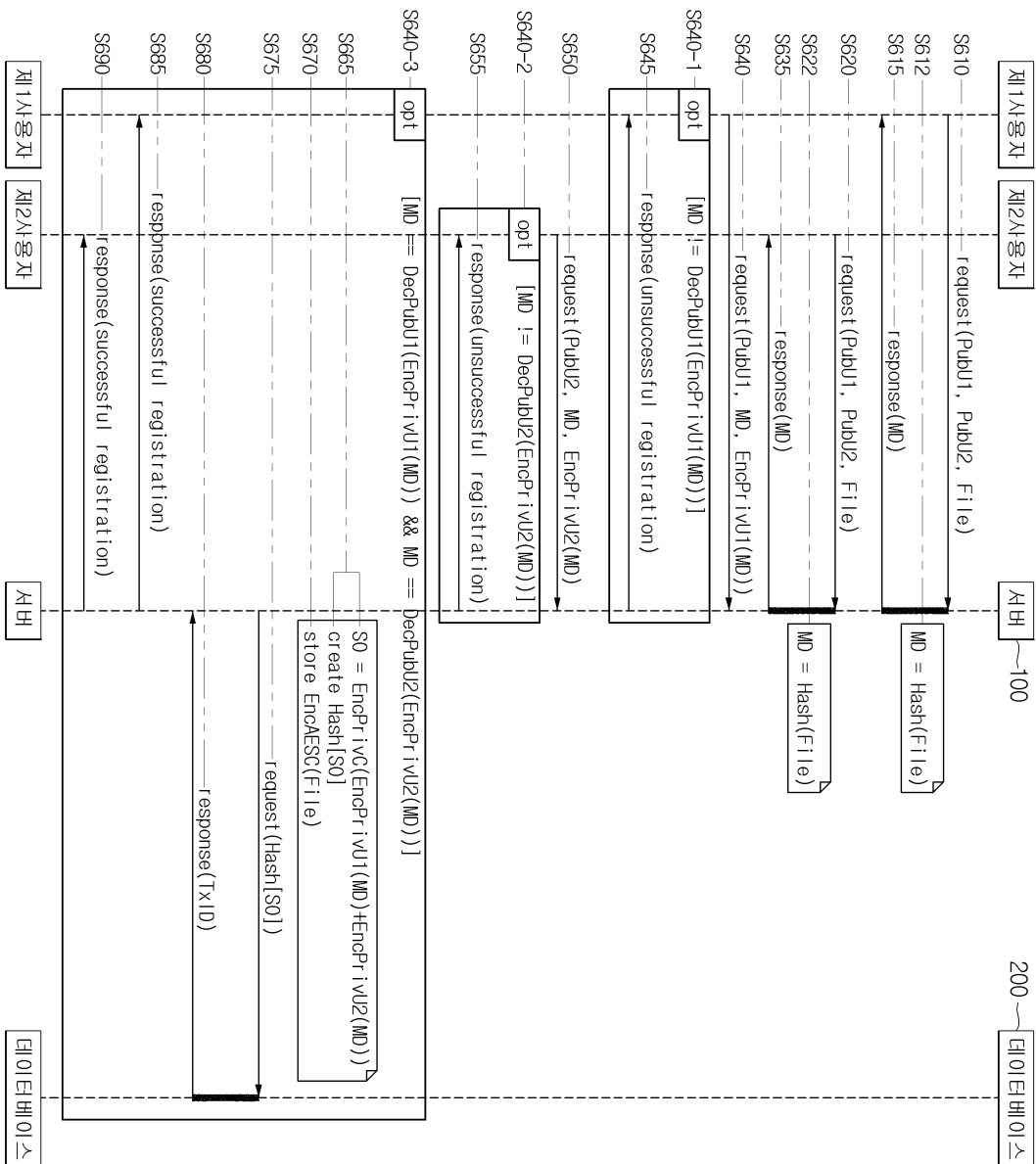
h45 = sha256digest(h4 + h5)
 h67 = sha256digest(h6 + h7)

h4567 = sha256digest(h45 + h67)

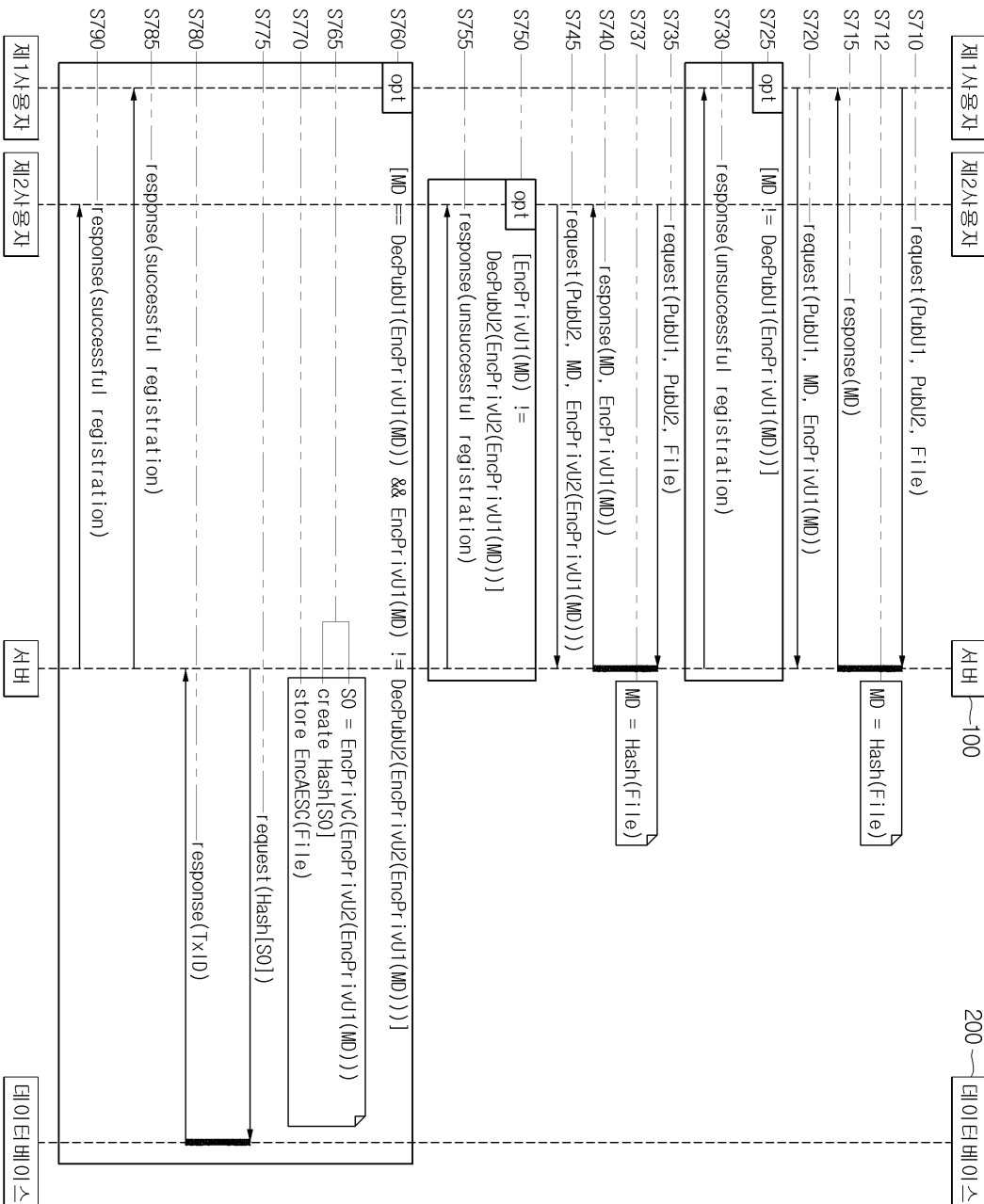
도면5



도면6



도면7



도면8

