



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2014년04월30일
(11) 등록번호 10-1390252
(24) 등록일자 2014년04월23일

(51) 국제특허분류(Int. Cl.)
G06F 21/62 (2013.01)
(21) 출원번호 10-2013-0041114
(22) 출원일자 2013년04월15일
심사청구일자 2013년04월15일
(56) 선행기술조사문헌
KR1020090009248 A
KR1020110071687 A
US20120173964 A1
US20120284633 A1

(73) 특허권자
인하대학교 산학협력단
인천광역시 남구 인하로 100, 인하대학교 (용현동)
(72) 발명자
김기창
인천 연수구 먼우금로 126, 207동 404호 (동춘동, 현대대림2차아파트)
유상봉
경기 부천시 원미구 조마루로 84, 2608동 404호 (상동, 하얀마을아이파크)
(74) 대리인
이원희

전체 청구항 수 : 총 10 항

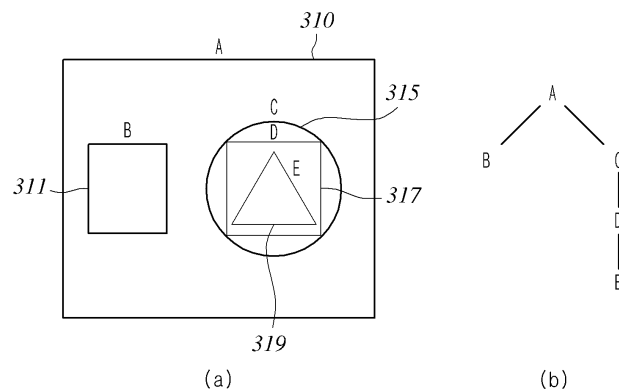
심사관 : 구본재

(54) 발명의 명칭 안전한 협업 디자인 작업을 위한 다중 레이어 암호화 파일 생성 장치 및 방법

(57) 요약

안전한 협업 디자인 작업을 위한 다중 레이어 암호화 파일 생성 장치 및 방법을 개시한다. 적어도 하나의 객체를 포함하는 디자인 파일을 저장하는 저장부 및 적어도 하나의 객체 중 하나 또는 하나 이상의 객체를 포함하는 레이어의 계층적 구조에 따라 트리를 생성하고, 생성된 트리에 포함된 객체를 서로 다른 키로 암호화하며, 레이어의 위치정보 및 적어도 하나의 키에 관한 정보를 저장하는 헤더를 생성하고, 생성된 헤더 및 상기 트리의 계층적 구조를 갖는 다중 레이어 암호화(MLE: Multiple layer Encryption) 파일을 생성하는 제어부를 포함하는 것을 특징으로 한다.

대표도



이 발명을 지원한 국가연구개발사업

과제고유번호 45394-1

부처명 한국연구재단

연구사업명 기본연구

연구과제명 대용량 DDoS 공격 진원지의 실시간 역추적 기술 개발

기 여 율 1/1

주관기관 인하대학교 산학협력단

연구기간 2012.05.01 ~ 2013.04.30

특허청구의 범위

청구항 1

적어도 하나의 객체를 포함하는 디자인 파일을 저장하는 저장부; 및

상기 적어도 하나의 객체 중 하나 또는 하나 이상의 객체를 포함하는 레이어의 계층적 구조에 따라 트리를 생성하고, 상기 생성된 트리에 포함된 객체를 서로 다른 키로 암호화하며, 상기 레이어의 위치정보 및 상기 적어도 하나의 키에 관한 정보를 저장하는 헤더를 생성하고, 상기 생성된 헤더 및 상기 트리의 계층적 구조를 갖는 다중 레이어 암호화(MLE: Multiple layer Encryption) 파일을 생성하는 제어부를 포함하는 것을 특징으로 하는 안전한 협업 디자인 작업을 위한 다중 레이어 암호화 파일 생성 장치.

청구항 2

제 1항에 있어서,

상기 제어부는,

상기 적어도 하나의 객체 중 하나 또는 하나 이상의 객체를 포함하는 레이어의 계층적 구조에 따라 상기 트리를 생성하는 트리구조 생성부;

상기 생성된 트리에 포함된 객체를 상기 서로 다른 키로 암호화하는 레이어 암호화부;

상기 레이어의 위치정보 및 상기 적어도 하나의 키에 관한 정보를 저장하는 헤더를 생성하는 헤더 생성부; 및

상기 트리의 계층적 구조 및 상기 헤더의 정보를 갖는 MLE 파일을 생성하는 다중 레이어 암호화 파일 생성부를 포함하는 것을 특징으로 하는 안전한 협업 디자인 작업을 위한 다중 레이어 암호화 파일 생성 장치.

청구항 3

제 1항에 있어서,

상기 디자인 파일은 계층적으로 구성된 상기 객체들의 집합인 것을 특징으로 하는 안전한 협업 디자인 작업을 위한 다중 레이어 암호화 파일 생성 장치.

청구항 4

제 2항에 있어서,

상기 생성된 트리에 포함된 노드는 상기 레이어에 포함된 객체를 포함하는 것을 특징으로 하는 안전한 협업 디자인 작업을 위한 다중 레이어 암호화 파일 생성 장치.

청구항 5

제 4항에 있어서,

상기 객체는 가장 외곽에 위치하는 객체인 상위 객체를 포함하는 것을 특징으로 하는 안전한 협업 디자인 작업을 위한 다중 레이어 암호화 파일 생성 장치.

청구항 6

제 2항에 있어서,

상기 트리의 에지는 계층적 구조를 보여주고, 상기 객체간의 부모 자식 관계를 하는 것을 특징으로 하는 안전한 협업 디자인 작업을 위한 다중 레이어 암호화 파일 생성 장치.

청구항 7

제 2항에 있어서,

상기 레이어 암호화부는 상기 트리의 각 레이어별로 관심 지역(ROI: Regions Of Interest)을 설정하고, 상기 ROI들은 상기 서로 다른 키로 암호화하여 접근 권한을 부여하는 것을 특징으로 하는 안전한 협업 디자인 작업을 위한 다중 레이어 암호화 파일 생성 장치.

청구항 8

제 1항에 있어서,

상기 MLE 파일은,

상기 레이어별 위치정보 및 상기 레이어별 암호화된 키를 저장하는 레이어 노드 테이블(LNTAB: Layer Node Table);

상기 MLE 파일에 대한 접근 권한이 있는 등록 사용자에게 대한 정보를 저장하는 사용자 노드 테이블(UNTAB: User Node Table); 및

상기 키 및 상기 키가 사용되는 타겟 노드의 인덱스 값을 저장하는 암호화 세션 키 노드 테이블(ESKNTAB: Encrypted Session Key Node Table)를 포함하는 것을 특징으로 하는 안전한 협업 디자인 작업을 위한 다중 레이어 암호화 파일 생성 장치.

청구항 9

제 8항에 있어서,

상기 레이어 노드 테이블은 상기 MLE 파일의 익스텐트 3 내지 12에 할당되고, 상기 사용자 노드 테이블은 상기 MLE 파일의 익스텐트 14 내지 23에 할당되며, 상기 암호화 세션 키 노드 테이블은 상기 MLE 파일의 익스텐트 25 내지 34에 할당되는 것을 특징으로 하는 안전한 협업 디자인 작업을 위한 다중 레이어 암호화 파일 생성 장치.

청구항 10

디자인 파일을 접속하는 단계;

상기 디자인 파일에서 적어도 하나의 객체 중 하나 또는 하나 이상의 객체를 포함하는 레이어의 계층적 구조에 따라 트리를 생성하는 단계;

상기 생성된 트리에 포함된 객체를 서로 다른 키로 암호화하는 단계;

상기 레이어의 위치정보 및 상기 적어도 하나의 키에 관한 정보를 저장하는 헤더를 생성하는 단계; 및

상기 트리의 계층적 구조 및 상기 헤더의 정보를 갖는 다중 레이어 암호화(MLE: Multiple layer Encryption) 파일을 생성하는 단계를 포함하는 것을 특징으로 하는 안전한 협업 디자인 작업을 위한 다중 레이어 암호화 파일 생성 방법.

명세서

기술분야

[0001] 본 발명은 안전한 협업 디자인 작업을 위한 다중 레이어 암호화 파일 생성 장치 및 방법에 관한 것으로서, 보다 상세하게는 공동으로 작업하는 디자인의 보안을 위해 다중 레이어 암호화(MLE: Multiple Layer Encryption) 파일을 생성하는 디자인 작업을 위한 다중 레이어 암호화 파일 생성 장치 및 방법에 관한 것이다.

배경 기술

[0002] 현대 디자인은 작업을 하는데 있어서 협업 디자인을 해야 하는 상황이 빈번히 발생한다. 협업 디자인은 수많은 디자이너들이 동일한 파일에서 작업을 해야 하는 경우가 발생할 수 있다. 이 경우 디자이너들은 원활하게 자신의 영역 안에서 작업할 수 있는 접근 제어 장치가 요구된다. 이 문제에 대한 전통적인 해결 방안은 역할기반 접근 제어(RBAC: Role-Based Access Control)이다. RBAC에서는 각 레이어에 롤(role)을 부여하고, 각 사용자에게 하나 혹은 복수개의 롤을 역시 부여한다. 각 사용자는 자신의 롤이 허용하는 레이어만 접근할 수 있다. RBAC은 그 동안 여러 협업 디자인 문제에 성공적으로 적용되었으며 광범위하게 연구되어 왔다. 하지만 디자인 스페이스의 크기와 복잡도가 증가하면서 "롤 폭발"이라고 불리는 확장성 문제를 노출하게 되었다. 특히 클라우드 시스템 같은 데이터 공유 환경에서 암호화 없는 접근 제어는 언제나 정보노출의 문제점을 지니고 있다는 것이 지적되고 있다.

[0003] 이와 같은 이유로 파일 접근 제어 목적으로 파일 암호화 해결방안이 제안되어 왔다. "Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers," Computers & Security 30 (2011) 320-331, 2011 에서 G. Wang, Q. Liu, J. Wu, M. Guo 은 사용자의 속성에 기반한 키 조합으로 파일을 암호화하고 키 조합에 참여한 사용자에게만 파일 접근을 허용하는 기법을 제안하였다.

[0004] 또한 M. A. Halcrow의 "eCryptfs: An Enterprise-class Cryptographic Filesystem for Linux,"에서 산업 암호 파일시스템(eCryptfs: Enterprise Cryptographic Filesystem)은 각 파일을 임의의 키로 암호화하고, 암호화된 키를 파일 헤더에 다시 한번 암호화된 형태로 기록하는 파일 시스템을 제안하고 있다. 이 경우 기록된 파일을 접근할 수 있는 사용자의 공개키로 암호화된 키를 다시 암호화하므로 (공개키 시스템을 사용하는 경우), 이 임의의 키를 복호화할 수 있는 사용자에게만 파일 접근을 허용하는 효과를 거둘 수 있다. 암호화된 키를 포함하여 파일을 접근하는데 필요한 모든 정보가 파일 헤더에 포함되어 있으므로 더 이상의 중앙 관리자의 필요 없이 파일을 자유롭게 그리고 안전하게 주고 받을 수 있다.

[0005] 하지만 여전히 여러 사용자가 각각 파일의 해당 부분만을 접근하도록 하는 다중 사용자의 동시 파일 접근을 허용하지는 못한다. 즉, eCryptfs는 합법적인 사용자만이 동일 파일을 전체적으로 접근할 수 있도록 하지만, 각 사용자가 자신에게 허용된 파일의 서로 다른 부분만을 접근하도록 하는 기능은 제공하지 못하고 있다.

발명의 내용

해결하려는 과제

[0006] 본 발명이 이루고자 하는 기술적인 과제는 다중 사용자가 안전하게 동일한 디자인 파일에 대해 협업할 수 있는 환경을 제공하는데 목적이 있다.

[0007] 또한, 협업 디자인이 복잡해져도 접근 제어의 유지비용이 증가하지 않는 것을 제공하는데 목적이 있다.

과제의 해결 수단

[0008] 적어도 하나의 객체를 포함하는 디자인 파일을 저장하는 저장부 및

[0009] 상기 적어도 하나의 객체 중 하나 또는 하나 이상의 객체를 포함하는 레이어의 계층적 구조에 따라 트리를 생성하고, 상기 생성된 트리에 포함된 객체를 서로 다른 키로 암호화하며, 상기 레이어의 위치정보 및 상기 적어도 하나의 키에 관한 정보를 저장하는 헤더를 생성하고, 상기 생성된 헤더 및 상기 트리의 계층적 구조를 갖는 다중 레이어 암호화(MLE: Multiple layer Encryption) 파일을 생성하는 제어부를 포함할 수 있다.

[0010] 상기 제어부는 상기 적어도 하나의 객체 중 하나 또는 하나 이상의 객체를 포함하는 레이어의 계층적 구조에 따

라 상기 트리를 생성하는 트리구조 생성부, 상기 생성된 트리에 포함된 객체를 상기 서로 다른 키로 암호화하는 레이어 암호화부, 상기 레이어의 위치정보 및 상기 적어도 하나의 키에 관한 정보를 저장하는 헤더를 생성하는 헤더 생성부 및 상기 트리의 계층적 구조 및 상기 헤더의 정보를 갖는 MLE 파일을 생성하는 다중 레이어 암호화 파일 생성부를 포함할 수 있다.

- [0011] 상기 디자인 파일은 계층적으로 구성된 상기 객체들의 집합일 수 있다.
- [0012] 상기 생성된 트리에 포함된 노드는 상기 레이어에 포함된 객체를 포함할 수 있다.
- [0013] 상기 객체는 가장 외곽에 위치하는 객체인 상위 객체를 포함할 수 있다.
- [0014] 상기 트리의 예지는 계층적 구조를 보여주고, 상기 객체간의 부모 자식 관계를 할 수 있다.
- [0015] 상기 레이어 암호화부는 상기 트리의 각 레이어별로 관심 지역(ROI: Regions Of Interest)을 설정하고, 상기 ROI들은 상기 서로 다른 키로 암호화하여 접근 권한을 부여할 수 있다.
- [0016] 상기 MLE 파일은 상기 레이어별 위치정보 및 상기 레이어별 암호화된 키를 저장하는 레이어 노드 테이블(LNTAB: Layer Node Table), 상기 MLE 파일에 대한 접근 권한이 있는 등록 사용자에게 대한 정보를 저장하는 사용자 노드 테이블(UNTAB: User Node Table) 및 상기 키 및 상기 키가 사용되는 타겟 노드의 인덱스 값을 저장하는 암호화 세션 키 노드 테이블(ESKNTAB: Encrypted Session Key Node Table)를 포함할 수 있다.
- [0017] 상기 레이어 노드 테이블은 상기 MLE 파일의 익스텐트 3 내지 12에 할당되고, 상기 사용자 노드 테이블은 상기 MLE 파일의 익스텐트 14 내지 23에 할당되며, 상기 암호화 세션 키 노드 테이블은 상기 MLE 파일의 익스텐트 25 내지 34에 할당될 수 있다.
- [0018] 디자인 파일을 접속하는 단계, 상기 디자인 파일에서 적어도 하나의 객체 중 하나 또는 하나 이상의 객체를 포함하는 레이어의 계층적 구조에 따라 트리를 생성하는 단계, 상기 생성된 트리에 포함된 객체를 서로 다른 키로 암호화하는 단계, 상기 레이어의 위치정보 및 상기 적어도 하나의 키에 관한 정보를 저장하는 헤더를 생성하는 단계 및 상기 트리의 계층적 구조 및 상기 헤더의 정보를 갖는 다중 레이어 암호화(MLE: Multiple layer Encryption) 파일을 생성하는 단계를 포함할 수 있다.

발명의 효과

- [0019] 본 발명을 따른 안전한 협업 디자인 작업을 위한 다중 레이어 암호화 파일 생성 장치 및 방법에 의하면, 다중 사용자가 안전하게 동일한 디자인 파일에 대해 협업할 수 있는 환경을 제공할 수 있다.
- [0020] 또한, 협업 디자인이 복잡해져도 접근 제어의 유지비용을 증가하지 않을 수 있다.

도면의 간단한 설명

- [0021] 도 1은 본 발명의 일 실시예에 따른 다중 레이어 암호화 파일 생성 장치를 도시한 블록도이다.
- 도 2는 본 발명의 일 실시예에 따른 다중 레이어 암호화 파일 생성 장치의 제어부를 도시한 블록도이다.
- 도 3(a) 및 (b)은 본 발명의 일 실시예에 따른 계층적 구조를 도시한 예시도이다.
- 도 4는 본 발명의 일 실시예에 따른 사전저의 트리를 도시한 예시도이다.
- 도 5는 본 발명의 일 실시예에 따른 레이어 노드 테이블을 도시한 예시도이다.
- 도 6(a), (b) 및 (c)은 본 발명의 일 실시예를 따른 디렉토리 익스텐트를 도시한 예시도이다.
- 도 7은 본 발명의 일 실시예에 따른 암호화 세션 키 노드 테이블을 도시한 예시도이다.
- 도 8(a), (b) 및 (c)은 본 발명의 일 실시예에 따른 각 사용자에게 제공 화면을 도시한 예시도이다.
- 도 9는 본 발명의 일 실시예에 따른 사용자 노드 테이블을 도시한 예시도이다.
- 도 10은 본 발명의 일 실시예에 따른 세 명의 사용자 등록 후의 암호화 세션 키 노드 테이블을 도시한 예시도이다.

도 11은 본 발명의 일 실시예에 따른 MLE 파일의 전체 구조를 도시한 예시도이다.

도 12는 본 발명의 일 실시예에 따른 서버와 사용자의 관계를 도시한 흐름도이다.

도 13은 본 발명의 일 실시예에 따른 다중 레이어 암호화 파일 생성 방법을 도시한 순서도이다.

발명을 실시하기 위한 구체적인 내용

- [0022] 이하 본 발명의 실시예를 첨부된 도면들을 참조하여 상세히 설명할 수 있다. 우선 각 도면의 구성요소들에 참조 부호를 부가함에 있어서, 동일한 구성요소들에 대해서는 비록 다른 도면상에 표시되더라도 가능한 한 동일한 부호를 가지도록 하고 있음에 유의해야 할 수 있다. 또한 본 발명을 설명함에 있어, 관련된 공지 구성 또는 기능에 대한 구체적인 설명이 당 업자에게 자명하거나 본 발명의 요지를 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명은 생략할 수 있다.
- [0023] 도 1은 본 발명의 일 실시예에 따른 다중 레이어 암호화 파일 생성 장치를 도시한 블록도이다.
- [0024] 도 1을 참조하면, 다중 레이어 암호화 파일 생성 장치(1)는 협업 디자인 작업을 할 때 디자인 파일을 레이어별로 접근 제어하는 장치일 수 있다. 다중 레이어 암호화 파일 생성 장치(1)는 입력부(110), 제어부(120) 및 저장부(130)를 포함할 수 있다. 다중 레이어 암호화 파일 생성 장치(1)은 단일 서버의 역할을 할 수 있다.
- [0025] 입력부(110)는 사용자 명령을 입력 받을 수 있다. 사용자 명령어는 패킷 형식일 수 있다. 또한 입력부(110)는 보안 소켓 계층(SSL: Secure Socket Layer) 프로토콜을 통해 암호화된 통신으로 사용자 명령을 입력 받을 수 있다. 입력부(110)는 수 많은 사용자를 동시에 서비스하기 위해 함수 "select()"의 시스템 콜을 사용할 수 있다.
- [0026] 제어부(120)는 입력부(110)에서 입력 받은 사용자 명령어에 따라 디자인 파일을 레이어별로 접근 제어를 할 수 있다. 디자인 파일은 계층적으로 구성된 레이어들의 집합일 수 있다. 제어부(120)는 레이어간의 경계를 트리로 표현하고 각 레이어를 서로 다른 키로 암호화할 수 있다. 또한 제어부(120)는 레이어들의 위치정보 및 레이어를 암호화하는데 사용된 적어도 하나의 키에 관한 정보를 파일 헤더에 저장하여 다중 레이어 암호화(MLE: Multiple Layer Encryption) 파일을 생성할 수 있다.
- [0027] 저장부(130)는 협업 디자인 작업에 필요한 정보를 저장할 수 있다. 저장부(130)은 적어도 하나의 객체를 포함하는 디자인 파일을 저장할 수 있다. 또한 저장부(130)는 키 암호 알고리즘, 세션 키의 정보, 사용자 정보 및 레이어의 위치정보 중 적어도 하나를 더 저장할 수 있다.
- [0028] 도 2는 본 발명의 일 실시예에 따른 다중 레이어 암호화 파일 생성 장치의 제어부를 도시한 블록도이다.
- [0029] 도 2를 참조하면, 제어부(120)는 디자인 파일을 레이어별로 접근 제어를 할 수 있다. 제어부(120)는 트리구조 생성부(121), 레이어 암호화부(122), 헤더 생성부(123) 및 다중 레이어 암호화 파일 생성부(124)를 포함할 수 있다.
- [0030] 트리구조 생성부(121)는 디자인 파일에서 적어도 하나의 객체 중 하나 또는 하나 이상의 객체를 포함하는 레이어의 계층적 구조에 따라 트리를 생성할 수 있다.
- [0031] 트리구조 생성부(121)는 계층 데이터 포맷(HDF: Hierarchical Data Format)을 이용할 수 있다. HDF는 데이터 셀을 계층적 구조로 파일에 저장하는 기술이다. 또한 데이터 셀은 유닉스 및 리눅스에서의 디렉토리 구조와 유사한 형태의 계층적 트리로 구성된다.
- [0032] 레이어 암호화부(122)는 트리에 포함된 객체를 서로 다른 키로 암호화할 수 있다. 레이어 암호화부(122)는 각 레이어별로 관심 지역(ROI: Regions of Interest)을 설정하고 각 ROI들을 서로 다른 키로 암호화하여 접근 권한을 부여할 수 있다. 예를 들어, 이미지 파일이 ROI1, ROI2 및 ROI3의 세부 레벨(LOD: Level of Detail) 순으로 존재한다고 할 때(ROI3이 가장 상세한 레벨), 레이어 암호화부(122)는 각 ROI를 상위 순서대로 키 (k1), (k1, k2) 및 (k1, k2, k3)으로 각각 암호화할 수 있다. 키 (k1, k2)는 (k1)으로 먼저 암호화하고, 그것을 다시 (k2)로 암호화하는 것을 의미한다. 키 (k1)을 보유한 사용자는 ROI1만 복호해서 이미지를 볼 수 있다. 또한 키 (k1, k2, k3)를 모두 보유한 사용자는 모든 ROI를 볼 수 있다. 레이어 암호화부(122)는 각 ROI를 서로 다른 키 조합으로 암호화함으로써 같은 이미지 파일이 여러 사용자에게 여러 레벨에서 보여지도록 할 수 있다.

- [0033] 헤더 생성부(123)는 레이어의 위치정보 및 암호화된 적어도 하나의 키에 관한 정보를 저장하는 헤더를 생성할 수 있다. 헤더 생성부(123)는 파일을 암호화하고 암호화된 키를 합법적 사용자의 공개키로 다시 암호화된 키를 저장하는 헤더를 생성할 수 있다.
- [0034] 다중 레이어 암호화 파일 생성부(124)는 트리의 계층적 구조 및 헤더의 정보를 갖는 MLE 파일을 생성할 수 있다. MLE 파일은 데이터 영역을 트리구조로 저장할 수 있다. 또한 MLE 파일은 메타 데이터 영역을 트리구조 및 트리의 각 노드에 접근할 수 있는 사용자 리스트로 저장할 수 있다. 메타 데이터 영역은 레이어 노드 테이블(LNTAB: Layer Node Table), 사용자 노드 테이블(UNTAB: User Node Table) 및 암호화 세션 키 노드 테이블(ESKNTAB: Encrypted Session Key Node Table)로 구성될 수 있다.
- [0035] 도 3(a) 및 (b)은 본 발명의 일 실시예에 따른 계층적 구조를 도시한 예시도이고, 도 4는 본 발명의 일 실시예에 따른 자전거의 트리를 도시한 예시도이며, 도 5는 본 발명의 일 실시예에 따른 레이어 노드 테이블을 도시한 예시도이고, 도 6(a), (b) 및 (c)은 본 발명의 일 실시예에 따른 디렉토리 익스텐트를 도시한 예시도이며, 도 7은 본 발명의 일 실시예에 따른 암호화 세션 키 노드 테이블을 도시한 예시도이다.
- [0036] 도 3 내지 도 7을 참조하면, 레이어 노드 테이블은 레이어 노드의 테이블로서, 각 레이어의 위치정보 및 세션 키가 저장될 수 있다. 세션 키는 각 레이어의 서로 다른 임의의 암호화한 키를 나타낸다.
- [0037] 도 3(a)은 디자인 파일의 객체를 표현하고 있다. 디자인 파일은 계층적으로 구성된 객체들의 집합으로 볼 수 있다. 디자인 파일은 객체(310), 객체(311), 객체(315), 객체(317) 및 객체(319)를 포함할 수 있다.
- [0038] 레이어는 상호간의 계층적 구조를 나타낼 수 있는 객체들의 집합일 수 있다. 또한 레이어 A는 객체(310), 객체(311), 객체(315), 객체(317) 및 객체(319)를 포함할 수 있고, 레이어 B는 객체(311)을 포함할 수 있다. 레이어 C는 객체(315), 객체(317) 및 객체(319)를 포함할 수 있고, 레이어 D는 객체(317) 및 객체(319)를 포함할 수 있으며, 레이어 E는 객체(319)를 포함할 수 있다.
- [0039] 여기서, 레이어 A의 객체(310), 레이어 B의 객체(311), 레이어 C의 객체(315), 레이어 D의 객체(317) 및 레이어 E의 객체(319)는 각 레이어들의 가장 외곽에 위치하는 객체일 수 있다. 가장 외곽에 위치하는 객체는 상위 객체라고 할 수 있고, 상위 객체는 각 레이어들의 계층적 구조의 최상위 객체일 수 있다. 레이어는 상위 객체를 통하여 레이어의 객체의 계층적 상호관계를 확인할 수 있다.
- [0040] 도 3(a)를 도 3(b)의 트리로 표현할 수 있다. 도 3(b)는 객체의 계층적 관계를 트리로 표현한 것으로써, 레이어 A는 모든 레이어를 포함한다. 레이어 C는 레이어 D와 레이어 E를 포함하고, 레이어 D는 레이어 E를 포함한다. 레이어 B와 레이어 E는 자기 자신만을 포함한다.
- [0041] 트리의 노드는 각 객체가 될 수 있으며, 예지는 객체간의 부모 자식 관계를 표현할 수 있다. 도 3(a)는 다섯 가지의 객체를 포함하고 있기 때문에 레이어 노드 테이블은 5개의 엘노드(lnode)를 가질 수 있다.
- [0042] lnode는 레이어 노드(layer node)를 의미하고, 각 객체가 계층적인 상호관계를 가지고 있으므로 "레이어"라는 표현을 사용될 수 있다. lnode는 유닉스 및 리눅스 파일 시스템에서의 inode와 유사할 수 있다. 다만 lnode는 해당 객체가 파일 공간의 어디에 물리적으로 존재하는지를 표현하는데 반해 inode는 해당 파일이 디스크 공간의 어느 블록에 물리적으로 존재하는지를 표현할 수 있다.
- [0043] 각 lnode는 64바이트 크기로서 다음의 구조를 지닐 수 있다.
- [0044] class LayerNode{
- [0045] // on-disk data
- [0046] int num_ext;
- [0047] int loc[MEXT];
- [0048] int loc_sub;
- [0049] int esk;

```
[0050]         unsigned char reserv[12];
[0051]         // on-memory data follows
[0052]         .....
[0053]     };
```

[0054] 변수 "num_ext"는 레이어가 소유한 데이터 익스텐트(혹은 블록)의 개수를 나타낸다. 익스텐트는 4096 바이트 블록이다. 각 레이어는 적어도 두 개의 익스텐트를 가질 수 있다. 하나는 레이어의 데이터를 포함하고 있는 데이터 익스텐트이고, 다른 하나는 레이어에 소속된 하위 레이어에 대한 정보를 담고 있는 디렉토리 익스텐트이다.

[0055] 배열 "loc[]"은 각 데이터 익스텐트의 위치를 담고 있고, MEXT는 10으로 정의할 수 있다. 배열 "loc[]"의 처음 8개의 방(배열 "loc[0]" 내지 배열 "loc[7]")은 위치정보에 대한 직접 포인터가 있고, 배열 "loc[8]"은 간접 포인터이며, 배열 "loc[9]"은 더블 간접 포인터가 들어 있을 수 있다.

[0056] 직접 포인터는 실제 해당 익스텐트의 위치를 직접 표시하는 포인터이다. 간접 포인터는 1024개의 직접 포인터가 저장되어 있는 익스텐트를 가르키는 포인터이다. 더블 간접 포인터는 1024개의 간접 포인터가 저장되어 있는 익스텐트를 가리키는 포인터이다. 위치정보는 직접 포인터, 간접 포인터 및 더블 간접 포인터를 이용하여 최대 $4096*8 + 1024*4096 + 1024*1024*4096$ 바이트(4G 바이트보다 조금 큰 크기) 크기의 파일에 대해 위치정보를 기록할 수 있다.

[0057] 변수 "loc_sub"는 하위 레이어에 대한 정보를 담고 있는 익스텐트이다. 만약 현재 레이어가 트리의 말단이라면 변수 "loc_sub"는 무의미하다. 하지만 현재 레이어가 트리의 말단이 아니라면, 변수 "loc_sub"는 하기 구조체 "dirent"(32 바이트 크기)의 리스트가 들어있는 익스텐트를 가리킨다.

```
[0058] struct dirent{
[0059]         char lname[MAX_NAME_LEN];
[0060]         int lnode_idx;
[0061]         char reserve[12];
[0062] }
```

[0063] 상기 구조체에서, 배열 "lname"은 하위 레이어의 이름을 말하고, 변수 "lnode_idx"는 하위 레이어가 레이어 노드 테이블에 기록되어 있는 인덱스를 말한다.

[0064] 변수 "esk"는 암호화 세션 키 노드에서 해당 하위 레이어의 세션 키 스케줄이 암호화되어 저장되어 있는 위치를 가르킬 수 있다.

[0065] 도 4는 자전거를 계층적 구조로 표현한 일 실시예이다. 최상위 계층은 자전거가 위치하고 자전거는 바퀴, 몸체 및 제어장치를 포함한다. 바퀴는 바퀴 프레임, 타이어 및 브레이크를 포함하고, 몸체는 핸들, 몸체 프레임 및 안장을 포함한다. 제어장치는 페달, 체인 및 기어를 포함한다. 도 4는 일 실시예를 보여주는 것이며, 본 발명의 디자인 협업에서도 객체를 계층적인 상호관계를 가지고 있는 레이어로 표현이 가능하다는 것을 보여주고 있다.

[0066] 도 5는 도 3(b)의 트리에 대한 레이어 노드 테이블을 나타내고 있다. LNODE 0는 레이어 A를 나타내고, LNODO 1은 레이어 A/B, LNODE 2는 레이어 A/C, LNODO 3는 레이어 A/C/D, LNODE 4는 레이어 A/C/D/E를 각각 나타낸다.

[0067] 도 3(b)의 트리가 각 레이어의 데이터 익스텐트를 모두 한 개라고 가정하면, 도 4의 레이어 노드 테이블은 모든 레이어의 변수 "num_ext"가 1이 된다. 또한 배열 "loc[0]"은 각 레이어의 데이터 익스텐트 위치를 나타낼 수 있다. 각 레이어의 디렉토리 익스텐트 위치는 도 4의 레이어 노드 테이블의 변수 "loc_sub" 열에 나와 있다. 레이어 B 및 레이어 E는 말단 레이어이므로 디렉토리 익스텐트에 아무 데이터가 없지만 레이어 A, 레이어 C 및 레이어 D는 서브 레이어를 가지고 있다.

[0068] 도 5는 레이어 A, 레이어 C 및 레이어 D의 디렉토리 익스텐트를 보여주고 있다. 도 5(a)는 레이어 A, 도 5(b)는

레이어 C 및 도 5(c)는 레이어 D의 디렉토리 익스텐트이다. 레이어 A는 두 개의 서브 레이어인 레이어 B 및 레이어 C를 가지고 있다. 따라서 도 5(a)은 레이어 B와 레이어 C의 LNODE 인덱스 값이 나타난다. 도 5(b) 및 도 5(c)는 레이어 C 및 레이어 D의 LNODE 인덱스 값을 각각 나타낸다.

- [0069] 도 4는 각 레이어에 대해 암호화된 세션 키(실제로는 세션 키 스케줄)가 ESKNTAB의 어디에 있는지가 변수 "esk" 열에 나와 있다. 예를 들면, 레이어 A(LNODE 0)는 변수 "esk" 값이 0이므로 도 6의 ESKNTAB의 인덱스 0에 있는 23 24 3a 4b ... 바이트 나열이 레이어 A의 암호화된 세션 키 스케줄이다.
- [0070] ESKNTAB 테이블은 암호화된 세션 키 노드(ESKNODE: Encrypted Session Key Node)들을 저장하고, 각 ESKNODE는 암호화된 세션 키와 세션 키가 사용되는 타겟 LNODE의 인덱스 값을 가진다. ESKNODE는 260바이트 크기이다.
- [0071] 도 8(a), (b) 및 (c)은 본 발명의 일 실시예에 따른 각 사용자에게 제공 화면을 도시한 예시도이고, 도 9는 본 발명의 일 실시예에 따른 사용자 노드 테이블을 도시한 예시도이며, 도 10은 본 발명의 일 실시예에 따른 세 명의 사용자 등록 후의 암호화 세션 키 노드 테이블을 도시한 예시도이다.
- [0072] 도 8 내지 도 10을 참조하면, 사용자 노드 테이블은 사용자 노드의 테이블이다.
- [0073] UNODE는 해당 MLE 파일에 대한 접근 권한이 등록 사용자들에 대한 정보를 표현하는 구조체이고 64 바이트의 크기를 가졌으며, 하기 구조를 가질 수 있다.
- [0074] `class UserNode{`
- [0075] `// on-disk data`
- [0076] `char uname[MAX_NAME_LEN];`
- [0077] `int home;`
- [0078] `int esk[MAX_ESK];`
- [0079] `unsigned char reserv[12];`
- [0080] `// on-memory data follows`
- [0081] `.....`
- [0082] `};`
- [0083] 배열 "uname[]"은 사용자의 이름을 나타낼 수 있다. 변수 "home"은 사용자의 홈 레이어 안에 있는 LNODE 인덱스 일 수 있다. 각 사용자는 홈 레이어가 정의되어 있어서 홈 레이어와 홈 레이어의 하위 트리에 있는 모든 레이어에 대한 접근 권한을 가질 수 있다.
- [0084] 도 8은 사용자 UA, 사용자 UC 및 사용자 UE가 제공받는 트리를 보여준다. 도 8(a)는 사용자 UA가 제공받는 트리이고, 도 8(b)는 사용자 UC가 제공받는 트리이며, 도 8(c)는 사용자 UE가 제공받는 트리이다.
- [0085] 사용자 UA가 도 8(a)의 레이어 A에 홈 레이어를 갖는다면 사용자의 변수 "home" 값은 0이다. 또한 사용자는 도 8(a)의 모든 레이어에 대해 접근 권한을 가질 수 있다. 배열 "esk[]"에는 사용자가 접근할 수 있는 레이어들에 대한 세션 키들이 사용자의 공개키로 암호화되어 저장된 ESKNODE들의 인덱스 번호가 들어있다. 사용자 X가 레이어 Y를 홈 레이어로 등록한다면 레이어 Y 및 레이어 Y에서 시작하는 트리의 모든 하위 레이어가 접근 가능 레이어로 될 수 있다. 레이어 Y 및 하위 레이어의 세션 키가 ESKNTAB에 암호화하여 저장되고 저장된 위치가 배열 "esk[]"에 저장될 수 있다.
- [0086] 도 9는 사용자 UA, 사용자 UC 및 사용자 UE의 사용자 노드 테이블을 보여준다. 사용자 UA는 레이어 A(LNODE 0), 사용자 UC는 레이어 A/C(LNODE 2) 및 사용자 UE는 레이어 A/C/D/E(LNODE 4)에 홈 레이어를 두고 있다. 사용자 UA의 홈 레이어(A 혹은 LNODE 0)는 트리의 루트이다. 사용자 UA는 트리의 모든 레이어를 접근할 수 있어야 하며 이를 위해 사용자 UA의 공개키로 모든 레이어의 세션 키를 암호화하여 저장한다.
- [0087] 도 10의 ESKNTAB 테이블은 인덱스 5 내지 9에 사용자 UA가 접근할 수 있는 모든 레이어의 암호화된 세션 키가 저장된다. 도 10의 ESKNTAB 테이블은 각 세션 키 옆에 변수 "lnode_idx"가 나타나 있으며 이 열이 타겟 LNODE

인덱스를 나타낸다. 도 10의 각 세션 키의 인덱스 값은 사용자 UA가 저장되어 있는 UNODE 0의 배열 "esk[]"에 저장된다. 사용자 UC 및 사용자 UE도 유사하게 등록된다. 사용자 UC는 홈 레이어 A/C를 가지고 있고, 레이어 A/C, 레이어 A/C/D 그리고 레이어 A/C/D/E를 접근할 수 있다. 레이어의 세션키 역시 사용자 UC의 공개키로 암호화되어 ESKNTAB 테이블 인덱스 10 내지 12에 저장된다. 인덱스 10 내지 12는 사용자 UC의 UNODE인 UNODE 1에 저장된다. UE는 홈 레이어 A/C/D/E에 있고, 하위 레이어가 없으므로 홈 레이어의 세션 키만 암호화되어 ESKNTAB 테이블의 인덱스 13에 저장된다.

[0088] ESKNTAB는 하기의 구조를 가진다.

```
[0089] class ESKNode{
[0090]     // on-disk data
[0091]     unsigned char ekey[EKEY_LEN];
[0092]     int lnode_idx;
[0093]     // on-memory data follows
[0094]     .....
[0095] };
```

[0096] 배열 "ekey[]"는 암호화된 세션 키이고, 변수 "lnode_idx"가 타겟 LNODE의 레이어 노드 테이블 인덱스이다. 도 10은 ESKNTAB 테이블의 예를 보여준다. 도 10의 ESKNTAB 테이블은 14개의 ESKNODE가 저장된다. ESKNODE 0 내지 4는 LNODE 0 내지 4의 세션 키를 MLE 서버의 마스터 공개키로 암호화한 값이 저장된다. ESKNODE 5 내지 9도 LNODE 0 내지 4에 대한 암호화된 세션 키이다. 하지만 ESKNODE 5 내지 9는 UA의 공개키로 암호화한 값들이다. UA는 자신의 개인키로 암호화한 값들을 복호화하여 모든 레이어에 접근할 수 있다. ESKNODE 10, 11 및 12는 LNODE 2, 3 및 4의 키가 UC의 공개키로 암호화한 값을 저장한다. UC는 공개키들을 복호화하여 레이어 A/C 및 레이어 A/C의 하위 레이어들을 접근 할 수 있다. 마지막으로 ESKNODE 13은 LNODE 4의 세션 키를 UE의 공개키로 암호화하고 저장하며, UE가 레이어 A/C/D/E를 접근하는데 사용된다.

[0097] 도 11은 본 발명의 일 실시예에 따른 MLE 파일의 전체 구조를 도시한 예시도이다.

[0098] 도 11을 참조하면, MLE 파일은 레이어 노드 테이블, 사용자 노드 테이블 및 암호화 세션 키 노드 테이블을 포함할 수 있다.

[0099] 레이어 노드 테이블은 레이어별 위치정보 및 레이어별 암호화된 키를 저장할 수 있다. 레이어 노드 테이블은 MLE 파일의 익스텐트 3 내지 12에 할당될 수 있다.

[0100] 사용자 노드 테이블은 MLE 파일에 대한 접근 권한이 있는 등록 사용자들에 대한 정보를 저장할 수 있다. 사용자 노드 테이블은 MLE 파일의 익스텐트 14 내지 23에 할당될 수 있다.

[0101] 암호화 세션 키 노드 테이블은 암호화된 키 및 암호화된 키가 사용되는 타겟 노드의 인덱스 값을 저장할 수 있다. 암호화 세션 키 노드 테이블은 MLE 파일의 익스텐트 25 내지 34에 할당될 수 있다.

[0102] MLE 파일은 각 테이블마다 테이블의 어떤 엔트리가 비어있고 어떤 엔트리가 사용되고 있는지 정보를 보여주는 비트맵을 포함한다.

[0103] LNBم는 LNTAB의 비트맵으로서 익스텐트 2에 할당될 수 있다. UNBM은 UNTAB의 비트맵으로서 익스텐트 13에 할당될 수 있다. ESKNBم는 ESKNTAB의 비트맵으로서 익스텐트 24에 할당될 수 있다. 또한 EBM 비트맵은 익스텐트 1에 할당될 수 있다. EBM 비트맵은 현재 MLE 파일의 각 익스텐트의 사용 및 비사용 여부를 보여준다. 마지막으로 익스텐트 0는 각 테이블이나 비트맵의 위치를 저장되는 수퍼블록일 수 있다.

[0104] MLE 파일이 처음 생성되면 메타 데이터들이 먼저 생성된다. MLE 파일은 루트 레이어를 위해 2개의 익스텐트가 추가로 할당될 수 있다. MLE 파일은 익스텐트 35 및 36을 루트 데이터 익스텐트로 할당될 수 있다. MLE 파일은

메타 데이터 이후 생성되는 다른 레이어는 초기 상태로 두 개의 익스텐트를 할당하여 하나는 데이터를 저장하고, 다른 하나는 저장된 레이어 밑으로 생성될 하위 레이어들의 이름과 LNODE 번호를 저장한다. 데이터 익스텐트는 데이터가 증가함에 따라 추가로 할당될 수 있다.

[0105] 슈퍼블록인 익스텐트 0의 구조는 하기와 같다.

```
[0106] class Superblock{
[0107]     // on-disk data
[0108]     int  fsize;
[0109]     char  marker[8];
[0110]     char  version;
[0111]     char  reserve[3];
[0112]     unsigned long  loc_ebm;
[0113]     unsigned long  loc_lnbm;
[0114]     unsigned long  loc_lntab;
[0115]     unsigned long  loc_unbm;
[0116]     unsigned long  loc_untab;
[0117]     unsigned long  loc_esknbm;
[0118]     unsigned long  loc_eskntab;
[0119]     // on-memory data follows
[0120]     켈켈
[0121] }
```

[0122] MLE 파일 시스템은 사용자가 사용자 프로그램을 이용하여 MLE 파일을 열고 읽거나 쓸 수 있도록 라이브러리 함수들을 정의할 수 있다. MLE 파일 시스템은 MLE_fcreate, MLE_lcreate, MLE_regiter_user, MLE_fopen, MLE_fread, MLE_fwrite, MLE_fclose 중 적어도 하나를 포함할 수 있다. 특히 MLE_fcreate, MLE_lcreate 및 MLE_regiter_user 함수는 관리자 프로그램만 사용할 수 있는 함수이다.

[0123] 함수 "MLE_fcreate()"은 MLE 파일을 생성하는 함수이다. 함수 "MLE_lcreate()"은 레이어를 추가하는 함수이다. 함수 "MLE_register_user()"는 사용자를 MLE 파일에 등록하는 함수이다. 또한 함수 "MLE_fopen()"은 저장부에 저장되어 있는 기존의 MLE 파일을 열어주는 함수이다. 함수 "MLE_fopen()"은 지정된 MLE 파일을 열고 모든 메타 데이터를 읽어서 하기의 구조체 "MLE_fs"를 생성할 수 있다.

```
[0124] class MLE_fs{
[0125]     Superblock *sb;
[0126]     EBM *ebm;
[0127]     Lnbm *lnbm;
[0128]     LayerNodeTable *lntab;
[0129]     Unbm *unbm;
[0130]     UserNodeTable *untab;
```

```

[0131]         Esknbm *esknbm;
[0132]         ESKNodeTable *esknTAB;
[0133]         Extents *extents;
[0134]         .....
[0135]     };

```

[0136] 구조체 "MLE_fs"는 해당 MLE 파일의 모든 메타 정보, 부모-자식 레이어 관계 및 각 사용자의 데이터 익스텐트의 물리적 위치 등 실행시간에 필요한 추가 정보를 가질 수 있다. MLE 파일은 접근하기 위해서 함수 "MLE_fcreate()" 또는 함수 "MLE_fopen()"을 호출하여 구조체 "MLE_fs"를 먼저 생성해야 한다.

[0137] 함수 "MLE_fwrite()"은 데이터를 MLE 파일에 기록하는 함수이다. 함수 "MLE_fwrite()"은 사용자가 데이터를 쓰려는 익스텐트에 사용자의 접근 권한이 있는 레이어어야 한다. 데이터는 해당 레이어의 세션 키로 암호화되어 기록될 수 있다.

[0138] 함수 "MLE_fread()"은 데이터를 읽기 위한 함수이다. 함수 "MLE_fread()"은 해당 레이어에 대해 사용자의 접근 권한이 있는 경우에만 허용된다. MLE 서버는 사용자가 요청한 레이어의 데이터를 암호화된 상태로 해당 레이어의 세션 키 (사용자의 공개키로 암호화된 키)와 함께 사용자에게 보낸다. 사용자 프로그램은 이 세션 키를 복호화하여 그 키로 데이터를 다시 복호화 할 수 있다.

[0139] 도 12는 본 발명의 일 실시예에 따른 서버와 사용자의 관계를 도시한 흐름도이다.

[0140] 도 12를 참조하면, 서버와 사용자의 관계는 MLE 파일을 사용하여 여러 사용자가 같은 파일에서 동시 작업을 하기 위해 단일 서버와 다중 사용자로 이루어지는 관계를 구현할 수 있다. 서버와 사용자는 SSL 프로토콜을 이용하여 암호화된 통신을 할 수 있다. 서버는 다중 사용자를 동시에 서비스하기 위해 함수 "select()"의 시스템 콜을 사용할 수 있다.

[0141] 사용자 패킷은 SSL 연결 요청 패킷, 사용자 패킷 "HELLO" 및 MLE 파일 명령어 패킷 중 적어도 하나의 패킷일 수 있다.

```

[0142] process_client_packet(){
[0143]     if SSL connection request packet
[0144]         call accept() to establish a connection;
[0145]     else if client HELLO
[0146]         call SSL_accept() to process SSL handshake protocol;
[0147]     else if an MLE client command
[0148]         process an MLE client command;
[0149] }

```

[0150] 사용자가 서버에 접속하여 협업 디자인을 작업하기 위해서는 다음과 같다.

[0151] 사용자 패킷이 SSL 연결 요청한다(S110).

[0152] 서버는 SSL 연결 요청을 확인 후, 사용자에게 서버 접속 허가한다(S120). 서버는 사용자와의 통신을 위한 소켓을 따라 만들고 소켓 번호를 리셋(rset) 비트 벡터에 표시할 수 있다. 다음 라운드에서의 함수 "select()"는 새로 생성된 소켓들까지 포함하여 모니터링을 할 수 있다.

[0153] 사용자는 서버 접속을 확인 후, 패킷 "HELLO"을 서버에 전달한다(S130). 사용자 패킷이 패킷 "HELLO"이면 패킷

은 해당 사용자와의 통신을 위해 할당된 소켓을 통해 들어온 패킷일 수 있다.

- [0154] 서버는 사용자가 SSL 프로토콜을 시작을 허가한다(S140). 서버는 함수 "SSL_accept()"를 호출하여 사용자와 SSL 프로토콜을 진행할 수 있다. SSL 프로토콜이 성공적으로 진행된다면, 서버와 클라이언트는 해당 통신 소켓을 통해 암호화된 통신을 할 수 있다.
- [0155] 사용자는 MLE 파일 명령어 패킷을 보낸다(S150). 사용자가 보낸 MLE 파일 명령어는 MLEFS_FCREATE, MLEFS_LCREATE, MLEFS_REGISTER_USER, MLEFS_FOPEN, MLEFS_READ, MLEFS_WRITE 및 MLEFS_CLOSE 중 적어도 하나이다.
- [0156] 서버는 사용자의 MLE 파일 명령어에 대한 실행을 한다(S160). 서버는 사용자의 MLE 파일 명령어에 해당되는 MLE 함수를 호출할 수 있다.
- [0157] 도 13을 본 발명의 일 실시예에 따른 다중 레이어 암호화 파일 생성 방법을 도시한 순서도이다.
- [0158] 도 13을 참조하면, 다중 레이어 암호화 파일 생성 방법은 다중 사용자가 동일한 디자인 파일을 안전하게 협업을 할 수 있게 한다.
- [0159] 사용자는 저장부(130)에 저장된 디자인 파일을 접속을 한다(S210). 사용자는 SSL 프로토콜을 이용하여 서버에 접속할 수 있다. 입력부(110)는 사용자 패킷을 입력을 받을 수 있다. 제어부(120)는 사용자가 협업에 사용되는 디자인 파일을 저장부(130)에서 불러올 수 있다.
- [0160] 트리구조 생성부(121)는 디자인 파일에서 적어도 하나의 객체 중 하나 또는 하나 이상의 객체를 포함하는 레이어의 계층적 구조에 따라 트리를 생성한다(S220). 트리구조 생성부(121)는 데이터 셀을 계층적 구조로 파일에 저장하는 HDF 기법을 이용할 수 있다.
- [0161] 레이어 암호화부(123)는 트리에 포함된 객체를 서로 다른 키로 암호화한다(S230). 레이어 암호화부(123)는 ROI를 설정하고 각 ROI들을 서로 다른 키로 암호화하여 접근 권한을 부여할 수 있다.
- [0162] 헤더 생성부(123)는 레이어의 위치정보 및 암호화된 적어도 하나의 키에 관한 정보를 저장하는 헤더를 생성한다(S240). 헤더 생성부(123)는 암호화된 키를 합법적 사용자의 공개키로 다시 암호화하여 저장하는 헤더를 생성할 수 있다.
- [0163] 다중 레이어 암호화 파일 생성부(124) 트리의 계층적 구조 및 헤더의 정보를 갖는 MLE 파일을 생성할 수 있다(S250). MLE 파일은 각 사용자가 자신에게 허용된 레이어들에 대해서만 작업을 할 수 있도록 강제함으로써 다중 사용자가 안전하게 같은 디자인 파일에 대해 협업할 수 있게 한다.
- [0164] 본 발명은 또한 컴퓨터로 읽을 수 있는 기록매체에 컴퓨터가 읽을 수 있는 코드로서 구현하는 것이 가능하다. 컴퓨터가 읽을 수 있는 기록매체는 컴퓨터 장치에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록 장치를 포함한다. 컴퓨터가 읽을 수 있는 기록매체의 예로는 ROM, RAM, CD-ROM, 자기 테이프, 플로피디스크, 광 데이터 저장장치 등이 있으며, 또한 캐리어 웨이브(예를 들어 인터넷을 통한 전송)의 형태로 구현되는 것도 포함한다. 또한 컴퓨터가 읽을 수 있는 기록매체는 네트워크로 연결된 컴퓨터 장치에 분산되어 분산방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수 있다.

부호의 설명

- [0165] 1: 다중 레이어 암호화 파일 생성 장치
- 110: 입력부
- 120: 제어부
- 121: 트리구조 생성부
- 122: 레이어 암호화부
- 123: 헤더 생성부
- 124: 다중 레이어 암호화 파일 생성부

130: 저장부

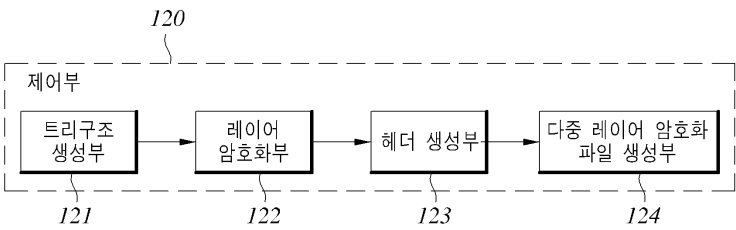
도면

도면1

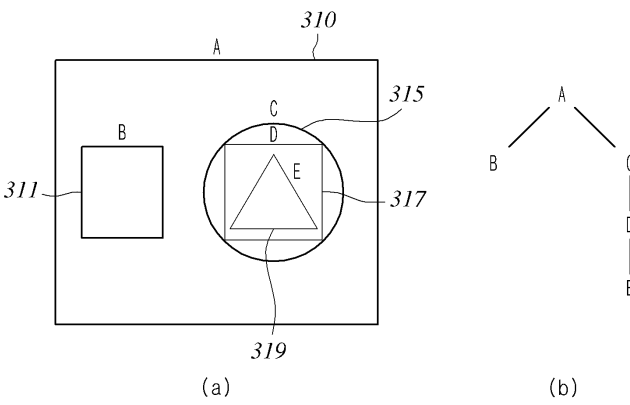
1



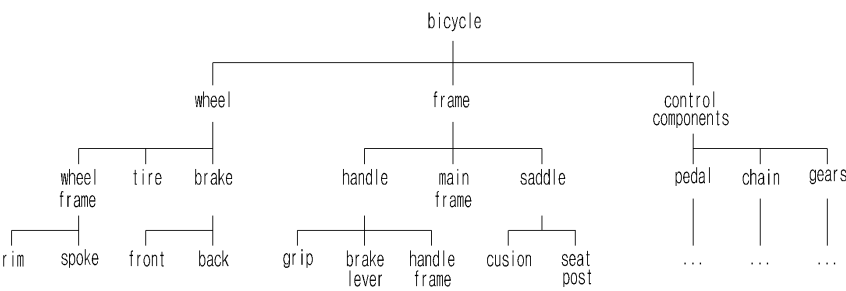
도면2



도면3



도면4



도면5

	num_ext	loc[0]	loc[1]...loc[9]	loc_sub	esk
0	1	37		38	0
1	1	39		40	1
2	1	41		42	2
3	1	43		44	3
4	1	45		49	4
					

도면6

ext 38

lname	lnode_idx
B	1
C	2

(a)

ext 42

lname	lnode_idx
D	3

(b)

ext 44

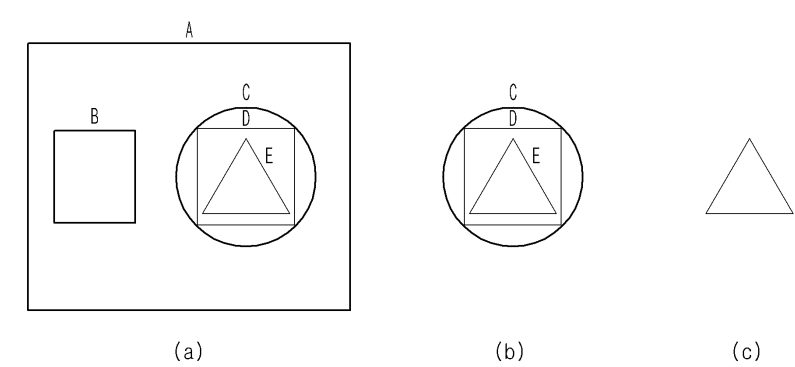
lname	lnode_idx
E	4

(c)

도면7

	ekey	lnode_idx
0	23 24 3a 4b	0
1	d 2c 44 52	1
2	21 110 3b	2
3	a2 67 21 33	3
4	de 34 55 4b	4

도면8



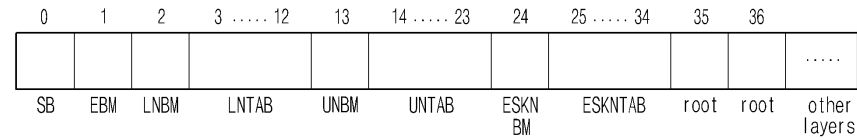
도면9

	uname	home	esk[0]	esk[1]	esk[2]	esk[3]	esk[4]	
0	UA	0	5	6	7	8	9	
1	UC	2	10	11	12			
2	UE	4	13					

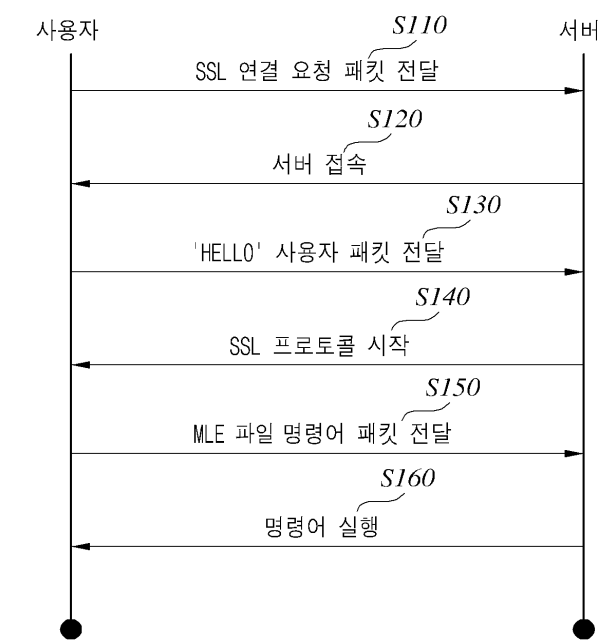
도면10

	ekey	Inode_idx
...		...
5	1d 22 24 52	0
6	11 22 33 44	1
7	12 aa bb 22	2
8	d1 24 54 4b	3
9	1a 2a 3a 22	4
10	13 14 15 16	2
11	21 31 41 51	3
12	1e 2e 3e 4e	4
13	22 33 41 53	4

도면11



도면12



도면13

