(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2020년09월03일
(11) 등록번호 10-2151746
(24) 등록일자 2020년08월28일

(51) 국제특허분류(Int. Cl.)
H04L 9/08 (2006.01) *G06Q 20/38* (2012.01)
H04L 29/08 (2006.01)
(52) CPC특허분류
H04L 9/0894 (2013.01)
G06Q 20/382 (2013.01)
(21) 출원번호 10-2018-0162895
(22) 출원일자 2018년12월17일
심사청구일자 2018년12월17일
(65) 공개번호 10-2020-0074474
(43) 공개일자 2020년06월25일
(56) 선행기술조사문헌
KR101025029 B1*
KR101778768 B1*
KR1020180104586 A*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
(주)에스유지
부산광역시 동래구 충렬대로238번길 31 ,태웅빌딩301호(수안동)
(72) 발명자
김창관
부산시 동래구 사직북로 64, 203동 1704호
설동철
경남 창원시 성산구 원이대로 495 트리비앙 222-1301
(74) 대리인
정병홍

전체 청구항 수 : 총 3 항 심사관 : 임대식

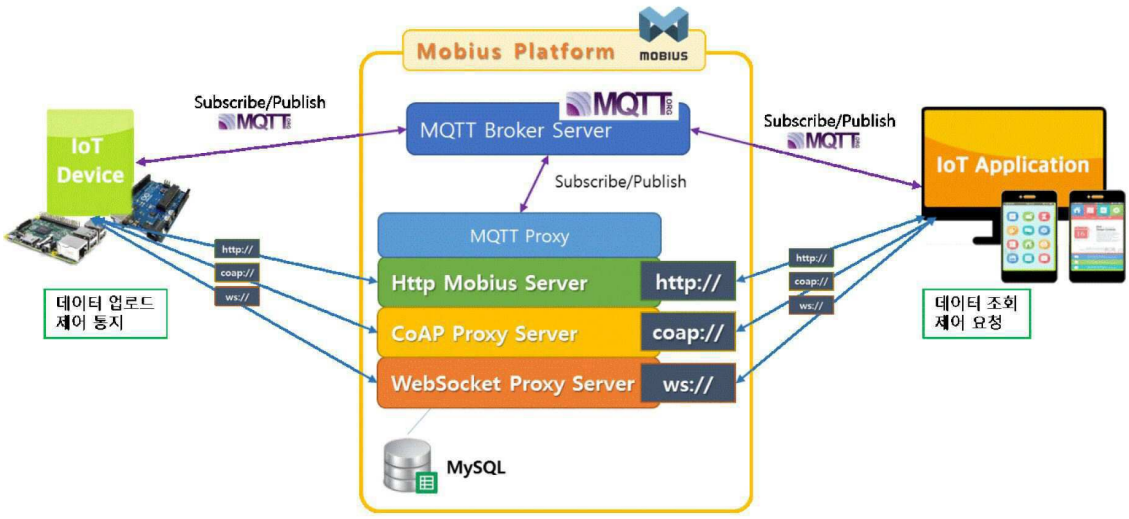
(54) 발명의 명칭 **블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법**

(57) 요 약

본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법은 블록체인을 데이터베이스 서버처럼 구성하고 IoT 플랫폼에 적용하여 데이터의 위변조를 막는 데이터 저장 방법에 있어서, IoT 디바이스와 사용자 APP을 연결하는 단계(S100); 상기 디바이스가 상기 디바이스의 비밀 키를 서명한 값, 지갑주소 및 데이터를 IoT 플

(뒷면에 계속)

대 표 도



랫폼으로 전달하는 단계(S200); 상기 IoT 플랫폼이 데이터베이스에서 상기 디바이스의 지갑주소에 매칭되는 사용자 지갑주소를 찾는 단계(S300); 상기 IoT 플랫폼이 JSON-RPC를 이용하여 API 함수를 호출하고 값을 블록체인 네트워크로 전달하는 단계(S400); 상기 네트워크는 스마트 컨트랙트에 작성된 내용으로 동작하여 데이터를 블록에 저장하는 단계(S500); 사용자가 자신의 지갑주소와 상기 사용자가 원하는 데이터 조건 값을 상기 IoT 플랫폼으로 전달하는 단계(S600); 상기 사용자가 자신의 지갑주소와 상기 디바이스 상태 변경 값, 비밀 키를 서명한 값을 상기 IoT 플랫폼으로 전달하는 단계(S700); 상기 IoT 플랫폼은 전달받은 상기 디바이스 상태 변경 값으로 상기 디바이스의 상태를 변경한 후, API 함수를 호출하여 상태 값을 상기 블록체인 네트워크로 전달하는 단계(S800) 및 상기 블록 체인 네트워크는 스마트 컨트랙트에 작성된 값으로 동작하여 블록에 상태값을 저장하는 단계(S900)를 포함하는 것을 특징으로 한다.

(52) CPC특허분류

H04L 67/16 (2013.01)

H04L 9/0825 (2013.01)

H04L 2209/38 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호	P0006910
부처명	산업통상자원부
과제관리(전문)기관명	한국산업기술진흥원
연구사업명	국가혁신클러스터사업(R&D)
연구과제명	블록체인 기반 선용품 거래추천 시스템 개발
기 여 율	100/100
과제수행기관명	(주)에스유지
연구기간	2018.10.01 ~ 2018.12.31

명 세 서

청구범위

청구항 1

블록체인을 데이터베이스 서버처럼 구성하고 IoT 플랫폼에 적용하여 데이터의 위변조를 막는 데이터 저장 방법에 있어서,

IoT 디바이스와 사용자 APP을 연결하는 단계(S100);

상기 디바이스가 상기 디바이스의 비밀 키를 서명한 값, 지갑주소 및 데이터를 IoT 플랫폼으로 전달하는 단계(S200);

상기 IoT 플랫폼이 데이터베이스에서 상기 디바이스의 지갑주소에 매칭되는 사용자 지갑주소를 찾는 단계(S300);

상기 IoT 플랫폼이 JSON-RPC를 이용하여 API 함수를 호출하고 값을 블록체인 네트워크로 전달하는 단계(S400);

상기 네트워크는 스마트 컨트랙트에 작성된 내용으로 동작하여 데이터를 블록에 저장하는 단계(S500);

사용자의 명령에 의해 IoT 디바이스가 사용자 APP를 통해 사용자의 지갑주소와 상기 사용자가 원하는 데이터 조건 값을 상기 IoT 플랫폼으로 전달하는 단계(S600);

상기 사용자가 자신의 지갑주소와 상기 디바이스의 상태 변경 값, 비밀 키를 서명한 값을 상기 IoT 플랫폼으로 전달하는 단계(S700);

상기 IoT 플랫폼은 전달받은 상기 디바이스 상태 변경 값으로 상기 디바이스의 상태를 변경한 후, API 함수를 호출하여 상태 값을 상기 블록체인 네트워크로 전달하는 단계(S800) 및

상기 블록 체인 네트워크는 스마트 컨트랙트에 작성된 값으로 동작하여 블록에 상태값을 저장하는 단계(S900)를 포함하고,

상기 스마트 컨트랙트는,

클라이언트에서 보호 할 데이터를 입력 받고 데이터를 해시하여 공개키를 생성하는 등록 단계;

데이터베이스에 실제 전송 받은 원본 데이터를 저장하는 단계;

블록체인에는 보호 할 원본 데이터는 전송하지 않고 데이터를 통해 생성 된 공개키를 전송하여 블록에 저장하여 등록을 완료하는 단계;

데이터를 호출 시 인증하는 단계;

클라이언트가 서버를 통해 데이터베이스에 저장한 원본 데이터를 호출하게 되면 블록체인에 저장된 공개키를 통해 조회하는 단계;

인증 과정의 서버는 증명자가 되어 전송받은 공개키를 통해 Zero-Knowledge Proof에 의하여 검증하는 단계;로 동작하는 것을 특징으로 하는 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법.

청구항 2

청구항 1에 있어서,

상기 단계(S100)에서 IoT 디바이스와 사용자 APP을 연결할 때, 상기 디바이스와 상기 사용자 각각의 지갑주소를 블록체인 네트워크에서 발급받는 것을 특징으로 하는 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법.

청구항 3

청구항 1에 있어서,

상기 단계(S600)에서 지갑주소와 데이터 조건 값을 상기 IoT 플랫폼으로 전달할 때 Post, Get 방식을 이용하는 것을 특징으로 하는 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 블록체인 기반의 IoT플랫폼을 활용하여 위변조를 막는 보안성이 강화된 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 관한 것이다.

배 경 기 술

[0003] IT기술의 발전으로 현재 4차 산업 혁명으로 도약하고 있다. 사물인터넷(IoT)은 각종 사물에 센서와 통신 기능들이 내장되어 인터넷에 연결하는 기술이다. 하지만 IoT는 서비스의 목적에 따라서 시스템 자원이 제한되어 있기 때문에 서버 및 네트워크 보안이 적용되기 어렵다. IoT 보안에 취약한 점을 연구한 사례들이 지속적으로 이루어지고 있다. IoT 서버 플랫폼 중에서도 특히, Mobius는 oneM2M 표준 규격을 따르는 IoT 디바이스들을 인증하고 실시간 센서 데이터를 받아서 MySQL 서버에 정보와 데이터를 저장하여 관리하는 방식으로 사용할 수 있다. 하지만 Mobius 구성도의 MySQL은 보안에 취약점과 위협이 되는 요소들이 많으며, 아직 해결되지 않는 부분들이 있다.

선행기술문헌

특허문헌

[0005] (특허문헌 0001) KR 10-1880175 B1

(특허문헌 0002) KR 10-1701131 B1

발명의 내용

해결하려는 과제

[0006] 따라서, 본 발명은 상술한 문제점을 해결하기 위한 것으로, 블록체인을 도입하여 새로운 IoT 플랫폼을 구성하고, 센서 데이터를 블록체인에 저장하여 공격자로부터 데이터 위변조를 막는 방법을 제안한다.

과제의 해결 수단

[0008] 상기의 목적을 이루기 위한 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법은 블록체인을 데이터베이스 서버처럼 구성하고 IoT 플랫폼에 적용하여 데이터의 위변조를 막는 데이터 저장 방법에 있어서, IoT 디바이스와 사용자 APP을 연결하는 단계(S100); 상기 디바이스가 상기 디바이스의 비밀 키를 서명한 값, 지갑주소 및 데이터를 IoT 플랫폼으로 전달하는 단계(S200); 상기 IoT 플랫폼이 데이터베이스에서 상기 디바이스의 지갑주소에 매칭되는 사용자 지갑주소를 찾는 단계(S300); 상기 IoT 플랫폼이 JSON-RPC를 이용하여 API 함수를 호출하고 값을 블록체인 네트워크로 전달하는 단계(S400); 상기 네트워크는 스마트 컨트랙트에 작성된 내용으로 동작하여 데이터를 블록에 저장하는 단계(S500); 사용자가 자신의 지갑주소와 상기 사용자가 원하는 데이터 조건 값을 상기 IoT 플랫폼으로 전달하는 단계(S600); 상기 사용자가 자신의 지갑주소와 상기 디바이스 상태 변경 값, 비밀 키를 서명한 값을 상기 IoT 플랫폼으로 전달하는 단계(S700); 상기 IoT 플랫폼은 전달받은 상기 디바이스 상태 변경 값으로 상기 디바이스의 상태를 변경한 후, API 함수를 호출하여 상태 값을 상기 블록체인 네트워크로 전달하는 단계(S800) 및 상기 블록 체인 네트워크는 스마트 컨트랙트에 작성된 값으로 동작하여 블록에 상태값을 저장하는 단계(S900)를 포함하는 것을 특징으로 한다.

발명의 효과

[0010] 본 발명은 IoT 데이터를 블록체인에 저장하여 IoT 디바이스 인증 및 데이터 위, 변조를 막을 수 있는 효과를 갖는다. 또한 기존의 데이터베이스를 그대로 사용하고 블록체인을 추가하여 데이터의 공개키만 생성해 저장하는 방식으로 무결성과 기밀성이 있다는 효과를 갖는다.

도면의 간단한 설명

[0012] 도 1은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 이더리움의 스마트 컨트랙트의 간략한 작성 실시 예이다.

도 2는 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 이더리움의 스마트 컨트랙트가 동작하는 과정이다.

도 3은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법의 모비우스와 IoT 디바이스 연동 과정을 보여준다.

도 4는 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법의 모비우스 서버 플랫폼 시스템 구성도이다.

도 5는 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 모비우스 Yellow Turtle S/W 아키텍처이다.

도 6은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 SCC 시스템의 구성도이다.

도 7은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에서 제안하는 IoT 서버 플랫폼 아키텍처이다.

도 8은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법의 데이터 저장 상세 설계도이다.

도 9는 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법의 데이터 조회 상세 설계도이다.

도 10은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법의 디바이스 조작 상세 설계도이다.

도 11은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 스마트 컨트랙트의 데이터 자료 구조 설계도이다.

도 12는 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 스마트 컨트랙트 저장 기능을 구현한 실시 예이다.

도 13은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 스마트 컨트랙트 조회 기능을 구현한 실시 예이다.

도 14는 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법의 디바이스와 사용자 매핑 관리 설계 실시 예이다.

도 15는 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 Ring Signatures 기법이다.

도 16은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 Zero-Knowledge Proof이다.

도 17은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 Zero-Knowledge Proof의 인증 프로토콜이다.

도 18은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 Zero-Knowledge Proof의 데이터 보호 방법을 보여준다.

도 19는 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 Zero-Knowledge Proof의 데이터의 공개키 생성 및 저장과정이다.

도 20은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 Zero-

Knowledge Proof의 데이터 인증과정이다.

도 21은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 다량의 데이터 검색을 위해 개선한 데이터 검색 과정이다.

도 22는 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 다량의 데이터 검색을 위해 개선한 스마트 컨트랙트 소스이다.

발명을 실시하기 위한 구체적인 내용

[0013] 이하에서는, 본 발명의 실시 예에 따른 도면을 참조하여 설명하지만, 이는 본 발명의 더욱 용이한 이해를 위한 것으로, 본 발명의 범주가 그것에 의해 한정되는 것은 아니다.

[0014] 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 있음을 의미한다. 또한, 명세서에 기재된 "...부", "모듈" 등의 용어는 적어도 하나의 기능이나 동작을 처리하는 단위를 의미하며, 이는 하드웨어 또는 소프트웨어로 구현되거나 하드웨어와 소프트웨어의 결합으로 구현될 수 있다.

[0015] 명세서 전체에서, 어떤 부분이 다른 부분과 "연결"되어 있다고 할 때, 이는 "직접적으로 연결"되어 있는 경우뿐 아니라, 그 중간에 다른 소자를 사이에 두고 "전기적으로 연결"되어 있는 경우도 포함한다.

[0017] 본 명세서에 있어서는 어느 하나의 구성요소가 다른 구성요소로 데이터 또는 신호를 '전송'하는 경우에는 구성요소는 다른 구성요소로 직접 상기 데이터 또는 신호를 전송할 수 있고, 적어도 하나의 또 다른 구성요소를 통하여 데이터 또는 신호를 다른 구성요소로 전송할 수 있음을 의미한다.

[0019] 그리고 본 명세서에서 모듈이라 함은, 본 발명의 기술적 사상을 수행하기 위한 하드웨어 및 상기 하드웨어를 구동하기 위한 소프트웨어의 기능적, 구조적 결합을 의미할 수 있다. 예컨대, 상기 모듈은 소정의 코드와 상기 소정의 코드가 수행되기 위한 하드웨어 리소스의 논리적인 단위를 의미할 수 있으며, 반드시 물리적으로 연결된 코드를 의미하거나, 한 종류의 하드웨어를 의미하는 것은 아님은 본 발명의 기술분야의 평균적 전문가에게는 용이하게 추론될 수 있다.

[0021] 설명에 앞서 본 명세서에는 다수의 양태 및 실시양태가 기술되며, 이들은 단순히 예시적인 것으로서 한정하는 것이 아니다.

[0022] 본 명세서를 읽은 후에, 숙련자는 다른 양태 및 실시예가 본 발명의 범주로부터 벗어남이 없이 가능함을 이해할 것이다.

[0024] 이하에서 설명되는 실시양태의 상세 사항을 다루기 전에, 몇몇 용어를 정의하거나 또는 명확히 하기로 한다.

[0026] 블록체인이란 관리 대상 데이터를 '블록'이라고 하는 소규모 데이터들이 P2P 방식을 기반으로 생성된 체인 형태의 연결고리 기반 분산 데이터 저장환경에 저장되어 누구라도 임의로 수정할 수 없고 누구나 변경의 결과를 열람할 수 있는 분산 컴퓨팅 기술 기반의 데이터 대변 방지 기술이다.

[0028] JSON-RPC란 JSON으로 인코딩된 원격 프로시저 호출로 간단한 프로토콜을 의미한다. 소량의 데이터 타입과 명령들만을 정의하고 있다.

[0030] 스마트 컨트랙트란 블록체인 기반으로 금융거래, 부동산 계약, 공증 등 다양한 형태의 계약을 체결하고 이행하는 것을 말한다.

[0032] GET과 POST는 HTTP프로토콜을 이용해서 서버에 전달할 때 사용하는 방식이다.

[0034] 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법은 블록체인을 데이터베이스 서버처럼 구성하고 IoT 플랫폼에 적용하여 데이터의 위변조를 막는 데이터 저장 방법에 있어서, IoT 디바이스와 사용자 APP을 연결하는 단계(S100); 상기 디바이스가 상기 디바이스의 비밀 키를 서명한 값, 지갑주소 및 데이터를 IoT 플랫폼으로 전달하는 단계(S200); 상기 IoT 플랫폼이 데이터베이스에서 상기 디바이스의 지갑주소에 매칭되는 사용자 지갑주소를 찾는 단계(S300); 상기 IoT 플랫폼이 JSON-RPC를 이용하여 API 함수를 호출하고 값을 블록체인 네트워크로 전달하는 단계(S400); 상기 네트워크는 스마트 컨트랙트에 작성된 내용으로 동작하여 데이터를 블록에 저장하는 단계(S500); 사용자가 자신의 지갑주소와 상기 사용자가 원하는 데이터 조건 값을 상기 IoT 플랫폼으로 전달하는 단계(S600); 상기 사용자가 자신의 지갑주소와 상기 디바이스 상태 변경 값, 비밀 키를 서명한 값을 상기 IoT 플랫폼으로 전달하는 단계(S700); 상기 IoT 플랫폼은 전달받은 상기 디바이스 상태 변경 값으로

상기 디바이스의 상태를 변경한 후, API 함수를 호출하여 상태 값을 상기 블록체인 네트워크로 전달하는 단계(S800) 및 상기 블록 체인 네트워크는 스마트 컨트랙트에 작성된 값으로 동작하여 블록에 상태값을 저장하는 단계(S900)를 포함하는 것을 특징으로 한다.

[0036] 또한, 상기 단계(S100)에서 IoT 디바이스와 사용자 APP을 연결할 때, 상기 디바이스와 상기 사용자 각각의 지갑 주소를 블록체인 네트워크에서 발급받는 것을 특징으로 한다.

[0038] 또한, 상기 단계(S600)에서 지갑주소와 데이터 조건 값을 상기 IoT 플랫폼으로 전달할 때 Post, Get 방식을 이용하는 것을 특징으로 한다.

[0040] 본 발명에서는 블록체인을 도입하여 새로운 IoT 플랫폼을 구성하고, 센서 데이터를 블록체인에 저장하여 공격자로부터 데이터 위변조를 막는 방법을 제안한다. IoT 서버 플랫폼 중에서 Mobius를 선택하였고, Mobius는 oneM2M 표준 규격을 따르는 IoT 디바이스들을 인증하고 실시간 센서 데이터를 받아서 MySQL 서버에 정보와 데이터를 저장하여 관리하는 방식이다. 하지만 Mobius 구성도의 MySQL은 보안에 취약점과 위협이 되는 요소들이 많으며, 아직 해결되지 않는 부분들이 있다. 이런 부분들을 블록체인을 도입하여 데이터를 기존에 서버 구성방식에서 MySQL 서버 같은 범용/통용적인 서버 구축 방식이 아닌, 블록체인을 데이터베이스처럼 구축하여 데이터 저장 방법을 제안한다.

[0041] 블록체인은 첫 번째 가상화폐인 비트코인이 나타나면서 각광 받은 기술이다. 모든 가상화폐는 각자에 기능을 부각 시키는 블록체인을 개발되어 나온다. 기존의 금융권 거래 방식인 중앙방식이 아닌 탈 중앙 방식이다. 따라서, 모든 사용자의 거래를 모아서 제 3자인 채굴자가 일정한 주기마다 블록을 찾아 생성하고 보상을 받으며, 중앙에서 관리하는 것이 아닌 여러 사람이 채굴자가 검증한 장부를 가지게 되는 공공 거래 장부 개념이다.

[0043] 다음은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 이더리움에 대해 상세히 설명하도록 한다.

[0044] 이더리움은 비탈릭 부테린(Vitalik Buterin)이 개발한 암호화폐이며 플랫폼이다. 기존에 비트코인에서 파생 되어 나온 가상화폐이다. 비트코인이 결제나 거래 관련 시스템 즉 화폐로서의 기능에 집중하는 반면, 이더리움은 핵심 기술인 블록체인(Block Chain)을 기반으로 거래나 결제 뿐 아니라 계약서, SNS, 이메일, 전자투표 등 다양한 애플리케이션을 투명하게 운영할 수 있게 확장성을 제공한다. 블록체인 기반이다 보니 이것들을 당연히 분산 애플리케이션(decentralized application)이 된다. 그래서 이것을 줄여서 DApp 또는 dApp(댑)이라고 부른다. 이더리움의 속성과 특징은 하기의 표 1과 같다.

튜링 완정성 (Turing-Completeness)	이더리움에서 Smart Contract를 사용하기 위해 만든 독자적인 언어이다. Solidity 언어라고 한다.
플랫폼을 통한 응용성 (DApps on Platform)	하나의 서비스가 아니라, 서비스를 창조해낼 수 있는 거대한 플랫폼이기 때문에 무한한 응용이 가능하다.
스마트 컨트랙트 (Smart Contract)	여러 가지 계약을 창조해낼 수 있으며, 해당 계약을 이행하는 것도 가제적으로 만들 수 있다. 즉, 하지만 파기할 수 없는 디지털 계약을 만들 수 있다.

[0045]

[0046] <표 1. 이더리움의 속성과 특징>

[0048] 다음은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 스마트 컨트랙트에 대해 상세히 설명하도록 한다.

[0049] 스마트 컨트랙트(Smart Contract)는 최초의 블록체인 기반 비트코인 스크립트이다. 비트코인 트랜잭션에 원시 언어인 OPCODE로 스크립트를 작성해서 보내면 조건에 따라 자동으로 거래를 수행한다. 하지만, 비트코인 스크립트는 반복문을 사용할 수 없고, 비트코인 잔고 외의 다른 정보를 관리 할 수 없는 한계가 있다. 이는 블록체인의 특이한 구조 때문인데 비트코인 스크립트에서 반복문을 허용할 경우 만약 스크립트 조건 때문에 무한 루프가 발생할 경우 네트워크 전체가 멈출 수 있다. 사용자는 무한 루프를 통해 쉽게 DoS(Denial of Service) 공격을

할 수 있다. 이더리움은 이러한 비트코인 스크립팅 시스템의 한계를 극복하고자 나온 스마트 컨트랙트 특화 블록체인 플랫폼이다. 비트코인 스크립팅 시스템의 한계인 다양한 상태 저장과 반복문을 허용한 스마트 컨트랙트를 만들었다. 이더리움의 스마트 컨트랙트의 간략한 작성 설명은 도 1과 같으며, 이더리움 스마트 컨트랙트가 동작하는 과정은 도 2와 같다.

[0051] 다음은 본 발명에 따른 IoT 서버 플랫폼(Mobius)에 대해 상세히 설명하도록 한다.

[0052] 사물인터넷 플랫폼 개발 및 오픈 소스 제공을 목적으로 Ocean(Open alliance for IoT standard)이 발족하였다. 모비우스(Mobius)는 Ocean에서 오픈 소스로 제공해주며, oneM2M 표준에 기반한 사물인터넷 플랫폼이다. 모비우스(Mobius)는 도 5와 같이 node.js의 express 모듈을 사용하여 개발되었으며, http, xml등의 모듈과 MQTT, CoAP 등 다양한 프로토콜을 지원한다. 계층적인 리소스 구조를 가지고 있으며, 리소스 저장을 위한 데이터베이스는 MySQL DBMS를 사용한다. 기본적인 기능 제공은 HTTP RESTful 방식을 따른다.

[0053] Mobius 플랫폼은 oneM2M 국제 표준을 기반으로 IoT(Internet of Things) 서

[0054] 비스 제공을 위해 다양한 IoT 디바이스 정보를 관리하고, 이들 IoT 디바이스의 접근 제어, 인증, 사용자 관리, 복수의 IoT 서비스 조합을 제공하여 애플리케이션을 통해 서비스하기 위한 플랫폼이다. Mobius 플랫폼은 IoT 디바이스 와 연동하기 위해 도 3과 같이 REST API(http, mqtt, coap, websocket)를 통해 연동된다.

[0055] Mobius는 node.js를 사용하여 JavaScript 언어로 개발된 Mobius 서버 플랫폼으로 국제 표준인 oneM2M 표준에 따라 개발되었다. Mobius 개발은 node.js의 express framework을 사용하지 않고 express 모듈을 사용하여 개발되었고, http, mqtt, express 등 다양한 node.js 모듈을 사용하고 있다. Mobius Platform은 oneM2M 표준을 준수하여 http, mqtt, coap, websocket 프로토콜과 Open API를 제공한다. oneM2M 표준에 따라 Mobius의 데이터 구조는 계층적인 리소스 구조를 가지고 있으며, 기본적으로 REST API 방식을 제공한다. Mobius Platform은 http Open API를 위한 http 서버와 mqtt 지원을 위한 mqtt 서버, coap을 위한 coap 서버, websocket 지원을 위한 서버로 구성되어 있으며, 리소스 저장을 위한 데이터베이스는 MySQL 데이터베이스MS를 사용한다. Mobius Platform의 구성도는 도 4와 같다.

[0057] 다음은 사물인터넷 환경에서 데이터, 인증, 네트워크 등 여러 보안 요소들이 필요하므로, 사물인터넷 환경의 여러 측면을 보완하기 위해 블록체인 기반의 IoT 서버 플랫폼 적용하기 위한 방법에 대해서 상세히 설명하도록 한다.

[0059] SecurePI 모니터링 관제 시스템을 적용할 수 있다. Secure PI 모니터링 관제 시스템은 도 6과 같은 Secure PI의 보안적 요소가 공격자로부터 공격을 당하거나 디바이스의 상태를 모니터링 하는 플랫폼이다.

[0061] SCC-Client는 수집한 Secure Pi의 데이터를 SCC-Server에 안전하게 전달하기 위해 SSL(Secure Sockets Layer)을 사용했고, SCC-Server는 SFTP(SSH File Transfer Protocol)를 이용하여 Secure Pi의 펌웨어 업데이트를 진행한다.

[0062] SCC-Server가 파일 입출력을 통해 Database에 저장한 데이터는 SCC-Web이 Json(JavaScript Object Notation)과 AJAX(Asynchronous JavaScript and XML)을 사용하여 웹페이지의 형태로 나타낸다.

[0063] Secure Pi 모니터링 관제 시스템 연구는 IoT 디바이스의 보안 기능들을 모니터링하는 시스템이다. 이 시스템에서 데이터들을 저장하는 장소는 범용 데이터 베이스 서버에 저장이 된다. IoT 디바이스들의 모든 데이터들이 중앙체계인 서버 한 곳에 저장되는데, 외부에서 공격자가 IoT 서버 플랫폼에 공격을 하거나 침입시도를 한다면 모든 데이터의 무결성과 신뢰성이 보장되지 못한다.

[0065] 다음은 범용 데이터베이스 취약점 문제를 해결하기 위해 블록체인 기술을 IoT 서버 플랫폼에 도입하기 위해 스마트 컨트랙트 설계 방법과 새로운 IoT 서버 플랫폼을 제안하고, 공격자로부터 공격을 막고 IoT 데이터의 위변조를 막는 방법을 제안한다.

[0066] 블록체인 기술로 만들어진 가상화폐 중 하나인 이더리움은 스마트 컨트랙트를 개발하여 DApp(decentralized application)을 연결하여 서비스를 할 수 있다.

[0067] 이더리움을 IoT 서버 플랫폼에 도입한 아키텍처는 도 7과 같다.

[0069] IoT 디바이스와 사용자 APP을 연결하기 위해서는 IoT 디바이스 각각의 지갑주소와 사용자 각각의 지갑주소를 이더리움 네트워크에서 발급해야 한다. 발급된 지갑주소를 이더리움 네트워크에서 관리하기 때문에 지갑주소가 있는 IoT 디바이스 및 사용자만 데이터를 조회하고 저장할 수 있다. 도 8은 데이터 저장 상세 설계, 도 9는 데이

터 조회 상세 설계, 도 10은 디바이스 조작 상세 설계를 나타낸다.

[0070] 사용자가 자신의 지갑주소와 디바이스 상태 변경 값, Private key를 Signature(서명)를 한 것을 IoT 서버에세 전달한다. IoT 서버는 전달받은 디바이스 상태 값으로 IoT 디바이스 상태를 변경한 후, 이더리움 네트워크에 API Function을 호출하여 상태 값을 전달한다. 이더리움은 스마트 컨트랙트에 작성된 내용으로 동작하여 블록에 상태값을 저장한다.

[0072] 다음은 스마트 컨트랙트의 자료구조 설계에 대해 상세히 설명하도록 한다.

[0073] IoT 디바이스가 이더리움 네트워크에 데이터를 저장하려면, 스마트 컨트랙트의 Data Struct를 설계해야 한다. 설계한 내용은 도 11과 같다.

[0074] 각각의 사용자가 사용하는 IoT 디바이스가 다를 수 있고, 여러 가지 IoT 디바이스를 사용자가 있을 수 있게 때문에 사용자 별로 Data Struct를 관리해야 한다. 또한, IoT 디바이스 마다 데이터 종류도 다르고 사용 목적도 다르기 때문에 사용자 Struct안에 Data Array를 각각 생성하여 데이터를 따로 관리해야한다.

[0075] 그러므로 도 11과 같이 설계하면 사용자 및 디바이스 각각의 데이터를 관리할 수 있게 되며, 또한 사용자와 디바이스 간에 지갑주소가 다르면 저장 및 조회를 못하게 막을 수 있다.

[0077] 다음은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용하는 스마트 컨트랙트 기능을 설계하는 방법에 대해 상세히 설명하도록 한다.

[0078] 블록체인 기술로 개발된 가상화폐 중 하나인 이더리움이 데이터베이스 서버로 사용되기 위해서는 기존에 데이터베이스 기능을 수행할 수 있도록 설계해야 한다. 기존의 데이터베이스에서 사용되는 SQL문의 기능인 Insert, Select, Delete, Update가 있다. 하지만 블록체인 기술은 데이터를 위변조 할 수 없기 때문에 Delete, Update 기능은 할 수가 없고, Insert, Select 기능만 가능하다. 스마트 컨트랙트에 Insert, Select 기능을 구현한 내용은 도 12 내지 도 13과 같다.

[0080] 다음은 디바이스와 사용자 매핑 관리 설계에 대해 상세히 설명하도록 한다.

[0081] IoT 디바이스의 실시간으로 쏟아지는 데이터를 각각의 사용자 Data Struct에 저장되려면 IoT 서버 플랫폼에서 디바이스와 사용자의 지갑주소 정보를 가지고 있어서 매핑을 하여 이더리움에게 전달해야 한다. IoT 서버 플랫폼에서 디바이스와 사용자의 지갑주소를 저장하는 곳은 범용 데이터베이스를 사용하여 관리한다. 지갑주소는 Public Key이기 때문에 공격자나 외부에 노출이 되어도 되는 데이터이다. Private Key를 알 수 없기 때문에 Public Key 만으로는 데이터를 위변조 할 수 없다. IoT 서버 플랫폼에서 디바이스와 사용자 매핑 관리설계는 도 14와 같다.

[0083] 다음은 본 발명에 따른 블록체인 기반의 IoT플랫폼을 활용한 데이터 저장 방법에 적용할 수 있는 Ring Signatures기술과 Zero-Knowledge Proof에 대해 상세히 설명하도록 한다.

[0084] 먼저, 도 15를 참조하여 Ring Signatures기술에 대해 설명하자면, 프로토콜을 이용한 디지털 자산으로 기존의 비트코인에 추적을 막기 위한 기술로 특정 그룹 내에서 키가 섞이도록 하고 거래확인을 위해서는 개인키가 필요로 한다. 따라서 거래내용을 제3자가 확인하는 것은 매우 어렵게 된다. 비트코인이 받는 사람이 키를 통하여 거래를 확인 할 수 있는 방법과 반대의 개념으로 완전한 거래내역의 익명화가 가능하다. 전송 속도 또한 평균적으로 2분정도 소요되는 빠른 승인 속도를 갖고 있다.

[0086] 다음은 도 16을 참조하여 Zero-Knowledge Proof에 대해 설명하자면, Zero-Knowledge Proof은 어떤 정보를 발설하지 않고, 그 정보를 알고 있음을 증명하는 방법으로 블록체인에 도입된 Zero-Knowledge Proof의 개념은 가상화폐나 데이터의 보유정보나 거래정보를 외부에 노출 시키지 않고 거래 또는 작업증명을 할 수 있는 증명 방법으로 완전성, 건실성, 영지식성 세 가지의 성질을 만족시킨 증명 방식이다.

[0087] 증명자는 비밀 키를 통해 문을 열 수 있고 검증자는 비밀 주문을 모르지만 증명자가 맞다는 것을 검증하는 것으로 증명자가 A나 B로 비밀 문까지 들어가고 검증자는 증명자에게 A쪽으로 돌아오라고 한다. 여기서 증명자가 비밀 키를 아는 경우, 증명자가 맞을 경우 돌아올 확률은 100%다. 그러나 증명자가 아니라도 돌아올 수 있는 확률은 50%다. 이 과정을 n번 계속 할 경우 증명자가 아닐 경우, 돌아오지 못할 확률은 점점 커지게 된다. 이렇게 확률을 통해 검증하는 방법을 Zero-Knowledge Proof이라 한다. 즉, 블록체인에서 Zero-Knowledge Proof을 활용하면 검증자가 거래당사자의 정보, 거래내용을 알지 못해도 거래 당사자가 맞다는 것을 확인 할 수 있다.

[0089] Zero-Knowledge Proof 인증 프로토콜의 등록은 기존의 비밀번호를 서버에 직접 넘겨 해시하는 시스템과 달리 도

17과 같이 클라이언트에서 공개키를 생성하여 아이디와 함께 저장한다. 이는 기존 시스템의 해시 암호화를 깨는 크래킹을 막을 수 있다는 장점이 있다. 사용자가 로그인을 할 때 서버와의 간단한 통신을 통해 서버에 비밀번호를 보내지 않아도 증명이 가능한 프로토콜이다.

[0090] 본 발명에 적용하는 방법으로는 Zero-Knowledge Proof을 이용한 인증 프로토콜의 아이디어를 이용하여 스마트 미터기의 데이터를 검증자에게 공개하지 않고 블록에 올리는 방법이다.

[0091] 스마트 미터기의 데이터를 블록체인에 저장하여 데이터 위, 변조를 막는 시스템은 검증자나 제3자가 블록조회나 컨트랙트 함수 호출을 통해서 데이터를 검색하여 사용자의 데이터를 훔쳐볼 수 있는 위험이 있다.

[0092] 이에 도 18과 같이 Zero-Knowledge Proof 인증 프로토콜을 이용하여 데이터를 블록체인에 올리지 않고 공개키를 블록체인에 저장하고 원본 데이터는 서버 데이터베이스에 저장한다. 데이터 호출을 할 때에는 사용자에게 데이터 베이스의 데이터를 보여주기 전에 블록체인 스마트 컨트랙트에 저장한 공개키와 함께 Zero-Knowledge Proof 과정을 거쳐서 증명이 완료되면 호출을 하여 데이터의 위, 변조를 막고 사용자의 데이터도 블록체인에 직접 올리지 않아 개인정보를 보호 할 수 있게 된다.

[0094] 본 발명에서 제안 시스템은 클라이언트, 서버, 블록체인 세 가지 단으로 구성된다. 블록체인에는 스마트 컨트랙트를 이용해 등록 단계와 인증 단계를 각각 통신 부담이 적은 비 대화식 Zero-Knowledge Proof 함수로 구현하였다. 먼저 등록 단계에서는 클라이언트에서 보호 할 데이터를 입력 받고 데이터를 해시하여 공개키를 생성한다. 서버에서는 데이터베이스에 실제 전송 받은 원본 데이터를 저장한다. 블록체인에는 보호 할 원본 데이터는 전송하지 않고 데이터를 통해 생성 된 공개키를 전송하여 블록에 저장하여 등록을 완료한다. 제안된 시스템에서 데이터를 호출 할 때에는 인증 과정을 거치게 된다. 클라이언트가 서버를 통해 데이터베이스에 저장한 원본 데이터를 호출하게 되면 블록체인에 저장된 공개키를 통해 조회한다. 인증 과정의 서버는 증명자가 되어 전송받은 공개키를 통해 Zero-Knowledge Proof에 의하여 검증하게 된다. 이를 통하여 원본 데이터를 블록체인에 저장이 하지 않아도 위, 변조가 되지 않았다는 것을 증명할 수 있다.

[0096] 다음은 도 19를 참조하여 Zero-Knowledge Proof를 이용한 블록체인 시스템에서 데이터의 공개키를 생성하고 저장하는 과정에 대해 상세히 설명하도록 한다.

[0098] 디바이스에서 디바이스의 공개키와 Signature로 디바이스를 인증 할 수 있는 정보와 함께 원본 데이터가 서버로 전송된다. 서버는 전송된 원본데이터를 데이터베이스에 저장하고 키 생성 모듈을 통하여 데이터의 공개키를 생성한다. 공개키를 생성하는 방법은 이산대수를 이용한 방법을 이용하여 임의의 소수 g와 r을 생성한다. 생성한 g와 r을 통해 서버를 통해 전송받은 데이터를 해시하여 하기의 표 2와 같이 공개키 p를 생성한다. 생성된 공개키 p로 데이터를 구하는 것은 이산대수 문제에 의해 불가능 하다. g와 r, p는 모두 공개해도 되는 정보이므로 트랜잭션을 보내 블록을 생성하여 저장시킨다. 블록이 체인에 연결되면 디바이스에서 전송된 데이터의 공개키가 블록체인에 저장되고 이 공개키는 블록체인에 저장된 정보는 위, 변조될 수 없다는 무결성으로 절대 변하지 않는다.

Key	생성 방법
g,r	Random Prime Number
p	$g^{H(data)} \bmod r$

[0099]

[0100] <표 2. 데이터의 공개키 생성>

[0102] 다음은 도 20을 참조하여 Zero-Knowledge Proof를 이용한 블록체인 시스템에서 데이터를 인증하는 과정에 대해 상세히 설명하도록 한다.

[0103] 사용자나 관리자가 데이터를 호출하면 서버는 데이터베이스에 날짜나 인덱스와 같은 기본키로 데이터를 검색한다. 동시에 서버가 블록체인에 저장했던 데이터의 공개키를 역시 기본키로 검색하여 공개키를 응답받는다. 키 생성 모듈에 전송받은 데이터의 공개키와 데이터, 임의의 소수 값 t를 통해 R1과 w의 값을 하기의 표 3과 같이 생성하고 블록체인 스마트 컨트랙트에 전송한다. 스마트 컨트랙트에서는 전송받은 w와 공개키의 값으로 R2를 생성한다. 전송받은 R1의 값과 비교하여 일치하면 데이터가 위, 변조되지 않았음을 인증하고 일치하지 않으면 데이터가 위, 변조되었음을 알려준다.

Key	생성 방법
t	Random Prime Number
R1	$g^t mod r$
u	$H(p, R1)$
w	$t - u \times H(data)$
R2	$g^w p^u mod r$

[0104]

[0105] <표 3. 데이터 인증을 위한 Zero-Knowledge Proof>

[0107] 본 발명의 제안은 블록체인에 데이터를 직접 저장하는 방식이 아닌 데이터의 공개키를 생성하고 블록체인에 저장하여 Zero-Knowledge Proof를 통한 검증 방법으로 데이터와 데이터의 공개키의 인증을 통해 데이터의 무결성을 확인한다. 데이터 하나의 무결성은 검증을 통해 쉽게 알 수 있지만 일정기간의 혹은 다량의 데이터를 한 번에 검색하여 검증을 하게 되면 데이터를 하나 씩 블록체인의 데이터의 공개키와 검증을 해야하기 때문에 성능에 문제가 발생하게 된다. 이에 데이터 저장 시에 실시간 데이터의 저장 뿐만 아니라 스마트 컨트랙트를 통해 일정기간의 데이터 누적량 또한 블록체인에 데이터누적량의 공개키로 저장이 되어야한다. 데이터의 검증은 제안한 방식과 같이 일정기간의 검색하고 싶은 데이터의 총량을 구해 블록체인의 스마트 컨트랙트를 통해 저장된 일정기간의 데이터 누적량 공개키와 Zero-Knowledge Proof를 하여 인증을 하게 된다. 만약 데이터의 위, 변조가 발생하게 되면 그 기간 동안 데이터를 일일이 검증하여 위, 변조된 데이터를 찾아 낼 수 있다.

[0108] 본 발명에서는 데이터의 검색 속도 향상을 위해 데이터 공개키의 값을 블록체인 사용자주소로 배열을 만들고 날짜를 배열의 인덱스로 만들어 검증할 때에 배열의 인덱스로 검색을 하여 일정 기간의 다량의 데이터를 배열로 검색해서 수행시간을 단축 시켰다. 도 23, 도24는 본 발명에 따른 실시 예로 기존 설계하였던 데이터 공개키 검색 방법에서 다량의 데이터 검색을 위해 개선한 데이터 검색 과정과 스마트 컨트랙트 소스이다.

[0110] 본 발명은 IoT 데이터를 블록체인을 활용하여 데이터 위, 변조를 막는 시스템을 제안하였고, 데이터 보호를 위한 Zero-knowledge proof 기술을 적용한 IoT 시스템 환경을 제안하였다.

[0111] 본 발명에서 제안한 시스템은 먼저 IoT 데이터를 블록체인에 저장하여 IoT 디바이스 인증 및 데이터 위, 변조를 막을 수 있다는 장점과 Zero-knowledge proof 기술을 적용하여 제3자가 블록 검색을 통해 사용자의 데이터 사용을 확인하지 못하게 하는 개인정보를 보호하는 장점이 있다. 스마트 미터기를 통해 전력량을 측정하고 요금을 납부하는 현재의 시스템은 데이터의 위조 및 변조, 요금을 계산하는 과정에서의 오류 등의 여러 가지 문제점이 있기 때문에 IoT 환경에서 블록체인을 적용하여 이 문제를 해결하였고 더 나아가 프로슈머 전력거래, 전기 자동차 충전기에도 Zero-knowledge proof이 프리컴파일 된 스마트 컨트랙트를 통해서 거래를 편리하고 안전하게 할 수 있다.

[0113] 이상 본 발명의 실시 예에 따른 도면을 참조하여 설명하였지만, 본 발명이 속한 분야에서 통상의 지식을 가진 자라면 상기 내용을 바탕으로 본 발명의 범주 내에서 다양한 응용 및 변형을 행하는 것이 가능할 것이다.

부호의 설명

도면

도면1

```
pragma solidity ^0.4.0;

contract SimpleStorage {
  uint storedData;

  function set(uint x) {
    storedData = x;
  }

  function get() returns (uint retVal){
    return storedData;
  }
}
```

컴파일러 버전

스마트 계약 이름

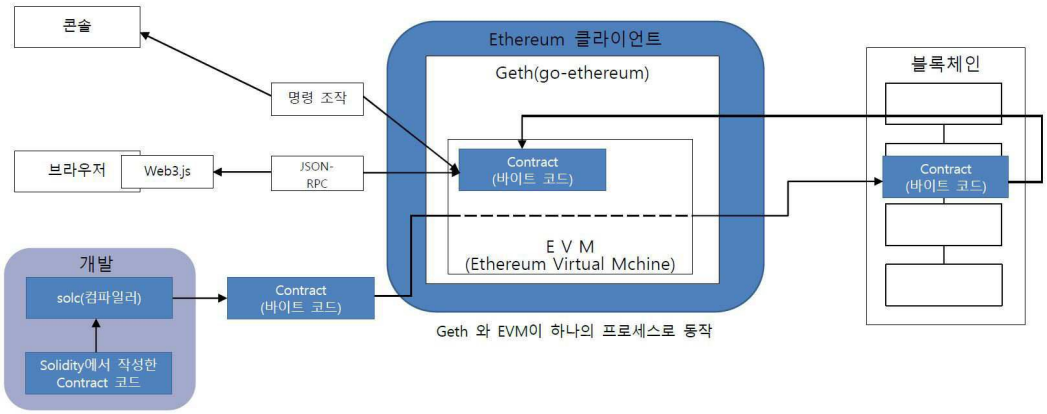
멤버변수와 타입

함수 선언 : 함수명(인자)

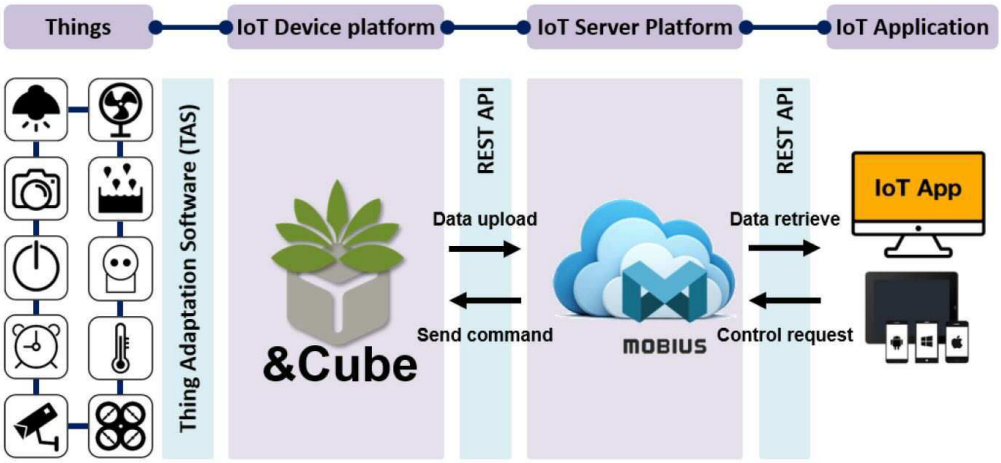
함수 내부

함수 선언 : 반환값이 있는 경우

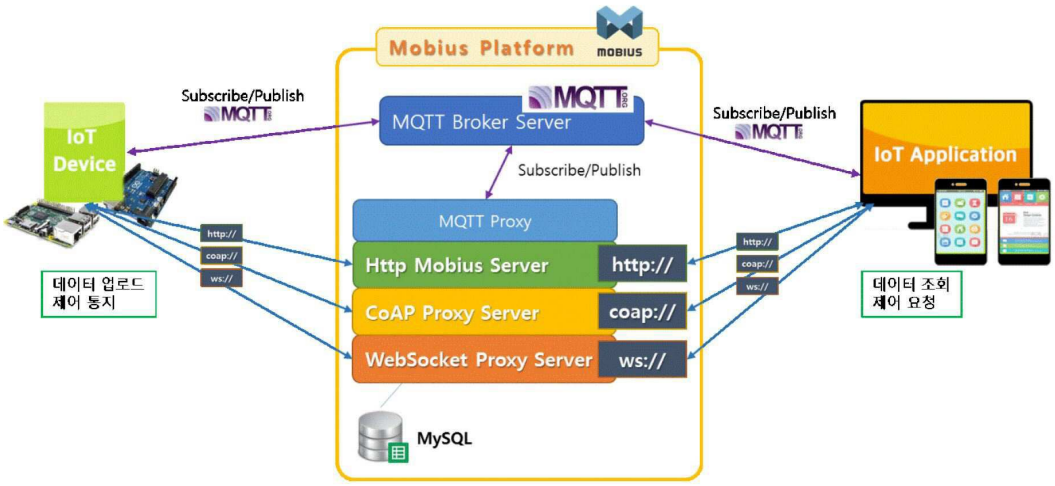
도면2



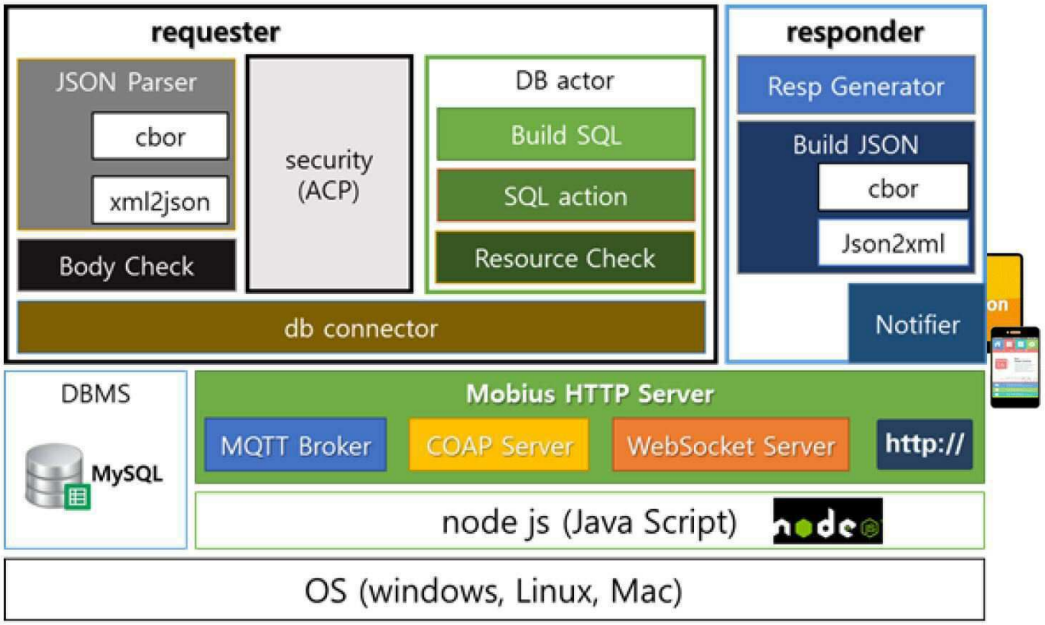
도면3



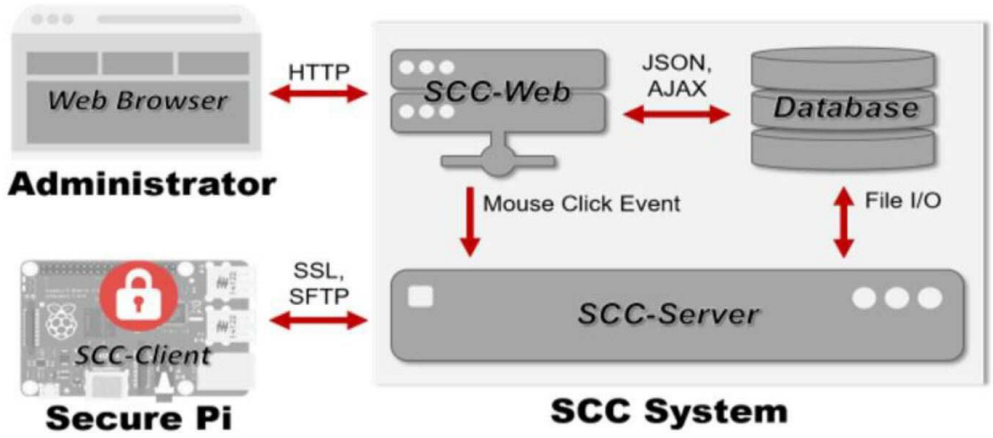
도면4


Mobius Platform
MOBIUS
Subscribe/Publish
MQTT
MQTT Broker Server
Subscribe/Publish
IoT Device
Subscribe/Publish
MQTT
IoT Application
MQTT Proxy
http://
coap://
ws://
Http Mobius Server
http://
CoAP Proxy Server
coap://
WebSocket Proxy Server
ws://
MySQL
데이터 업로드
제어 통지
데이터 조회
제어 요청

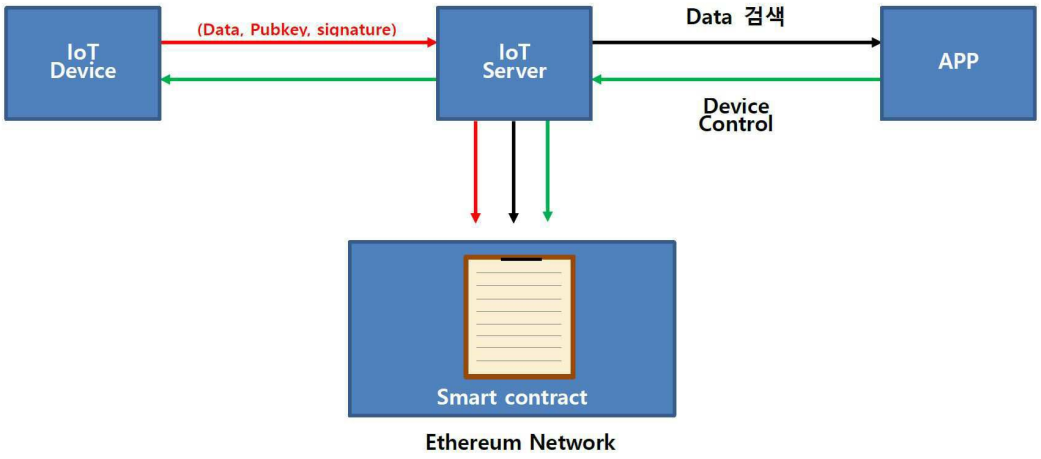
도면5


requester
JSON Parser
cbor
xml2json
Body Check
security
(ACP)
DB actor
Build SQL
SQL action
Resource Check
db connector
responder
Resp Generator
Build JSON
cbor
Json2xml
Notifier
DBMS
MySQL
Mobius HTTP Server
MQTT Broker
COAP Server
WebSocket Server
http://
node js (Java Script)
OS (windows, Linux, Mac)

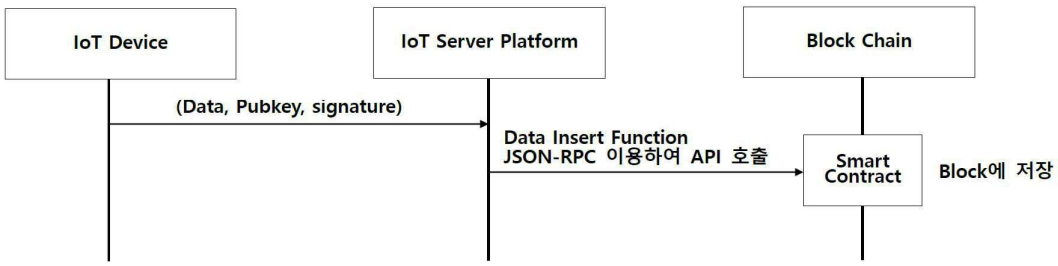
도면6


Web Browser
HTTP
SCC-Web
JSON,
AJAX
Database
Administrator
Mouse Click Event
File I/O
SSL,
SFTP
SCC-Client
SCC-Server
Secure Pi
SCC System

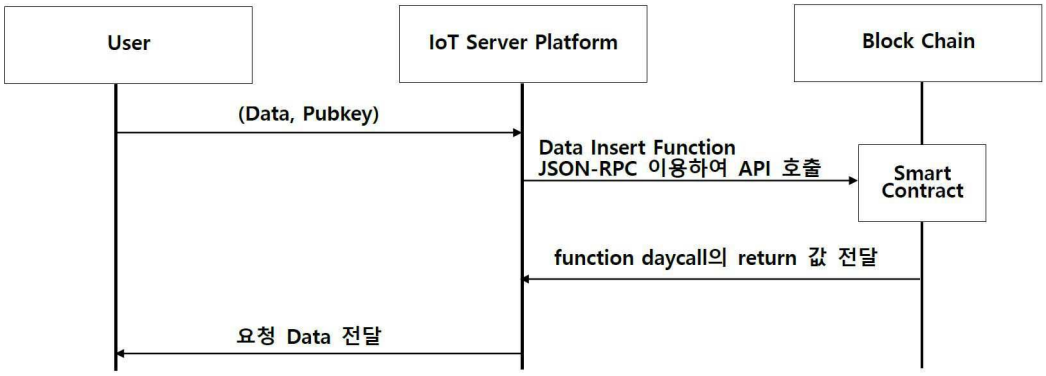
도면7


IoT
Device
(Data, Pubkey, signature)
IoT
Server
Data 검색
APP
Device
Control
Smart contract
Ethereum Network

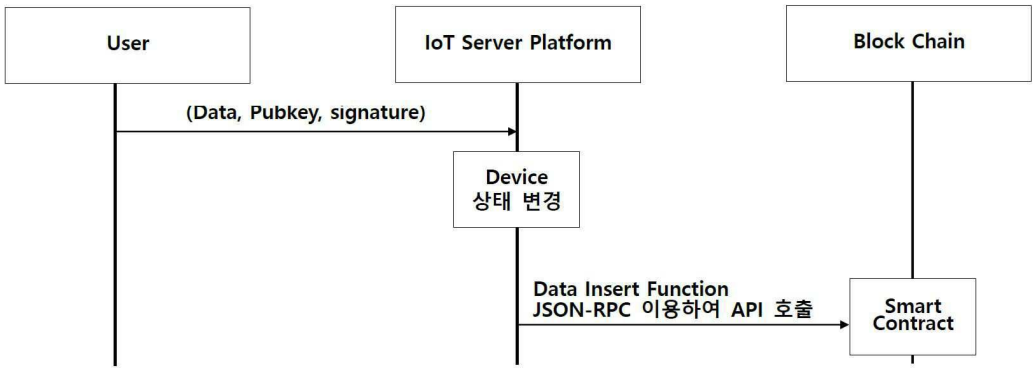
도면8


IoT Device
IoT Server Platform
Block Chain
(Data, Pubkey, signature)
Data Insert Function
JSON-RPC 이용하여 API 호출
Smart
Contract
Block에 저장

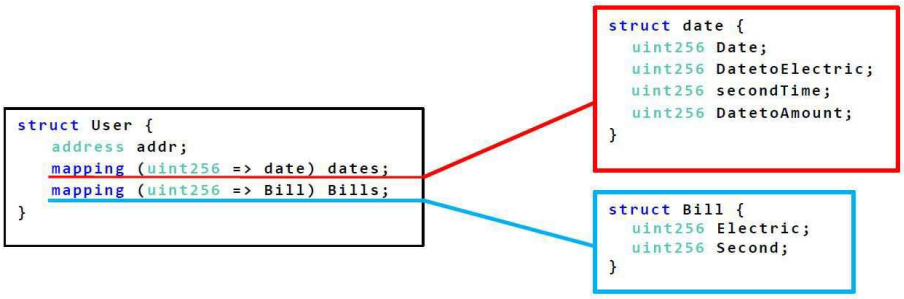
도면9



도면10



도면11



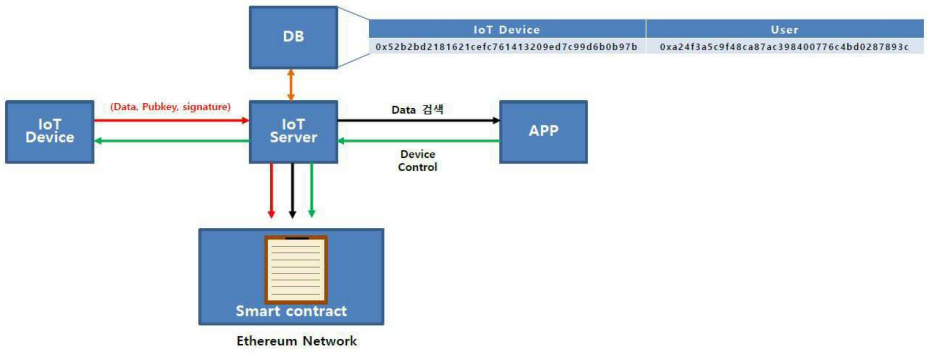
도면12

```
function datasender(address user, uint256_UsertoElectric, uint256_date, uint256_second, uint256_billdate) {
   User u = Users[user];

   u.addr = user;
   u.dates[_date].date = _date;
   u.dates[_date].datetoElectric += _UsertoElectric;
   u.dates[_date].SECONDtIME += _second;

   u.Bills[_billdate].Electric += _UsertoElectric;
   u.Bills[_billdate].Second += _second;
}
```

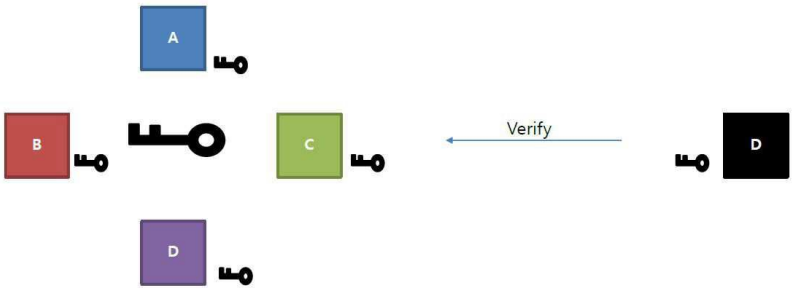
도면13

```
function datacall(address user, uint256 _data) returns (uint256 _searchdate, uint256 _searchamount, uint256 _searchsecond){
    User u = Users[user];

    _searchdate = u.dates[_date].Date;
    _searchamount = u.dates[_date].DatetoElectric;
    _searchsecond = u.dates[_date].SecondTime;
}

function monthcall(address user, uint256 _data) returns (uint256 _searchamount, uint256 _searchsecond){
    User u = Users[user];

    _searchamount = u.Bills[_date].Electric;
    _searchsecond = u.Bills[_date].Second;
}

function ElecBillmonthcheck(address user, uint256 _billdate) payble returns (uint256 bill, uint256 result, bool result2){
    User u = Users[user];

    uint256 electric = u.Bills[_billdate].Electric;
    uint256 second = u.Bills[_billdate].Second;

    result = ((((electric / 10000) * 220) / 1000) * (second / 3600000)) / 1000;

        if (result < 10) {
            bill = 100000000000000000;

            // if (!Owner.send(bill)) {
            //     Owner.transfer(bill);
            // }

            result2 = true;

     } else if (result > 9 && result < 101){
        bill = ((result * 60) / 1000) * 1000000000000000000;

        // if (!Owner.send(bill)) {
        //     Owner.transfer(bill);
        // }

        result2 = true;
    }
```

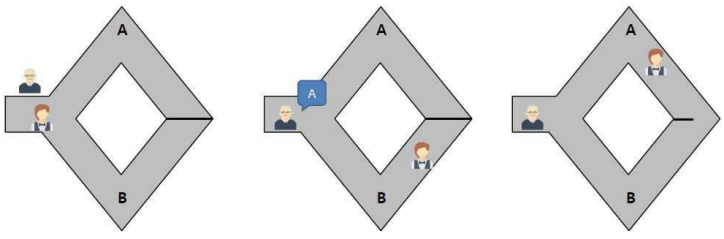
도면14



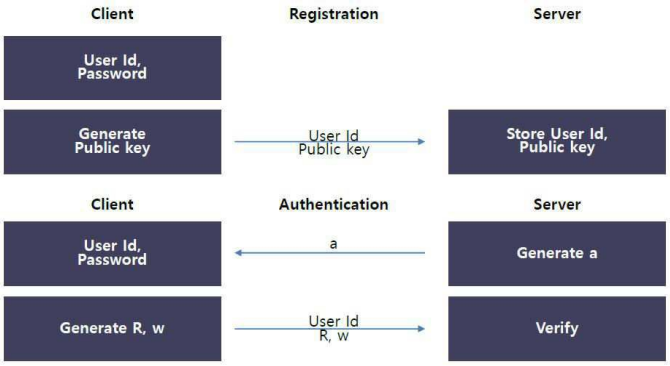
도면15



도면16

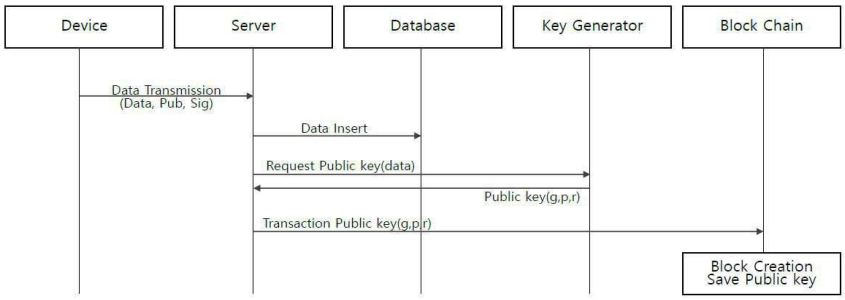


도면17

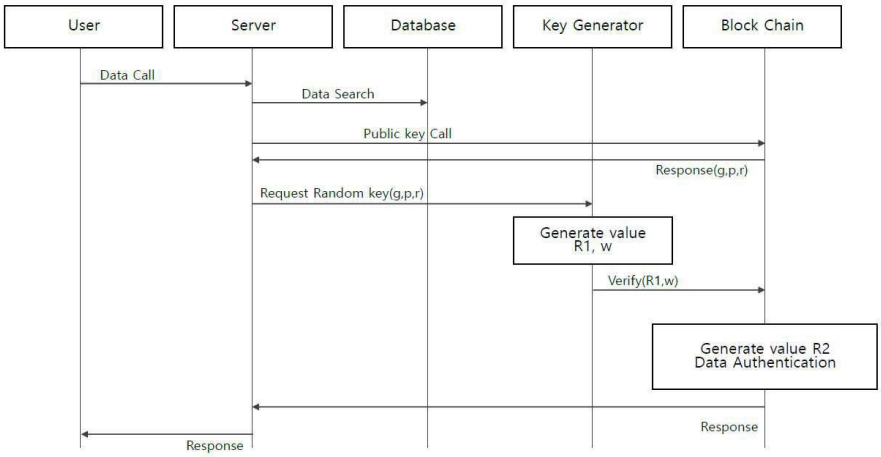


도면18

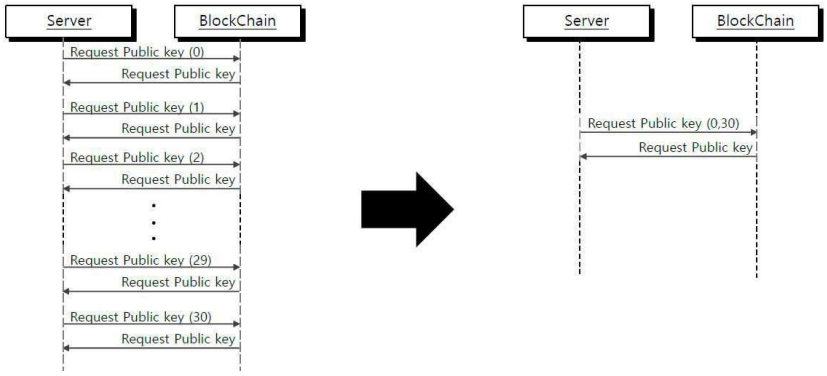
도면19



도면20



도면21



도면22

```
function registration(address user, uint256 id, uint256 g, uint256 r, uint256 y){
    User u =Users[user];
    u.addr = user;
    u.mems[id].g0 = g;
    u.mems[id].r0 = r;
    u.mems[id].y0 = y;

}

function PublicKey(address user, uint256 id) constant returns (uint256 g, uint256 r, uint256 y){
    User u = Users[user];


    g = u.mems[id].g0;
    r = u.mems[id].r0;
    y = u.mems[id].y0;

}

function PublickeyArray(address user, uint256 id1, uint256 id2) constant returns (uint256[30] g, uint256[30] r, uint256[30]p){
    User u = Users[user];
    uint256 j = 0;
    for(uint256 I = id1;i<=id2;i++){
        g[j]=u.mems[i].g0;
        r[j]=u.mems[i].r0;
        p[j]=u.mems[i].y0;
    }

}
```