(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0122170
(43) 공개일자 2022년09월02일

(51) 국제특허분류(Int. Cl.)
H04L 9/40 (2022.01) *G16Y 30/10* (2020.01)
H04L 65/40 (2022.01) *H04L 9/06* (2006.01)
H04L 9/08 (2006.01)
(52) CPC특허분류
H04L 63/0485 (2013.01)
G16Y 30/10 (2020.01)
(21) 출원번호 10-2021-0026452
(22) 출원일자 2021년02월26일
심사청구일자 2021년02월26일

(71) 출원인
인천대학교 산학협력단
인천광역시 연수구 아카데미로 119 (송도동)
(72) 발명자
전광길
인천광역시 연수구 해송로 70, 209동 805호(송도동, 송도웰카운티2단지)
(74) 대리인
특허법인충정

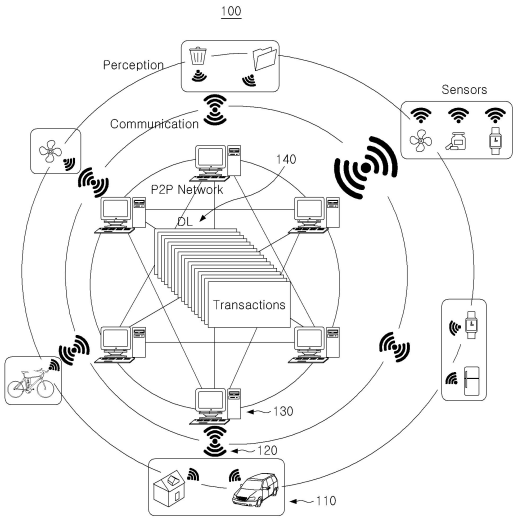
전체 청구항 수 : 총 9 항

(54) 발명의 명칭 **ＩｏＴ 네트워크 분산 보안 관리 시스템**

(57) 요 약

IoT 네트워크 분산 보안 관리 시스템은 IoT와 분산 원부(Distributed Ledger)의 두 가지 기술을 병합하기 위하여 트랙잭션 체인, 원부 정리 및 양자 보안 디지털 서명의 기술을 제안하여 IoT 노드와 P2P 네트워크 간의 안전한 데이터 전송을 보장할 수 있다.

대 표 도 - 도2



(52) CPC특허분류
H04L 63/061 (2013.01)
H04L 67/104 (2022.05)
H04L 67/12 (2022.05)
H04L 9/0643 (2013.01)
H04L 9/0825 (2013.01)
H04L 9/50 (2022.05)

명 세 서

청구범위

청구항 1

일반 가정 내의 사물인 센싱 디바이스로 구성되어 서로 통신하는 인식 계층;

리소스가 많은 컴퓨팅 노드인 피어(Peer)로 구성되고 각각의 피어가 서로 연결되어 P2P 네트워크를 형성한 P2P 네트워크 계층;

상기 인식 계층의 센싱 디바이스와 상기 피어 간의 통신을 제공하는 통신 계층; 및

상기 각각의 피어는 복수의 센서 및/또는 액추에이터에 연결되고, 상기 각각의 피어가 새로운 트랜잭션을 생성하고, 상기 P2P 네트워크를 통해 전송하고, 다른 피어가 트랜잭션(Transaction)을 확인하고, 성공적으로 확인되면 새로운 트랜잭션을 분산 장부(Distributed Ledger, DL)에 저장하도록 데이터 저장 서비스를 제공하는 DL 계층을 포함하는 IoT 네트워크 분산 보안 관리 시스템.

청구항 2

청구항 1에 있어서,

상기 분산 장부는 개별 트랜잭션을 블록으로 캡슐화하지 않고, 각 트랜잭션으로 저장하고, 상기 각 트랜잭션은 프루닝 키(Pruning Key)를 저장하는 IoT 네트워크 분산 보안 관리 시스템.

청구항 3

청구항 2에 있어서,

상기 각각의 피어는 상기 프루닝 키를 사용하여 트랜잭션 소유자인 다른 피어와의 합의 후, 더 이상 필요하지 않은 트랜잭션을 삭제하여 상기 분산 원부를 정기적으로 정리하는 IoT 네트워크 분산 보안 관리 시스템.

청구항 4

청구항 1에 있어서,

상기 각 피어는 센서에서 보낸 데이터를 저장하기 위해 상기 분산 원부를 유지하고, 원부 사본을 로컬에 저장하고, 센서가 데이터를 전송할 때 마다 새로운 트랜잭션을 생성하여 다른 피어로 전송하며, 상기 분산 원부의 보안 및 유효성을 키 생성(Keygen), 서명(Sign), 확인(Verify), 키압축(Keycompress)의 4개의 구성요소로 이루어진 DL-OTS(One-Time Signature) 방식으로 처리하는 IoT 네트워크 분산 보안 관리 시스템.

청구항 5

청구항 4에 있어서,

상기 키 생성은 보안 매개 변수(n)를 입력으로 받아들여 키 쌍(sk, pk)을 반환하는 것이고, 포스트 퀀텀 보안을 위해 하기의 수학식 1에 의해 해시 함수(SHA384)을 사용하여 시드(Seed)로 알려진 단일 초기값으로부터 모든 개인 키(sk) 값들을 생성하고, 개인 키(sk)를 공개 키(pk)로 변환하는 IoT 네트워크 분산 보안 관리 시스템.

[수학식 1]

$$\sum_{i=1}^{l} sk_{i-1} = H^{i}(seed)$$

$$\left[\sum_{i=0}^{l-2} pk_i = H^{w}(sk_i)\right] \cup \left[pk_{l-1} = H^{w_\circ}(sk_{l-1})\right]$$

여기서, ski는 개인 키의 개별값, pki는 공개 키의 개별값, H는 해시 함수 SHA384, Wc는 마지막 sk 값(체크섬에 사용됨)을 해당 pk 값으로 변환하는데 사용되는 해시 반복 횟수, W는 sk 값을 해당 pk 값으로 변환하는데 사용된 해시 반복 횟수, l 은 키/서명 값의 수임.

청구항 6

청구항 4에 있어서,

상기 키압축은 공개 키(pk)를 암호화 방식으로 압축하는 것으로 Merkle 해시 트리를 사용하여 상기 공개 키를 단일 384 비트 long 값으로 압축하는 IoT 네트워크 분산 보안 관리 시스템.

청구항 7

청구항 4에 있어서,

상기 각각의 피어에서의 분산 장부는 일련의 트랜잭션으로 구성되고, 상기 트랜잭션에는 다음 데이터 항목, 트랜잭션 시간, 센서 데이터(IoT 데이터), 프루닝 키(Pruning Key)(prnkey), 디지털 서명 및 후속 트랜잭션의 ID(succID)가 포함하는 IoT 네트워크 분산 보안 관리 시스템.

청구항 8

청구항 4에 있어서,

상기 각각의 피어는 상기 분산 장부에 트랜잭션이 저장되어 있고, 다른 피어의 합의 후에 새로운 트랜잭션을 추가하고, 각각의 새로운 트랜잭션은 이전 트랜잭션에서 ID를 가져오는 IoT 네트워크 분산 보안 관리 시스템.

청구항 9

청구항 4에 있어서,

상기 각각의 피어는 새로운 거래를 분산 원부에 저장하려면, 거래 소유자인 피어는 DL-OTS(One-Time Signature)를 사용하여 자신의 거래에 서명하고, 상기 서명은 분산 원부에 저장된 ID와 일치하는 IoT 네트워크 분산 보안 관리 시스템.

발명의 설명

기 술 분 야

[0001] 본 발명은 IoT 네트워크 분산 보안 관리 시스템에 관한 것으로서, 더욱 상세하게는 IoT와 분산 원부(Distributed Ledger)의 두 가지 기술을 병합하기 위하여 트랙잭션 체인, 원부 정리 및 양자 보안 디지털 서명의 기술을 제안하여 IoT 노드와 P2P 네트워크 간의 안전한 데이터 전송을 보장하는 IoT 네트워크 분산 보안 관리 시스템에 관한 것이다.

배 경 기 술

[0002] 분산 원부(Distributed Ledger)는 IoT(Industrial Internet of Things) 시스템의 보안 및 개인 정보 문제를 극복하는 데 도움이 될 수 있지만 이 두 기술을 병합하면 특정 문제가 발생할 수 있다.

[0003] Cryptocurrencies(비트코인)는 금융 거래의 역사를 유지하기 위해 분산 원부 기술을 시작했다. 암호 화폐의 DL 크기는 수백 GB에 접근하는 반면, IoT 노드는 스토리지에 제한이 있다. 마찬가지로 암호 화폐는 계산 비용이 많이 드는 합의 메커니즘을 통합하는 반면, IoT 노드는 계산과 에너지가 제한적이다.

[0004] IoT는 저장 공간, 계산 능력 및 에너지(배터리 수명)가 부족하고, 센서에서 생성된 데이터를 클라우드에 저장한다.

[0005] IoT 시스템에서 데이터의 보안, 개인 정보 보호 및 신뢰성과 관련된 문제가 발생할 수 있으며, 엄청난 양의 데이터를 클라우드로 전송하면, 네트워크 리소스에 과부하가 발생한다. 마찬가지로 악성 클라우드는 데이터의 보안 및 개인 정보를 손상시킬 수 있는 문제점이 있다.

선행기술문헌

특허문헌

[0006] (특허문헌 0001) 한국 등록특허번호 제10-1924026호

발명의 내용

해결하려는 과제

[0007] 이와 같은 문제점을 해결하기 위하여, 본 발명은 IoT와 분산 원부(Distributed Ledger)의 두 가지 기술을 병합하기 위하여 트랙잭션 체인, 원부 정리 및 양자 보안 디지털 서명의 기술을 제안하여 IoT 노드와 P2P 네트워크 간의 안전한 데이터 전송을 보장하는 IoT 네트워크 분산 보안 관리 시스템을 제공하는데 그 목적이 있다.

과제의 해결 수단

[0008] 상기 목적을 달성하기 위한 본 발명의 특징에 따른 IoT 네트워크 분산 보안 관리 시스템은,

[0009] 일반 가정 내의 사물인 센싱 디바이스로 구성되어 서로 통신하는 인식 계층;

[0010] 리소스가 많은 컴퓨팅 노드인 피어(Peer)로 구성되고 각각의 피어가 서로 연결되어 P2P 네트워크를 형성한 P2P 네트워크 계층;

[0011] 상기 인식 계층의 센싱 디바이스와 상기 피어 간의 통신을 제공하는 통신 계층; 및

[0012] 상기 각각의 피어는 복수의 센서 및/또는 액추에이터에 연결되고, 상기 각각의 피어가 새로운 트랜잭션을 생성하고, 상기 P2P 네트워크를 통해 전송하고, 다른 피어가 트랜잭션(Transaction)을 확인하고, 성공적으로 확인되면 새로운 트랜잭션을 분산 장부(Distributed Ledger, DL)에 저장하도록 데이터 저장 서비스를 제공하는 DL 계층을 포함한다.

[0014] 분산 장부는 개별 트랜잭션을 블록으로 캡슐화하지 않고, 각 트랜잭션으로 저장하고, 상기 각 트랜잭션은 프루닝 키(Pruning Key)를 저장한다.

[0016] 각각의 피어는 상기 프루닝 키를 사용하여 트랜잭션 소유자인 다른 피어와의 합의 후, 더 이상 필요하지 않은 트랜잭션을 삭제하여 상기 분산 원부를 정기적으로 정리할 수 있다.

발명의 효과

[0017] 전술한 구성에 의하여, 본 발명은 IoT와 분산 원부(Distributed Ledger)의 두 가지 기술을 병합하여 IoT 노드와 P2P 네트워크 간의 안전한 데이터 전송을 보장할 수 있는 효과가 있다.

도면의 간단한 설명

[0018] 도 1은 IoT 기반의 시스템의 레이아웃을 간략하게 나타낸 도면이다.

도 2는 본 발명의 실시예에 따른 IoT 네트워크 분산 보안 관리 시스템의 구성을 간략하게 나타낸 도면이다.

도 3은 본 발명의 실시예에 따른 DL-OTS 서명 생성을 설명하기 위한 도면이다.

도 4는 본 발명의 실시예에 따른 DL-OTS 키 압축을 나타낸 도면이다.

도 5는 본 발명의 실시예에 따른 트랜잭션 구조와 체인을 나타낸 도면이다.

도 6은 본 발명의 실시예에 따른 원부 정리 모습을 나타낸 도면이다.

도 7은 본 발명의 실시예에 따른 DL-for-IOT를 통합한 스마트 홈 시스템의 일례를 나타낸 도면이다.

발명을 실시하기 위한 구체적인 내용

[0019] 명세서 전체에서, 어떤 부분이 어떤 구성요소를 "포함"한다고 할 때, 이는 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라 다른 구성요소를 더 포함할 수 있는 것을 의미한다.

[0020] 사물 인터넷(IoT)은 내장된 기술을 사용하여 물리적 사물이 서로 연결하고 통신 할 수 있도록 한다. 많은 수의 물리적 개체가 서로 연결되어 있으므로 많은 양의 데이터가 생성된다.

[0021] 도 1은 IoT 기반의 시스템의 레이아웃을 간략하게 나타낸 도면이다.

[0022] IoT 시스템은 4 계층으로 아키텍처, 인식(감지) 계층(10), 네트워킹(통신) 계층(20), 서비스(클라우드 또는 블록 체인) 계층(30) 및 인터페이스(응용 프로그램) 계층(40)이다.

[0023] 감지 계층(10)에는 물리적 사물에 내장 된 장치(센서 및 액추에이터)가 포함되어 통신 할 수 있다. 센서에서 생성된 데이터는 네트워킹 계층(20)을 사용하여 클라우드로 전송된다. 클라우드는 데이터를 저장하고 애플리케이션이 해당 데이터를 처리할 수 있다.

[0024] 엄청난 양의 데이터를 클라우드로 전송하면, 네트워크 리소스에 과부하가 발생한다. 마찬가지로 악성 클라우드는 데이터의 보안 및 개인 정보를 손상시킬 수 있다. 마지막으로 클라우드 서버가 서비스를 중단하여 해당 IoT 시스템을 유휴 상태로 만들 수 있다.

[0025] 분산 원부(DL)는 IoT 시스템을 위한 적절한 대체 클라우드 서버를 제공한다. 중앙 서버에 데이터를 저장하는 대신 피어는 데이터를 직접 저장한다. 각 피어는 데이터베이스를 로컬로 유지하는 반면, 데이터 변경은 피어의 상호 합의에 의해 이루어진다. 거의 항상 각 피어의 데이터 사본이 동기화될 수 있다.

[0026] IoT 기반 시스템에 DL 기술을 채택하면 몇 가지 문제가 발생한다.

[0027] 암호 화폐에서 거래는 입출력 관계에 강하게 묶여 있다. 각각의 새로운 거래는 이전 거래에 의해 이전에 잠긴 코인을 사용한다. 그러나 IoT 환경에서는 거래 내용이 암호 화폐와 다르다. 예를 들어, 센서에서 생성된 데이터는 IoT 트랜잭션의 일례이다.

[0028] 따라서, IoT 트랜잭션을 정당한 체인으로 묶는 것은 IoT 환경에 통합된 DL의 과제이다.

[0029] 암호 화폐의 DL 크기는 일반적으로 매우 크다(예: 현재 비트 코인 원장의 크기가 200GB에 가까움). 반면 IoT 노드는 저장 공간이 제한된다. DL은 비트코인의 작업 증명(PoW)과 같은 계산 비용이 많이 드는 합의 메커니즘을 통합하여 원부에 새로운 트랜잭션, 블록을 추가할 수 있도록 한다. 에너지와 계산 능력이 제한되어있는 IoT 장치는 PoW와 같은 합의 메커니즘에 적합하지 않다.

[0031] 기존 DL(Bitcoin, Ethereum 등)에서 사용하는 디지털 서명 체계는 양자 안전이 아닌 ECDSA(Elliptic Curve Digital Signature Algorithm)이다.

[0032] DL 기술은 IoT에 혁명을 일으킬 수 있지만 두 기술의 통합과 관련된 과제가 있다. 두 기술을 병합하기 전에 관련 문제를 신중하게 고려해야 한다. 이러한 문제는 계산, 저장, 통신 및 에너지가 부족한 IoT 장치 때문이다.

[0033] IoT 장치는 작업 증명과 유사한 합의 메커니즘의 부하를 전달할 수 없다. 또한, IoT 장치는 계속 증가하는 원부 크기를 지원할 수 없다. 마지막으로, 기존 DL에서 사용하는 디지털 서명 체계는 비양자 복원력이 있다. 우리는 IoT 기반 시스템을 위한 양자 보안 DL, 즉 DL-for-IoT를 제안한다.

[0034] 본 발명의 DL은 상당히 효율적인 양자 보안 서명 체계인 새로운 DL-OTS를 설계한다.

[0035] 본 발명의 DL-OTS는 기존의 WOTS에 비해 키 생성, 서명 확인, 키 압축이 79%, 76%, 55% 감소된다.

[0036] 본 발명의 DL-OTS는 널리 사용되는 WOTS에 비해 48.7%의 에너지를 절약한다.

[0037] 본 발명은 IoT 환경에서 통합된 DL에 대한 트랜잭션 검증 규칙을 공식화한다.

[0038] 본 발명은 계속 증가하는 원부 크기를 피하기 위해 원부 정리 메카니즘이 제공되고, 계산 비용이 많이 드는 합의 알고리즘(예: 작업 증명)을 제거하기 위해 내장된 합의 메커니즘이 고안되었다.

[0040] 도 2는 본 발명의 실시예에 따른 IoT 네트워크 분산 보안 관리 시스템의 구성을 간략하게 나타낸 도면이다.

[0041] 본 발명의 실시예에 따른 IoT 네트워크 분산 보안 관리 시스템(100)은 4개의 계층으로, 인식 계층(110), 통신 계층(120), P2P 네트워크 계층(130) 및 DL 계층(140)으로 구성된다.

[0042] 인식 계층(110)은 일반 가정 사물(예: 알람 시계, 냉장고, 자전거 등)에 연결된 센싱 디바이스 또는 작동 장치로 구성되어 서로 통신할 수 있다.

[0043] 인식 계층(110)의 장치는 계산 및 저장 기능이 제한적이다.

[0044] P2P 네트워크 계층(130)은 리소스가 많은 컴퓨팅 노드(예: PC 또는 랩톱)인 피어(Peer)로 구성된다.

[0045] 피어는 서로 연결되어 P2P 네트워크를 형성한다. 각 피어는 분산 원부(DL)를 유지하기에 충분한 컴퓨팅 및 스토리지 기능을 보유한다.

[0046] 통신 계층(120)은 인식 계층(110)의 센싱 디바이스와 피어 간의 통신을 제공한다. 마지막으로 DL 계층(140)은 데이터 저장 서비스를 제공한다.

[0047] 본 발명의 DL 기반 IoT 네트워크 분산 보안 관리 시스템(100)에서 피어는 센서 및/또는 액추에이터를 대신하여 데이터를 유지한다.

[0048] 각 피어는 복수의 센서 및/또는 액추에이터에 연결된다. 해당 센서 중 하나에서 데이터를 수신하면 피어는 새로운 트랜잭션을 생성하고 P2P 네트워크를 통해 전송한다.

[0049] 모든 다른 피어는 트랜잭션을 확인하고 성공적으로 확인되면 새로운 트랜잭션을 분산 원부(DL)에 저장한다. 신뢰할 수 있는 피어만 DL에 새로운 트랜잭션을 저장할 수 있도록 하는 메커니즘을 제공한다.

[0050] DL 계층(140)은 각각의 피어가 새로운 트랜잭션을 생성하고, P2P 네트워크를 통해 전송하고, 다른 피어가 트랜잭션을 확인하고, 성공적으로 확인되면 새로운 트랜잭션을 분산 장부(Distributed Ledger, DL)에 저장하도록 데이터 저장 서비스를 제공한다.

[0051] IoT 환경에 통합된 DL에 대한 새로운 트랜잭션 확인 규칙을 고안했다. 본 바명에서 제안한 트랜잭션 확인 규칙은 DL이 트랜잭션을 입력/출력 관계로 바인딩 할 수 있도록 한다.

[0052] 따라서, 각 트랜잭션에는 정확히 하나의 선행 작업과 "zero or one"의 후속 작업이 있다. 트랜잭션 y가 이미 존재하는 트랜잭션 x의 후속 작업이 되려면 y의 서명이 이전에 x에 저장된 키를 확인해야 한다.

[0053] 따라서, 거래 y(ownerY)의 소유자는 거래 x(ownerX)의 소유자와 합의 후에 만 거래에 서명할 수 있다. 이러한 방식으로 owner-X는 owner-Y의 보증인 역할을 수행한다. 이미 저장된 트랜잭션의 소유자에 의한 새로운 트랜잭션은 일반적인 (PoW) 합의 메커니즘의 필요성을 제거한다.

[0054] 본 발명의 DL은 개별 트랜잭션을 블록으로 캡슐화하지 않으므로 블록 모음이 아닌 트랜잭션 모음이 되고, 각 트랜잭션은 저장한다.

[0055] 각 트랜잭션은 프루닝 키(Pruning Key)를 저장하여 트랜잭션 소유자가 더 이상 필요하지 않은 트랜잭션을 삭제할 수 있도록 한다.

[0056] 각 피어는 프루닝 키를 사용하면, 피어가 다른 트랜잭션 소유자인 다른 피어(소유자)와 합의 후 불필요한 트랜잭션(거래)를 제거하여 정기적으로 분산 원부를 정리할 수 있다. 피어는 자원이 풍부한 노드이며, 각 피어는 원부 사본을 로컬에 저장한다. 그러나 피어가 원부를 로컬에 저장하기 위해 비정상적인 스토리지 용량을 포함할 필요가 없다.

[0057] 본 발명은 필요할 때마다 피어가 불필요한 트랜잭션을 안전하게 제거할 수 있는 원부 프루닝 메커니즘을 제안하기 때문에 원부가 정상적인 저장 용량을 가진 피어가 저장할 수 있을 만큼 컴팩트하다.

[0059] IoT 네트워크 분산 보안 관리 시스템(100)은 세 가지 기본 엔티티가 있다.

[0060] 첫째는 센서/액추에이터가 내장된 물리적인 것이다(예: 자동차, 냉장고, 문, 커피 메이커 등). 둘째는 물리적 사물과 컴퓨팅 노드 사이에서 데이터를 전송하는 통신 매체이다. 셋째는 리소스가 풍부한 컴퓨팅 노드의 P2P 네트워크이다.

[0061] 여기서, 각 엔티티의 역할을 개별적으로 정의한다.

[0062] 내장된 센서/액추에이터가 있는 물리적 사물은 단순하게 IoT 아키텍처의 인식 계층을 나타낸다.

[0063] 내장된 센서는 물리적 조건을 해당 디지털 형식으로 변환한다. 액추에이터는 디지털 신호를 적절한 물리적 동작으로 변환한다. 예를 들어, 자동차에 내장된 센서는 대기 온도를 디지털 형식으로 알려주고, 액추에이터는 컴퓨팅 장치에서 디지털 신호를 수신할 때 커피 메이커의 전원을 온시킨다.

[0064] 간단히 말하면, 센서와 액추에이터는 물리적인 사물이 인간의 삶을 용이하게하는 행동을 감지, 통신 및 수행할 수 있도록 한다.

[0065] 통신 계층(120)은 IoT 아키텍처의 네트워크 계층을 나타낸다. 통신 매체에는 센서/액추에이터와 컴퓨팅 노드간에 데이터를 전송할 수 있는 장치가 포함된다.

[0066] 통신에 사용되는 장치에는 블루투스, Wi-Fi, 근거리 통신 및 무선 센서 네트워크 등이 포함될 수 있다.

[0067] P2P 네트워크 계층(130)은 IoT 아키텍처의 두 계층, 즉 서비스에 대한 서비스를 제공하는 리소스가 풍부한 컴퓨팅 노드의 네트워크이다. 계층 및 응용 계층. 피어(컴퓨팅 노드)는 물리적 사물에서 보낸 데이터를 수신 및 처리하고 사용자의 요구 사항을 충족하기 위해 적절한 조치를 수행한다. 피어는 센서에서 보낸 데이터를 저장하기 위해 분산 원부(DL)를 유지한다. 각 피어는 자체 원부 사본을 유지한다. 원부의 보안 및 유효성은 새로운 OTS 체계(예: DL-OTS)의 도움으로 보장된다.

[0068] 센서 데이터는 거래 형태로 분산 원부에 저장된다. 센서가 데이터를 전송할 때 마다 담당 피어는 새로운 트랜잭션을 생성하고 다른 피어에게 전송한다.

[0069] 모든 다른 피어는 본 발명에서 제안된 OTS 방식의 도움으로 트랜잭션을 확인한 다음 원부에 저장한다. 새로운 트랜잭션이 액추에이터를 시작해야 함을 나타내면 해당 피어가 역할을 수행하고 적절한 액추에이터를 시작한다.

[0071] 본 발명에서 제안한 분산 장부(Distributed Ledger, DL) for 사물 인터넷 시스템(Industrial Internet of Things, IoT)의 기본 구성 요소는 해시 기반 일회성 서명(One-Time Signature, OTS) 방식이다.

[0072] 본 발명은 OTS 방식을 기존의 모든 OTS 방식에 비해 가장 작고 효율적인 방식인 DL-OTS 방식을 제공한다.

[0073] 각 피어는 분산 원부의 보안 및 유효성을 DL-OTS 방식으로 처리할 수 있다.

[0074] DL-OTS는 키 생성(Keygen), 서명(Sign), 확인(Verify), 키압축(Keycompress)의 4개의 구성요소로 이루어진다. Keygen은 보안 매개 변수(n)를 입력으로 받아들이고, 키 쌍(sk, pk)을 반환한다. sk는 개인 키와 pk는 공개 키를 나타낸다.

[0075] Sign은 메시지(m)와 개인 키(sk)를 입력으로 받아들이고, 즉, m의 서명(σ^m)을 반환한다.

[0076] Verify는 메시지(m), m의 서명(σ^m) 및 공개 키(pk)를 입력으로 받아들이고, 두 출력 중 하나를 반환하고, 성공하거나 실패한다. Keycompress는 공개 키를 암호화 방식으로 압축한다.

[0078] 키 생성(Key Generation)

[0079] DL(Distributed Ledger)-OTS(One-Time Signature)의 키 생성 프로세스를 설명한다. 해시 기반 OTS 체계에서 키와 서명은 모두 여러 값으로 구성된다(l 로 표시됨). DL-OTS에서 키와 서명은 각각 총 17개의 값으로 구성된다.

[0080] 이 중 16개 값은 서명할 메시지 해시의 가능한 알파벳에 해당한다. 즉, {0, 1, 2, · · · f} 및 17번째 값이 검사 합계(Checksum)로 청구된다(WOTS(Winternitz-OTS) 및 그 변형이 검사 합계로 사용하는 것과 같음). 또한, OTS 체계에서 개인 키(sk)를 공개 키(pk)로 변환하는데 여러 해시 반복(w로 표시)이 포함된다.

[0081] DL-OTS에서 각 sk 값(17번째 값 제외)은 48번 해시된다(DL-OTS를 고안하는 동안 수행된 실험 결과에 따라 48개를 선택함). 17번째 sk 값(체크섬에 사용됨)은 해당 pk 값을 생성하기 위해 " l * w" 시간(Wc로 표시)동안 해시

된다.

[0082] 마지막으로 DL-OTS는 하기의 수학식 1과 같이, 시드(Seed)로 알려진 단일 초기값으로부터 모든 sk 값들을 생성한다.

[0083] 본 발명은 적절한 레벨의 포스트 퀀텀 보안을 위해 해시 함수(SHA384)을 사용한다.

수학식 1

$$\sum_{i=1}^{l} sk_{i-1} = H^{i}(seed)$$

[0084]
$$\left[\sum_{i=0}^{l-2} pk_i = H^{w}(sk_i)\right] \cup \left[pk_{l-1} = H^{w_c}(sk_{l-1})\right]$$

[0085] 여기서, ski는 개인 키의 개별값, pki는 공개 키의 개별값, H는 해시 함수 SHA384, Wc는 마지막 sk 값(체크섬에 사용됨)을 해당 pk 값으로 변환하는데 사용되는 해시 반복 횟수, W는 sk 값을 해당 pk 값으로 변환하는데 사용된 해시 반복 횟수, l 은 키/서명 값의 수를 나타낸다.

[0087] 서명 생성(Signature Creation)

[0088] 서명 생성 프로세스는 서명할 메시지의 해시로 시작한다(예를 들어, hm = H(msg)). 본 발명은 16 진수 표현으로 메시지 해시를 처리한다.

[0089] SHA384를 사용하기 때문에 hm은 총 96개의 16 진수 기호로 구성된다(하기의 수학식 2). 심볼을 1부터 96까지 인덱싱한다.

수학식 2

[0090]
$$h_m = \sum_{i=1}^{96} m_i = \{m_1, m_2, \cdots, m_{96}\}$$

[0091] 다음으로 심볼에 포함된 인덱스를 다음과 같이 분류한다.

[0092] 심볼은 심볼 0을 포함하는 인덱스 세트, 심볼 1을 포함하는 인덱스 세트, 심볼 F(알고리즘 1의 단계 1 내지 3)을 포함하는 인덱스 세트로 구성된다.

[0093] 다음으로, 본 발명은 16개의 정수값을 얻기 위해 16개의 세트 각각에서 자릿수의 합계를 계산한다(단계 4 내지 5). 모듈러스 연산자(Modules Operator)를 사용하여 16개의 값이 모두 1에서 W까지의 범위에 있는지 확인한다(단계 6).

[0094] 다음 단계에서는 위의 계산된 16개 값에 대한 체크섬을 계산한다.

[0095] 본 발명은 W에서 16개 값을 각각 빼고, 이러한 모든 차이를 합산한다(단계 7 내지 8). 마지막으로 첫 번째 "l-1"인 sk 값의 사후 이미지와, 해당 정수값의 횟수를 계산하여 메시지의 서명을 생성한다(단계 9). 반면에 마지막 서명 요소는 체크섬(단계 10)과 동일한 횟수에 대한 l 번째 sk 요소의 사후 이미지를 계산하여 생성된다.

[0096] 도 3은 예제를 통해 서명 생성 프로세스를 설명한다.

[0097] 도 3은 예시 메시지인 "Good Message 707070"에 대한 DL-OTS 서명 생성을 나타낸 것이다.

[0098] 전술한 단계 1 내지 10의 서명 생성 알고리즘은 다음과 같다.

Algorithm 1 DL-OTS.*sign()*

Input: $h_m, [sk_0, sk_1, \cdots, sk_{l-1}]$
Output: $[\sigma_0, \sigma_1, \cdots, \sigma_{l-1}]$

1: $\sum_{i=0}^{l-2} \left[ind_i \leftarrow \phi \right]$
2: $j \leftarrow 1$
3: $\forall x \in h_m \bullet \left[\left[ind_x \leftarrow ind_x \cup j \right] \wedge \left[j \leftarrow j+1 \right] \right]$
4: $\sum_{i=0}^{l-2} \left[value_i \leftarrow 0 \right]$
5: $\sum_{i=0}^{l-2} \left[\forall d \in ind_i \bullet (value_i \leftarrow value_i + d) \right]$
6: $\sum_{i=0}^{l-2} \left[value_i \leftarrow (value_i \% w + 1) \right]$
7: $checksum \leftarrow 0$
8: $\sum_{i=0}^{l-2} \left[checksum \leftarrow checksum + (w - value_i) \right]$
9: $\sum_{i=0}^{l-2} \left[\sigma_i \leftarrow H^{value_i}(sk_i) \right]$
10: $\sigma_{l-1} \leftarrow H^{checksum}(sk_{l-1})$

[0099]

[0100] 서명 확인(Signature Verification)

[0101] 서명 확인 중에 검증자는 전술한 알고리즘 1의 단계 1 내지 8에 따라 16개 값과 체크섬을 계산한다.

[0102] 다음으로 검증자는 첫 번째 "l-1" σ-요소의 각각의 사후 이미지를 "W-(해당값)" 횟수로 계산한다. 이러한 방식으로 검증자는 총 l 개의 값들을 계산한다. 본 발명은 서명 키[vk]로서 이러한 값들의 세트를 확인할 수 있다.

[0103] 서명 키[vk]는 다음의 수학식 3에 의해 계산될 수 있다.

수학식 3

$$\left[\sum_{i=0}^{l-2} vk_i = H^{w-value_i}(\sigma_i) \right] \cup$$

$$\left[vk_{l-1} = H^{w_c - checksum}(\sigma_{l-1}) \right]$$

$$\left[\sum_{i=0}^{l-1} vk_i == pk_i \right] \implies \text{“}Accepted\text{”}$$

[0104]

[0105] 마지막으로 검증자는 계산된 값들의 각각을 해당하는 pk 값과 비교한다. 모든 해당 값이 같으면, 수학식 3과 같이, 검증자가 서명을 수락하고, 그렇지 않으면 서명이 거부되고 유효하지 않게 된다.

[0107] 키 압축(Key Compression)

[0108] 일반 공개 키는 l개의 값으로 구성되며 각 값은 384 비트이다. 그러나 Merkle 해시 트리를 사용하여 공개 키를 단일 384 비트 long 값으로 압축한다.

[0109] 도 4는 본 발명의 실시예에 따른 DL-OTS 키 압축을 나타낸 도면이다.

[0110] 도 4는 압축 트리의 구조를 설명하고, 하기의 알고리즘 2는 키 압축 절차를 자세히 설명한다.

[0111] pk는 총 17개의 해시 값으로 구성된다. DL-OTS는 단순한 Merkle 해시 트리를 사용하여 pk를 압축한다. 단계 1은 높이 4의 완벽한 이진 해시 트리인 배열 N을 정의한다. N의 길이는 31이다(높이 4의 완벽한 이진 트리는 총 31개의 노드를 포함하기 때문이다).

[0112] 완벽한 이진 트리(높이 4)에는 16개의 리프 노드가 있다. 단계 2는 N의 리프 노드(Leaf Node)에 pk 값들을 저장한다.

[0113] pk는 17개의 값으로 구성되며, 이 중 처음 16개 값은 N의 리프 노드에 저장된다(17번째 값은 N의 루트 노드와 연결되고 해시됨).

[0114] 단계 3은 새 변수 j를 시작하여 외부 while 루프를 작동한다(단계 4 내지 9). 외부 while 루프는 트리 높이와 같은 횟수만큼 반복된다. 외부 루프가 반복될 때 마다 트리의 새로운 상위 수준이 생성된다.

[0115] 내부 while 루프(단계 6 내지 8)는 해당 레벨의 노드 수와 동일한 횟수만큼 반복된다. 내부 루프의 각 반복은 트리의 새 노드를 계산한다. 새로운 부모 노드를 계산하기 위해 두 자식 노드를 연결하고 해시를 계산한다(단계 7). 마지막으로 단계 10에서 트리의 루트(N)를 마지막 pk 값과 연결하고 해시를 계산한다. 단계 10에서 계산된 값은 압축된 pk가 된다(PK로 표시됨).

[0116]

Algorithm 2 DL-OTS.*keyCompress()*

Input: $[pk_0, pk_1, \cdots, pk_{l-1}]$
Output: PK

1: define $N[(l-1)*2-1]$ ▷ Declares a perfect binary tree N of height 4
2: $\sum_{i=0}^{l-2} N[i+l-2] \leftarrow pk_i$ ▷ Stores pk_0 to pk_{15} in leaf nodes of N
3: define $j \leftarrow (l-1)*2-1$
4: **while** $j > 1$ **do** ▷ Loop iterates 4 times, and generates an upper level of N in each repetition
5: define $k \leftarrow (j/2)$
6: **while** $k < j$ **do** ▷ Computes a new node of N in each of the repetition
7: $N[k/2] \leftarrow H(N[k] + N[k+1])$
8: $k \leftarrow k+2$
9: $j \leftarrow j/2$
10: $PK \leftarrow H(N[0] + pk_{l-1})$ ▷ Computes compressed pk from root of N and 17^{th} pk-vale

[0117] 위의 변수의 설명을 간략하게 정리하면 다음과 같다.

[0118]

Symbol	Description
l	No. of values in key/signatures
n	Bit-length of an individual key/signature value
h_m	Hash of the message to be signed
l_k	No. of values in key
w	No. of hash iterations used to transform an sk-value to the corresponding pk-value
w_c	Number of hash iterations used to transform the last sk-value (used for checksum) to the corresponding pk-value
sk_i	An individual value in private key
pk_i	An individual value in public key
PK	The compressed public key
H	Hash function SHA384
$H^n(m)$	SHA384 applied on m for n times
σ^m	Signatures created on message m
vk	Verification key computed by the verifier from the signatures (to be compared with pk)
trx_i	i^{th} transaction stored in the ledger
$prnKey$	Pruning key (a PK to authorize a peer to remove an un-necessary transaction from DL)
$succID$	The ID of the transaction trx_i stored in the transaction trx_{i-1}
$\mathcal{F}$	A forger who accepts the challenge to break DL-OTS
$\mathcal{A}$	An adversary who accepts the challenge to break oneway-ness of H
$m^{\mathcal{Q}}$	Message queried by the $\mathcal{F}_{DL-OTS}$ during the attack process
$m^{\mathcal{F}}$	Message returned by the $\mathcal{F}_{DL-OTS}$ at the end of the attack process

[0120] IoT를 위한 포스트 퀀텀(Post Quantum) DL 제안

[0121] 본 발명의 DL은 일련의 트랜잭션으로 구성된다. 트랜잭션에는 다음 데이터 항목, 트랜잭션 시간, 센서 데이터(IoT 데이터), 프루닝 키(Pruning Key)(prnkey), 디지털 서명 및 후속 트랜잭션의 ID(succID)가 포함된다.

[0122] $trx_i[n]$을 사용하여 트랜잭션 i의 n번째 데이터 항목을 표시한다(여기서, $0 \leq n \leq 4$). succID와 prnkey는 본 발명에서 제안한 OTS 체계를 사용하여 생성 된 공개 키이다. 첫 번째 트랜잭션(즉, 제네시스 트랜잭션(Genesis Transaction))은 succID로만 구성된다.

[0123] 소유자(Owner)는 제네시스 트랜잭션을 생성한다. 각각의 새로운 거래는 이전 거래의 소유자와 합의한 후에, 가능한 이전 거래에서 ID를 얻는다.

[0124] 제네시스 트랜잭션(예: trx_0)을 생성하는 동안, 소유자는 제안된 OTS 체계(즉, DL-OTS)를 사용하여 공개 키를 생성하고 해당 공개 키를 제네시스 트랜잭션 (알고리즘 3)에 succID로 저장한다. 관련 개인 키(즉, seed)를 사용하면, trx_0의 소유자가 다른 노드가 새 트랜잭션을 생성하도록 승인할 수 있다.

[0125] 두 번째 트랜잭션의 소유자(예: trx_1)는 trx_0에 저장된 succID에 해당하는 개인 키를 사용하여 트랜잭션에 서명해야 한다. 검증 피어들은 trx_1의 서명을 따라 공개 키를 생성한다. 이러한 공개 키는 trx_0에 저장된 succID와 동일해야 한다.

[0126] 유사하게, trx_1에 저장된 succID는 trx_1의 소유자가 다른 노드에서 새로운 트랜잭션(예: trx_2)을 생성하도록 승인할 수 있도록 한다. trx_2의 소유자는 trx_1에 저장된 succID에 해당하는 개인 키를 사용하여 트랜잭션에 서명해야 한다.

[0127] 도 5는 본 발명의 실시예에 따른 트랜잭션 구조와 체인을 나타낸 도면이다.

[0128] 도 5는 트랜잭션의 구조와 체인을 설명하고, 알고리즘 4는 새로운 트랜잭션을 생성하는 프로세스를 설명한다.

Algorithm 3 DL-for-IoT.*createGenesis()*

Input: ϕ
Output: Genesis Transaction (trx_0)
1: $trx_0[0] \leftarrow trx_0[1] \leftarrow trx_0[2] \leftarrow trx_0[3] \leftarrow \phi$
2: $(sk,pk) \leftarrow$ *DL-OTS.keyGen()*
3: $trx_0[4] \leftarrow$ *DL-OTS.keyCompress(pk)*

[0129]

Algorithm 4 DL-for-IoT.*newTransaction()*

Input: sk_σ (sk corresponding to $trx_{i-1}[4]$), *IoT data*
Output: trx_i
1: $trx_i[0] \leftarrow$ *currentTime*
2: $trx_i[1] \leftarrow$ *IoT data*
3: $(sk_{prn},pk_{prn}) \leftarrow$ *DL-OTS.keyGen()*
4: $trx_i[2] \leftarrow$ *DL-OTS.keyCompress*(pk_{prn})
5: $trx_i[3] \leftarrow$ DL-OTS.sign$(trx_i[0]+trx_i[1]+trx_i[2], sk_\sigma)$
6: $(sk_{succ},pk_{succ}) \leftarrow$ *DL-OTS.keyGen()*
7: $trx_i[4] \leftarrow$ *DL-OTS.keyCompress*(pk_{succ})

[0130]

[0131] 거래 확인 및 수락(Transaction Verification and Acceptance)

[0132] 본 발명의 DL, 즉 DL-for-IoT에서 채택한 트랜잭션 확인 규칙을 설명한다.

[0133] 피어는 이미 DL에 트랜잭션이 저장되어 있고, 신뢰할 수 있는 다른 피어의 합의 후에 새로운 트랜잭션을 추가할 수 있다. 각각의 새로운 트랜잭션은 이전 트랜잭션에서 ID를 가져오며, 이는 이전 트랜잭션 소유자의 합의 후에만 가능하다.

[0134] 피어에서 새로운 트랜잭션을 제출하면, 모든 다른 피어들은 먼저 새로운 트랜잭션(알고리즘 5에 제공된 해당 절차)을 확인한 다음 원부에 추가한다.

[0135] 본 발명은 신뢰할 수 있는 피어만 거래를 원부에 저장하도록 허가할 수 있다.

[0136] 원부에 지분을 보유한 피어(트랜잭션 형태)는 신뢰할 수 있으며, 다른 피어가 새로운 트랜잭션을 추가하도록 허용할 수 있다. 새로운 피어도 이전 신뢰할 수 있는 피어에서 승인되었으므로 신뢰할 수 있다. 본 발명은 악성 피어를 위한 공간이 최소화된다.

[0137] 피어가 어떤 유형의 악의적인 행동을 저지르더라도 해당 피어와 승인자는 신뢰할 수 있는 피어에 의해 둘 다 감시할 수 있다. 원부는 대다수의 피어들이 정직하기만 하면 안전할 수 있다. 다음의 시나리오는 본 발명의 접근 방식을 이해하는데 도움이 된다.

[0139] 1. 피어(P1)는 초기 시드(추정 된 시드)를 무작위로 생성한다.

[0140] 2. $seed_a$에서 P1은 DL-ITS 개인 키(sk_a)를 생성한다.

[0141] 3. P1은 sk_a를 해당 공개 키(pk_a)로 변환한다.

[0142] 4. P1은 pk_a를 암호화 방식으로 압축한다. 압축된 공개 키는 succID를 나타낸다(succIDa 라고 칭함).

[0143] 5. P1은 트랜잭션에 succIDa를 포함합니다(trx_a).

[0144] 6. 다른 피어 (P2)가 새로운 트랜잭션(trx_b)을 원한다.

[0145] 7. P2는 $seed_a$ (succIDa에 해당)에 대해 P1을 요청한다.

[0146] 8. P1은 P2에 $seed_a$를 제공한다.

[0147] 9. P2는 $seed_a$에서 sk_a를 생성한다.

[0148] 10. P2는 sk_a를 사용하여 trx_b에 서명한다.

[0149] 11. 검증자는 trx_b의 서명에서 pk_a를 계산한다.

[0150] 12. Verifier는 succIDa를 생성하기 위해 pk_a를 압축한다.

[0151] 13. Verifier는 succIDa가 원부에 이미 존재하는지 확인한다.

[0152] 기존 신뢰할 수 있는 피어(P1)에서 제공한 시드가 없으면 새로운 피어(P2)가 자신의 트랜잭션에 서명할 수 없다.

[0154] 신뢰할 수 있는 피어와 악의적인 피어(Trusted vs Malicious Peers)

[0155] 본 발명은 신뢰할 수 있는 피어와 악의적인 피어를 어떻게 구분하는지 설명한다. 거래를 원부에 성공적으로 저장할 수있는 피어는 신뢰할 수 있는 피어이다.

[0156] 각각의 피어는 새로운 거래를 원부에 저장할 수 있으려면 거래 소유자(피어)가 거래의 유효한 서명을 제공해야 한다(그렇지 않으면 거래가 확인되지 않아 거부됨). 소유자(피어)는 제안 된 계획 "DL-OTS"를 사용하여 자신의 거래에 서명한다.

[0157] 서명은 이미 원부에 저장된 ID와 일치해야 한다. 해당 ID를 가지고 있는 이미 저장된 거래는 새로운 거래의 선행자 역할을 한다. 선행 거래의 소유자는 새로운 트랜잭션의 소유자에게 시드를 제공하여 새로운 트랜잭션에 서명할 수 있도록 한다.

[0158] 기존의 소유자가 새로운 소유자에게 시드(seed)를 제공하면, 새로운 소유자가 신뢰할 수 있는 사용자(User)/피어(Peer)임을 확인할 수 있다.

[0159] 각각의 새로운 피어는 자신의 거래를 원부에 저장할 수 있도록 동일한 절차를 따르기 때문에 원부에 거래가 저장된 모든 피어가 신뢰할 수 있는 피어이다. 이미 저장된 ID에 해당하는 서명을 제공하지 못한 피어는 악의적인 피어로 식별되고 트랜잭션이 거부된다.

[0161] 원부 정리(Ledger Pruning)

[0162] DL-for-IoT를 사용하면, 불필요한 트랜잭션을 제거하여 원부 확장성을 제공 할 수 있다. 소유자는 더 이상 필요하지 않은 거래를 삭제할 수 있다. 트랜잭션(예: trx_i)을 제거하려면 trxi의 소유자는 이미 trx_i에 저장된 프루닝 키(Pruning Key)(prnkey)에 해당하는 개인 키를 사용하여 서명해야 한다.

[0163] 다른 피어는 먼저 서명을 확인한 다음 해당 트랜잭션을 제거한다. trx_i를 제거하기 위해 저장된 succID는 트랜잭션 $trx_{(i-1)}$(알고리즘 6)에 복사된다.

[0164] 이전에 trx_i에 저장된 succID는 기본적으로 트랜잭션의 ID인 $trx_{(i+1)}$이다. 이러한 succID를 $trx_{(i-1)}$에 복사한 후, $trx_{(i+1)}$은 trx_i의 위치를 얻는 반면, 이전 trx_i는 체인(원부)을 떠난다.

[0165] 도 6은 본 발명의 실시예에 따른 원부 정리 모습을 나타낸 도면이다.

[0166] 도 6은 트랜잭션 제거 전후의 DL 이미지를 보여줍니다.

Algorithm 6 DL-for-IoT.*removeTransaction()*

Input: σ_{trx_i} (signatures on transaction to be removed, created using the corresponding sk_{prn})
Output: DL with trx_i removed from it

1: $vk \leftarrow DL\text{-}OTS.verificationKey(trx_i, \sigma_{trx_i})$
2: $vkCompressed \leftarrow DL\text{-}OTS.keyCompress(vk)$
3: **if** $vkCompressed = trx_i[2]$ **then**
4: $\quad trx_{i-1}[4] \leftarrow trx_i[4]$

[0167]

[0168] 도 7은 본 발명의 실시예에 따른 DL-for-IOT를 통합한 스마트 홈 시스템의 일례를 나타낸 도면이다. 도 7은 DL-for-IoT를 통합한 스마트 홈의 아키텍처를 설명한 것이다.

[0169] DL-for-IOT를 통합한 스마트 홈 시스템

[0170] DL-for-IOT를 통합한 스마트 홈 시스템은 마이너, 풀 노드(에이전트라고도 함) 및 IoT 노드를 포함한다. IoT 노드(센서 등)는 데이터를 생성하고, 채굴자는 데이터를 원부에 저장한다. 에이전트는 IoT 노드를 대신하여 채굴자와 통신한다.

[0171] IoT 노드가 블록 체인 노드와 직접 통신하는 스마트 홈 시스템을 제안한다. 이러한 스마트 홈 시스템에서 블록 체인 노드는 마이너와 에이전트의 부하를 공유한다.

[0172] 본 발명은 IoT 노드가 피어와 직접 통신할 수 있다. 피어는 센서 데이터를 수신하고, 원부에 저장할 수 있을 만큼 충분한 자원이 있다.

[0173] 상당히 가벼운 트랜잭션 검증 및 합의 메커니즘으로 인해 에이전트의 역할을 제거할 수 있다. 본 발명의 스마트 홈 시스템은 IoT 장치가 키를 생성하거나 저장한다고 가정하지 않지만, 피어는 IoT 노드 그룹을 대신하여 이러한 작업을 수행한다.

[0174] 피어는 관련 IoT 센서에서 수신한 데이터를 저장하기 위해 DL을 유지 관리할 책임이 있다. 데이터 조각을 DL에 저장하는 동안 피어는 두 개의 OTS 키 쌍(즉, succID 및 prnKey)을 생성한다. 각 피어는 자체 키를 로컬로 유지한다.

[0175] 키 쌍을 저장한다는 것은 두 개의 해시 결과, 즉 seed 및 압축된 공개 키만 저장하는 것을 의미한다(각 384 비트 길이). 따라서, 디스크에 있는 키의 총 크기는 0.096KB이다. 본 발명은 DL-for-IOT를 통합한 스마트 홈 시스템을 다음과 같이 요약할 수 있다.

[0176] 두 가지 유형의 노드, IoT 노드 및 피어가 있다.

[0177] IoT 노드(예: 센서 등)는 데이터를 생성한다. 피어는 이중 역할을 하고, 각 피어는 일련의 IoT 노드에 대한 에이전트 역할을 수행한다. 피어는 원부를 유지할 수 있고, IoT 데이터를 원부에 저장한다.

[0178] 센서가 생성한 데이터는 해당 센서의 에이전트 역할을 하는 피어가 원부에 저장(마이닝)한다.

[0179] 피어는 센서 데이터를 원부에 마이닝하는데 필요한 키를 만들고 저장한다. 각 피어는 해당 키를 로컬로 유지할 수 있다.

[0180] 이상에서 본 발명의 실시예에 대하여 상세하게 설명하였지만 본 발명의 권리범위는 이에 한정되는 것은 아니고 다음의 청구범위에서 정의하고 있는 본 발명의 기본 개념을 이용한 당업자의 여러 변형 및 개량 형태 또한 본 발명의 권리범위에 속하는 것이다.

부호의 설명

[0181] 100: IoT 네트워크 분산 보안 관리 시스템

110: 인식 계층

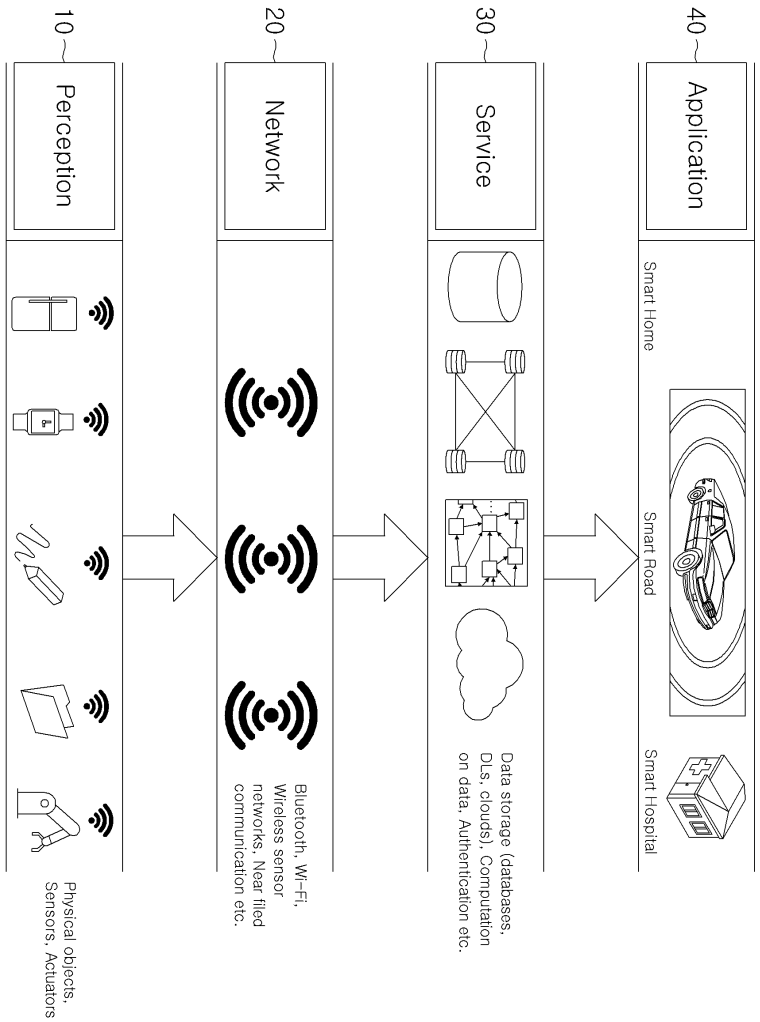
120: 통신 계층

130: P2P 네트워크 계층

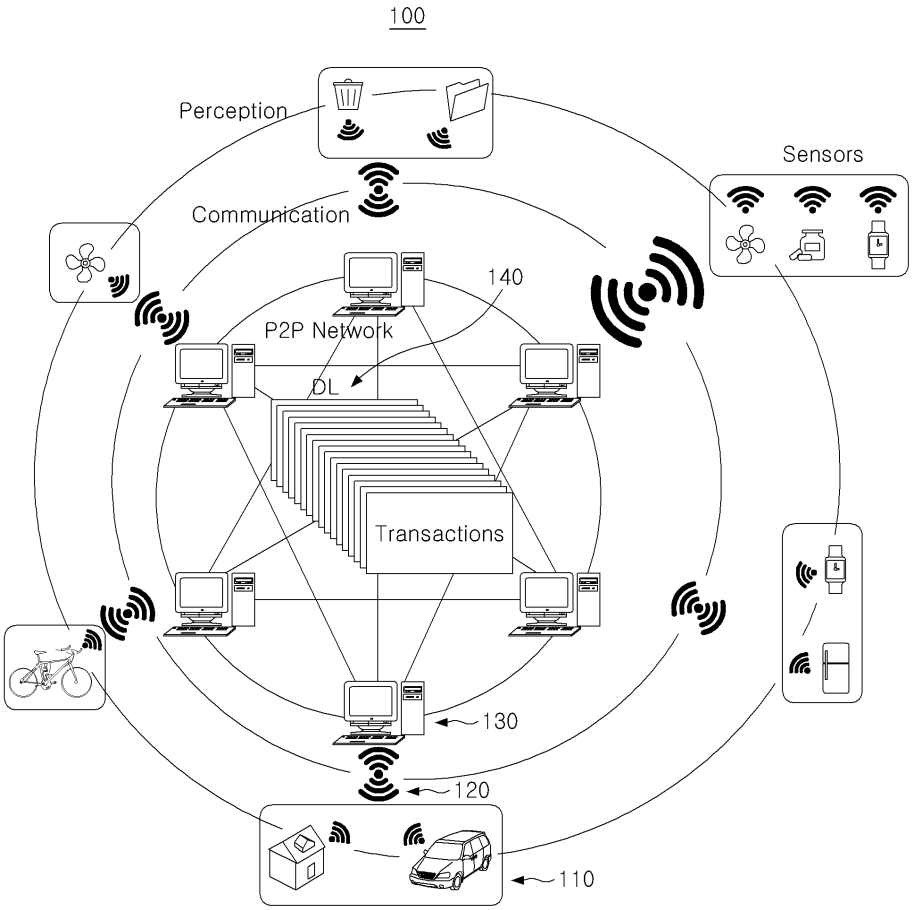
140: DL 계층

도면

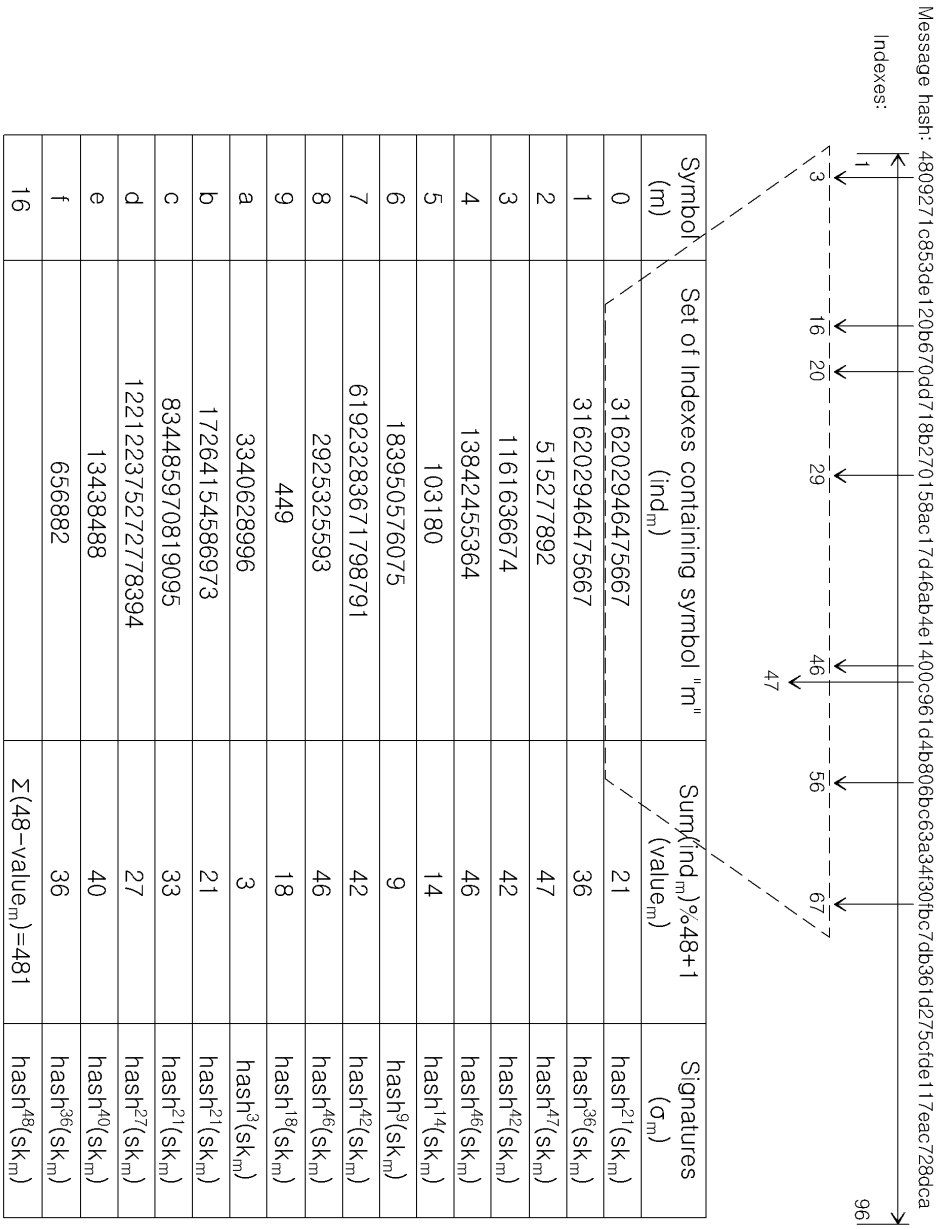
도면1



도면2

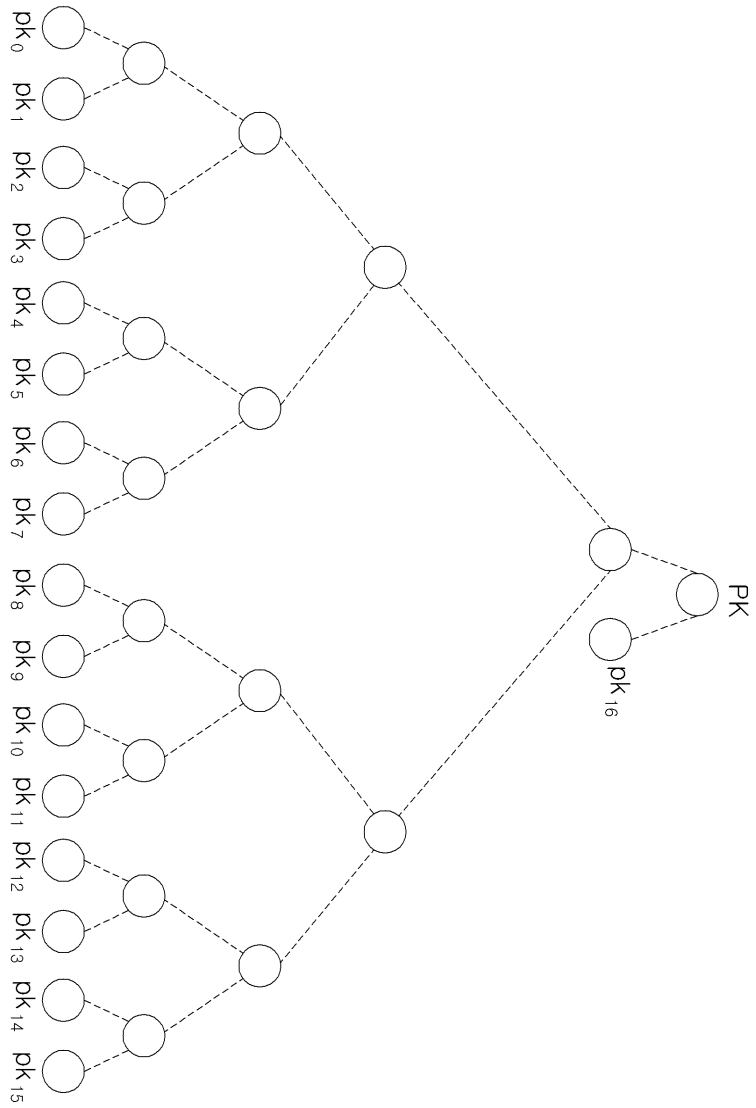

100
Perception
Sensors
Communication
140
P2P Network
DL
Transactions
130
120
110

도면3

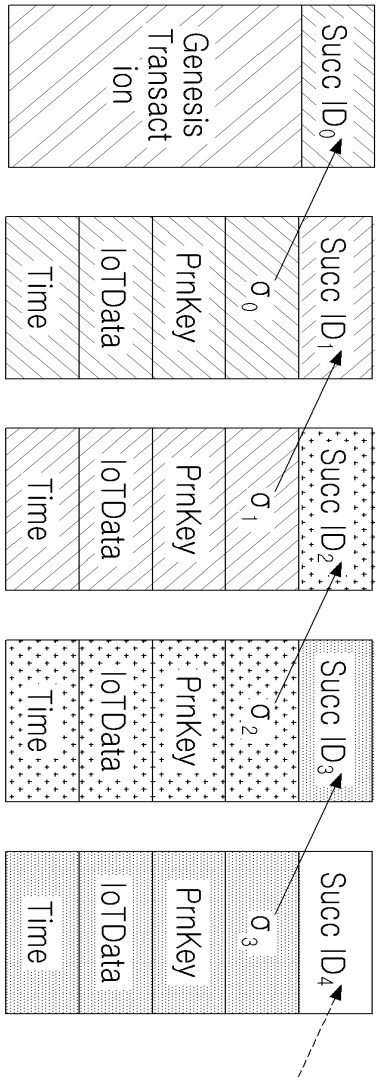


Symbol (m)	Set of Indexes containing symbol "m" (ind_m)	$Sum(ind_m)\%48+1$ ($value_m$)	Signatures (σ_m)
0	3162029464756667	21	$hash^{21}(sk_m)$
1	3162029464756667	36	$hash^{36}(sk_m)$
2	515277892	47	$hash^{47}(sk_m)$
3	1161636674	42	$hash^{42}(sk_m)$
4	13842455364	46	$hash^{46}(sk_m)$
5	103180	14	$hash^{14}(sk_m)$
6	183950576075	9	$hash^{9}(sk_m)$
7	619232836717798791	42	$hash^{42}(sk_m)$
8	2925325593	46	$hash^{46}(sk_m)$
9	449	18	$hash^{18}(sk_m)$
a	3340628996	3	$hash^{3}(sk_m)$
b	172641545869 73	21	$hash^{21}(sk_m)$
c	834485970819095	33	$hash^{21}(sk_m)$
d	12212237527278394	27	$hash^{27}(sk_m)$
e	13438488	40	$hash^{40}(sk_m)$
f	656882	36	$hash^{36}(sk_m)$
16		$\Sigma(48-value_m)=481$	$hash^{48}(sk_m)$

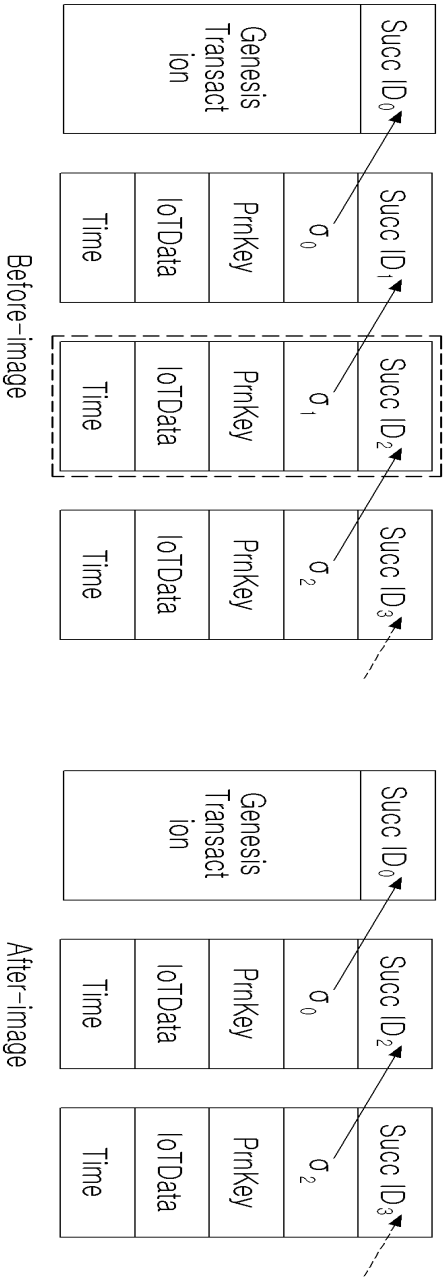
도면4


pk0
pk1
pk2
pk3
pk4
pk5
pk6
pk7
pk8
pk9
pk10
pk11
pk12
pk13
pk14
pk15
pk16
PK

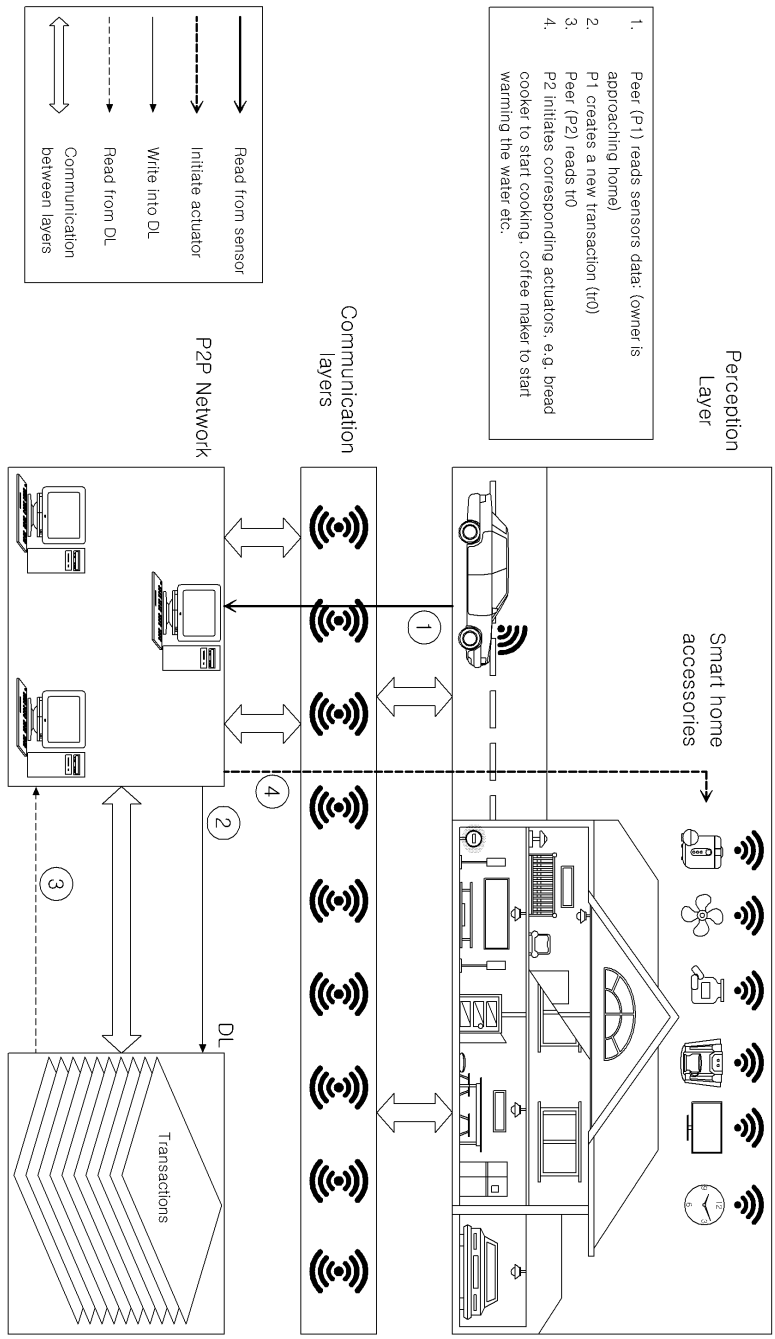
도면5


Genesis Transaction
Succ ID0
Time
IoTData
PrnKey
σ0
Succ ID1
Time
IoTData
PrnKey
σ1
Succ ID2
Time
IoTData
PrnKey
σ2
Succ ID3
Time
IoTData
PrnKey
σ3
Succ ID4

도면6


Before-image
Genesis Transaction
Succ ID0
Succ ID1
σ0
PrnKey
IoTData
Time
Succ ID2
σ1
Succ ID3
σ2
After-image

도면7


1. Peer (P1) reads sensors data: (owner is approaching home)
2. P1 creates a new transaction (tr0)
3. Peer (P2) reads tr0
4. P2 initiates corresponding actuators. e.g. bread cooker to start cooking. coffee maker to start warming the water etc.
Read from sensor
Initiate actuator
Write into DL
Read from DL
Communication between layers
Perception Layer
Smart home accessories
Communication layers
P2P Network
DL
Transactions
1
2
3
4