(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2019-0038561
(43) 공개일자 2019년04월08일

(51) 국제특허분류(Int. Cl.)
H04L 9/32 (2006.01)
(52) CPC특허분류
H04L 9/3218 (2013.01)
H04L 9/3239 (2013.01)
(21) 출원번호 10-2019-7003851
(22) 출원일자(국제) 2017년07월07일
심사청구일자 없음
(85) 번역문제출일자 2019년02월08일
(86) 국제출원번호 PCT/GB2017/052004
(87) 국제공개번호 WO 2018/007828
국제공개일자 2018년01월11일
(30) 우선권주장
1611948.9 2016년07월08일 영국(GB)

(71) 출원인
칼립톤 인터네셔널 리미티드
지브롤터 지브롤터 자이로스 페시지 지브로 하우스 4
(72) 발명자
데이비스, 라스
영국 웨스트 석세스 알에이치19 3에이에프 이스트 그린스테드 37 하이 스트리트 크라운 하우스
(74) 대리인
특허법인 무한

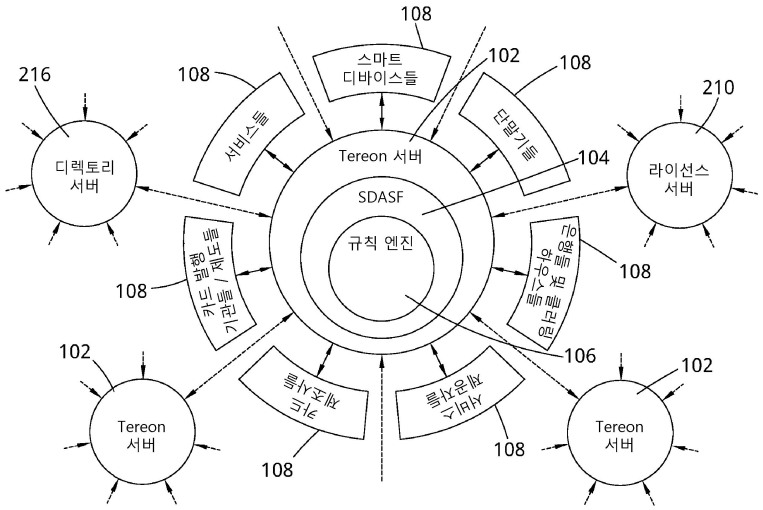
전체 청구항 수 : 총 196 항

(54) 발명의 명칭 **분산 트랜잭션 처리 및 인증 시스템**

(57) 요 약

제1 엔티티와 관련된 장치에서 데이터 트랜잭션 레코딩 방법은 제1 시드 데이터를 결정하는 단계와, 상기 제1 엔티티 및 제2 엔티티간의 제1 데이터 트랜잭션의 상기 레코드를 생성하는 단계와, 적어도 상기 제1 시드 데이터 및 상기 제1 데이터 트랜잭션의 레코드를 결합하여 제2 시드 데이터를 결정하는 단계와, 상기 제2 시드 데이터를 해싱하여 제1 해시를 생성하는 단계? 상기 제1 해시는 상기 제1 엔티티와 관련된 데이터 트랜잭션들의 히스토리를 포함함 -와, 상기 제1 데이터 트랜잭션의 상기 레코드에 대한 상기 제1 해시를 메모리에 저장하는 단계를 포함한다.

대 표 도



(52) CPC특허분류
H04L 2209/38 (2013.01)
H04L 2209/56 (2013.01)

명 세 서

청구범위

청구항 1

제1 엔티티와 관련된 장치에서 데이터 트랜잭션 레코딩 방법에 있어서,

제1 시드 데이터를 결정하는 단계;

상기 제1 엔티티 및 제2 엔티티간의 제1 데이터 트랜잭션의 상기 레코드를 생성하는 단계;

적어도 상기 제1 시드 데이터 및 상기 제1 데이터 트랜잭션의 레코드를 결합하여 제2 시드 데이터를 결정하는 단계;

상기 제2 시드 데이터를 해싱하여 제1 해시를 생성하는 단계? 상기 제1 해시는 상기 제1 엔티티와 관련된 데이터 트랜잭션들의 히스토리를 포함함 -; 및

상기 제1 데이터 트랜잭션의 상기 레코드에 대한 상기 제1 해시를 메모리에 저장하는 단계

를 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 2

제1항에 있어서,

상기 제1 시드 데이터는 스타팅 해시(starting hash)를 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 3

제2항에 있어서,

상기 스타팅 해시는 상기 제1 엔티티와 관련된 이전 데이터 트랜잭션의 레코드를 해싱한 결과인 데이터 트랜잭션 레코딩 방법.

청구항 4

제2항에 있어서,

상기 스타팅 해시는 랜덤 해시(random hash)를 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 5

제4항에 있어서,

상기 랜덤 해시는 상기 장치로부터의 서명, 상기 랜덤 해시가 생성된 날짜 및/또는 시간 중에서 적어도 하나를 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 6

제1항 내지 제5항 중 어느 한 항에 있어서,

제2 시드 데이터를 제공하는 단계는,

상기 제1 시드 데이터 및 상기 제1 데이터 트랜잭션의 상기 레코드와 제1 영지식 증명(first zero-knowledge proof) 및 제2 영지식 증명(second zero-knowledge proof)을 결합하는 단계

를 더 포함하고,

상기 제1 영지식 증명은 상기 스타팅 해시가 상기 제1 엔티티와 관련된 상기 이전 데이터 트랜잭션의 상기 트루 해시를 포함한다는 증명을 포함하고,

상기 제2 영지식 증명은 제2 해시가 상기 제2 엔티티와 관련된 이전 데이터 트랜잭션의 상기 트루 해시를 포함한다는 증명을 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 7

제6항에 있어서,

제2 시드 데이터를 제공하는 단계는,

상기 제1 시드 데이터, 상기 제1 데이터 트랜잭션의 상기 레코드, 상기 제1 영지식 증명 및 상기 제2 영지식 증명과 제3 영지식 증명을 결합하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 8

제7항에 있어서,

상기 제3 영지식 증명은 랜덤 데이터로부터 생성되는 데이터 트랜잭션 레코딩 방법.

청구항 9

제7항에 있어서,

상기 제3 영지식 증명은 상기 제1 영지식 증명 또는 상기 제2 영지식 증명의 반복인 데이터 트랜잭션 레코딩 방법.

청구항 10

제7항에 있어서,

상기 제3 영지식 증명은 상기 제2 영지식 증명에 대응하는 상기 제1 데이터 트랜잭션의 제2 레코드를 이용하여 구성되는 데이터 트랜잭션 레코딩 방법.

청구항 11

제6항에 있어서,

상기 제1 데이터 트랜잭션은 적어도 두 개의 스테이지들을 포함하고,

제2 시드 데이터를 제공하는 단계는

상기 제1 데이터 트랜잭션의 상기 제1 스테이지의 레코드와 상기 제1 영지식 증명을 결합하는 단계; 및

상기 제1 데이터 트랜잭션의 상기 제2 스테이지의 레코드와 상기 제2 영지식 증명을 결합하는 단계

를 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 12

제11항에 있어서,

제2 시드 데이터를 제공하는 단계는,

상기 제1 데이터 트랜잭션의 상기 제2 스테이지의 레코드로부터 제3 영지식 증명을 구성하는 단계; 및

상기 제1 데이터 트랜잭셩의 상기 제2 스테이지의 레코드와 상기 제2 영지식 증명 및 상기 제3 영지식 증명을 결합하는 단계

를 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 13

제11항에 있어서,

상기 제1 데이터 트랜잭션은 적어도 세 개의 스테이지들을 포함하고,

제2 시드 데이터를 제공하는 단계는,

상기 제1 데이터 트랜잭션의 상기 제3 스테이지의 레코드와 상기 제1 영지식 증명을 결합하는 단계; 및

상기 제1 데이터 트랜잭션의 상기 제3 스테이지의 레코드와 상기 제3 영지식 증명을 결합하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 14

제11항에 있어서,

상기 제1 데이터 트랜잭션은 적어도 세 개의 스테이지들을 포함하고,

제2 시드 데이터를 제공하는 단계는

상기 제1 데이터 트랜잭션의 상기 제3 스테이지의 레코드와 상기 제1 영지식 증명을 결합하는 단계; 및

랜덤 데이터와 상기 제3 영지식 증명을 결합하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 15

제11항에 있어서,

상기 제1 데이터 트랜잭션은 적어도 세 개의 스테이지들을 포함하고,

제2 시드 데이터를 제공하는 단계는,

상기 제1 데이터 트랜잭션의 상기 제3 스테이지의 레코드와 상기 제1 영지식 증명을 결합하는 단계; 및

상기 제1 데이터 트랜잭션의 제4 스테이지의 레코드와 상기 제2 영지식 증명을 결합하는 단계

를 포함하고,

상기 제1 데이터 트랜잭션의 상기 제4 스테이지는 상기 제1 데이터 트랜잭션의 상기 제3 스테이지의 반복인 데이터 트랜잭션 레코딩 방법.

청구항 16

제11항에 있어서,

상기 제1 데이터 트랜잭션은 적어도 세 개의 스테이지들을 포함하고,

제2 시드 데이터를 제공하는 단계는,

상기 제1 데이터 트랜잭션의 상기 제3 스테이지의 레코드와 제3 영지식 증명을 결합하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 17

제6항 내지 제16항 중에서 어느 한 항에 있어서,

상기 제1 영지식 증명은 상기 제1 엔티티와 관련된 상기 장치에 의해 구성되고,

상기 제2 영지식 증명은 상기 제2 엔티티와 관련된 장치에 의해 구성되는 데이터 트랜잭션 레코딩 방법.

청구항 18

제17항에 있어서,

상기 제1 영지식 증명 및 상기 제2 영지식 증명을 구성하는 단계는,

키 교환 알고리즘을 사용하는 단계

를 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 19

제18항에 있어서,

상기 키 교환 알고리즘은 PAKE 알고리즘을 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 20

제1항 내지 제19항 중 어느 한 항에 있어서,

상기 제2 엔티티와 관련된 장치에 상기 제1 해시를 전송하는 단계;

상기 제2 엔티티와 관련된 장치로부터 제2 해시를 수신하는 단계- 상기 제2 해시는 상기 제2 엔티티와 관련된 이전 데이터 트랜잭션의 해시를 포함함 -;

상기 제1 파티(first party) 및 상기 제2 파티(second party)간의 제2 데이터 트랜잭션의 레코드를 생성하는 단계;

상기 제1 해시 및 상기 제2 해시와 상기 제2 데이터 트랜잭션의 상기 레코드를 결합하여 제3 시드 데이터를 결정하는 단계;

상기 제3 시드 데이터를 해싱하여 제3 해시를 생성하는 단계- 상기 제3 해시는 상기 제1 엔티티와 관련된 데이터 트랜잭션들의 히스토리 및 상기 제2 엔티티와 관련된 데이터 트랜잭션들의 히스토리를 포함함 ?;

상기 제2 데이터 트랜잭션의 상기 레코드에 대한 상기 제3 해시를 상기 메모리에 저장하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 21

제20항에 있어서,

제3 시드 데이터를 제공하는 단계는,

상기 제2 데이터 트랜잭션의 상기 레코드, 상기 제1 해시 및 상기 제2 해시와 제3 영지식 증명 및 제4 영지식 증명을 결합하는 단계

를 더 포함하고,

상기 제3 영지식 증명은 상기 제1 해시가 상기 제1 데이터 트랜잭션의 트루 해시를 포함한다는 증명을 포함하고,

상기 제4 영지식 증명은 상기 제2 해시가 상기 제2 엔티티와 관련된 상기 이전 데이터 트랜잭션의 상기 트루 해시를 포함한다는 증명을 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 22

제20항 또는 제21항에 있어서,

상기 제2 엔티티와 관련된 상기 이전 데이터 트랜잭션은 상기 제1 데이터 트랜잭션인 데이터 트랜잭션 레코딩 방법.

청구항 23

제1항 내지 제22항 중 어느 한 항에 있어서,

상기 제1 엔티티 및/또는 상기 제2 엔티티의 식별자와 상기 해시들 각각을 연관 시키는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 24

제1항 내지 제23항 중 어느 한 항에 있어서,

상기 제1 해시를 재계산하는 단계; 및

매치(match)를 결정하기 위해서 상기 생성된 제1 해시를 상기 재계산된 제2 해시와 비교하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 25

제24항에 있어서,

상기 비교가 성공적이지 않는 경우, 추가 데이터 트랜잭션들을 취소하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 26

제1항 내지 제25항 중 어느 한 항에 있어서,

상기 제1 데이터 트랜잭션에 대응하는 시스템 해시를 시스템 장치에 생성하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 27

제26항에 있어서,

제2 시드 데이터를 제공하는 단계는,

상기 제1 시드 데이터 및 상기 제1 데이터 트랜잭션의 상기 레코드와 상기 시스템 해시를 결합하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 28

제26항 또는 제27항에 있어서,

상기 시스템 해시는 상기 시스템 장치상의 이전 데이터 트랜잭션의 레코드를 해싱한 결과인 데이터 트랜잭션 레코딩 방법.

청구항 29

제1항 내지 제28항 중 어느 한 항에 있어서,

제2 시드 데이터를 제공하는 단계는,

라이선스 장치로부터 라이선스 해시를 수신하는 단계; 및

상기 제2 시드 데이터를 제공하기 위해서 상기 제1 시드 데이터 및 상기 제1 데이터 트랜잭션의 상기 레코드와 상기 라이선스 해시를 결합하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 30

제29항에 있어서, 상기 라이선스 장치에서:

상기 제1 해시를 수신하는 단계;

라이선스 입력을 제공하기 위해서 상기 라이선스 해시와 상기 제1 해시를 결합하는 단계;

상기 라이선스 입력을 해싱하여 제2 라이선스 해시를 생성하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 31

제1항 내지 제30항 중 어느 한 항에 있어서,

제2 시드 데이터를 제공하는 단계는,

디렉토리 장치로부터 디렉토리 해시를 수신하는 단계; 및

상기 제2 시드 데이터를 제공하기 위해서 상기 제1 시드 데이터 및 상기 제1 데이터 트랜잭션의 상기 레코드와 상기 디렉토리 해시를 결합하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 32

제31항에 있어서, 상기 디렉토리 서버에서:

상기 제1 해시를 수신하는 단계;

디렉토리 입력을 제공하기 위해서 상기 디렉토리 해시와 상기 제1 해시를 결합하는 단계;

상기 디렉토리 입력을 해싱하여 제2 디렉토리 해시를 생성하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 33

제1항 내지 제32항 중 어느 한 항에 있어서,

제2 시드 데이터를 제공하는 단계는,

상기 제1 데이터 트랜잭션에 대한 암호화 키로부터 키 해시를 생성하는 단계; 및

상기 제2 시드 데이터를 제공하기 위해서 상기 제1 시드 데이터 및 상기 제1 데이터 트랜잭션의 상기 레코드와 상기 키 해시를 결합하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 34

제33항에 있어서,

상기 암호화 키는 공개 키 또는 개인 키를 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 35

제1항 내지 제34항 중 어느 한 항에 있어서,

상기 제1 시드 데이터 및 상기 제1 데이터 트랜잭션의 상기 레코드를 결합하는 단계는 상기 제1 데이터 트랜잭션이 완료되자마자 수행되는 데이터 트랜잭션 레코딩 방법.

청구항 36

제1항 내지 제35항 중 어느 한 항에 있어서,

상기 메모리는 원격 장치에 위치하는 데이터 트랜잭션 레코딩 방법.

청구항 37

제36항에 있어서,

다른 장치들로부터 수신된 해시들에 대응하는 상기 제1 해시를 상기 원격 장치에서 비교하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 38

제36항 또는 제37항에 있어서,

상기 장치에 연결된 다른 장치들에 상기 제1 해시를 수신할 것을 예상하도록 통지하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 39

제1항 내지 제38항 중 어느 한 항에 있어서,

상기 메모리에 해시들의 체인을 저장하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 40

제39항에 있어서,

전송된 상기 해시 체인들에 대한 액세스를 제한하도록 하는 장치 상에 위치한 제2 메모리에 상기 해시들의 체인을 전송하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 41

제39항 또는 제40항에 있어서,

상기 해시 체인에서 해시를 수정 또는 삭제하는 단계

를 더 포함하고,

상기 해시 체인에서 해시를 수정 또는 삭제하는 단계는,

상기 해시 체인에서 서브젝트 해시를 재생성하는 단계;

상기 레코드가 수정되지 않았는지 여부를 확인하는 단계;

상기 재생성된 해시를 레코딩하는 단계;

상기 레코드를 수정 또는 삭제하는 단계;

상기 서브젝트 해시의 결합 및 상기 수정/삭제된 레코드를 해싱하여 상기 레코드에 대한 새로운 해시를 생성하는 단계; 및

상기 새로운 해시를 레코딩하는 단계

를 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 42

제41항에 있어서,

상기 새로운 해시를 이용하여 시스템 해시를 생성하는 단계

를 더 포함하는 데이터 트랜잭션 레코딩 방법.

청구항 43

제1 엔티티와 관련된 장치에 있어서,

상기 장치는 제1항 내지 제42항 중 어느 한 항의 방법을 수행하는 장치.

청구항 44

제43항에 있어서,

상기 장치는 서버를 포함하는 장치.

청구항 45

제43항에 있어서,

상기 장치는 유저 장치를 포함하는 장치.

청구항 46

제45항에 있어서,

상기 유저 장치는 개인용 컴퓨터, 스마트폰, 스마트 태블릿 또는 사물 인터넷(IoT) 가능 장치 중에서 적어도 하나를 포함하는 장치.

청구항 47

제46항에 있어서,

상기 유저 장치는 상기 장치상의 메모리에서 상기 제1 해시를 저장하는 장치.

청구항 48

제47항에 있어서,

상기 유저 장치는 해당 서버로부터 오프 라인인 경우에만, 상기 장치상의 메모리에서 상기 제1 해시를 저장하는 장치.

청구항 49

제43항 내지 제48항 중 어느 한 항에 있어서,

상기 장치는 상기 제2 엔티티와 관련된 장치에 상기 제1 해시를 전송하는 장치.

청구항 50

제49항에 있어서,

상기 장치는 상기 제1 데이터 트랜잭션의 상기 레코드의 서명되고, 암호화된 카피를 상기 제2 엔티티와 관련된 상기 장치로 전송하고,

상기 서명은 상기 제1 데이터 트랜잭션의 상기 레코드(that record)에 대한 목적지 서버의 표시(indication)을

포함하는 장치.

청구항 51

제50항에 있어서,

상기 장치는 특정 오프라인 공개 키로 상기 레코드에 서명하는 장치.

청구항 52

제50항에 있어서,

상기 장치는 상기 장치에 속하는 키로 상기 레코드에 서명하는 장치.

청구항 53

제50항 내지 제52항 중 어느 한 항에 있어서,

상기 목적지 서버만 상기 제1 데이터 트랜잭션의 상기 레코드의 상기 암호화된 카피(encrypted copy)을 해독할 수 있는 장치.

청구항 54

제48항 내지 제53항 중 어느 한 항에 있어서,

상기 장치가 대응하는 서버와 연결을 회복할 때, 상기 장치는 상기 관련된 해시들 및 그것의 오프라인 데이터 트랜잭션들의 상기 암호화된 레코드들을 대응하는 서버로 전송하는 장치.

청구항 55

제54항에 있어서,

상기 장치는 자신이 보유하는 다른 엔티티들을 포함하는 데이터 트랜잭션들의 레코드들의 복사본들을 상기 다른 엔티티들에 대응하는 서버들로의 전송을 위해 자신에 대응하는 서버에 전송하는 장치.

청구항 56

제55항에 있어서,

상기 전송은 상기 레코드들이 적용되는 모든 서버들에 상기 레코드들을 수신할 것을 기대하도록 통지하는 것을 포함하는 장치.

청구항 57

제43항 내지 제56항 중 어느 한 항에 있어서,

상기 장치는 상기 제1 데이터 트랜잭션에서 이것의 부분을 식별하기 위해 고유의 내부 트랜잭션 번호를 생성하는 장치.

청구항 58

라이선스 장치는,

제1 엔티티와 관련된 장치로부터 제1 해시를 수신하고- 상기 제1 해시는 상기 제1 엔티티에 관련된 데이터 트랜잭션들의 히스토리를 포함함 -;

라이선스 입력을 제공하기 위해서 라이선스 해시와 상기 제1 해시를 결합하고;

상기 라이선스 입력을 해싱하여 제2 라이선스 해시를 생성하고; 및

메모리에 상기 제2 라이선스 해시를 저장하는 라이선스 장치.

청구항 59

디렉토리 장치는,

제1 엔티티와 관련된 장치로부터 제1 해시를 수신하고- 상기 제1 해시는 상기 제1 엔티티와 관련된 데이터 트랜잭션들의 히스토리를 포함함 -;

디렉토리 입력을 제공하기 위해 디렉토리 해시와 상기 제1 해시를 결합하고;

상기 라이선스 입력을 해싱하여 제2 디렉토리 해시를 생성하고; 및

메모리에 상기 제2 디렉토리 해시를 저장하는 디렉토리 장치.

청구항 60

실행될 때 컴퓨팅 장치가 제1항 내지 제42항 중 어느 한 항의 방법을 수행하게 하는 코드 부분들을 포함하는 컴퓨터 판독 가능 매체.

청구항 61

장치로부터 제1 서비스에 액세스하는 방법에 있어서,

요청 서버에 상기 장치의 식별자를 제공하는 단계;

상기 식별자에 기초하여 상기 장치가 상기 제1 서비스에 대한 액세스를 요청하는 것을 허가하는 단계;

상기 장치가 상기 제1 서비스가 위치하는 제1 호스트 서버로부터 상기 제1 서비스에 액세스하게 하는 단계- 상기 액세스는 상기 요청 서버를 통해 이뤄짐 -

를 포함하는 액세스 방법.

청구항 62

제61항에 있어서,

상기 허가하는 단계는,

상기 식별자에 기초하여 상기 사용자 장치가 상기 제1 서비스에 액세스하도록 허가되는지를 확인하는 단계

를 포함하는 액세스 방법.

청구항 63

제62항에 있어서,

확인하는 단계는,

상기 식별자에 기초하여 상기 사용자가 적어도 하나의 기준(criteria)을 만족하는지 확인하는 단계

를 포함하는 액세스 방법.

청구항 64

제63항에 있어서,

제1 기준이 상기 제1 호스트 서버 또는 상기 요청 서버에 저장되고, 제2 기준이 다른 서버에 위치하는 액세스 방법.

청구항 65

제61항 내지 제64항 중 어느 한 항에 있어서,

상기 허가하는 단계는,

상기 요청 서버 및 상기 제1 호스트 서버간의 통신에 대한 서명을 검증하는 단계

를 포함하는 액세스 방법.

청구항 66

제61항 내지 제65항 중 어느 한 항에 있어서,

상기 허가하는 단계는 상기 요청 서버에서 수행되는 액세스 방법.

청구항 67

제66항에 있어서,

상기 허가하는 단계는,

상기 요청 서버에서 상기 장치가 상기 제1 서비스에 액세스하도록 이전에 허가되었는지를 결정하는 단계

를 포함하는 액세스 방법.

청구항 68

제61항 내지 제65항 중 어느 한 항에 있어서,

상기 허가하는 단계는 디렉토리 서버에서 수행되는 액세스 방법.

청구항 69

제68항에 있어서,

상기 허가하는 단계는,

상기 요청 서버가 상기 디렉토리 서버로부터 상기 장치에 대한 허가를 요청하는 단계

를 포함하는 액세스 방법.

청구항 70

제68항 또는 제69항에 있어서,

상기 액세스하게 하는 단계는,

상기 디렉토리 서버가 상기 제1 호스트 서버에 대한 식별자를 상기 요청 서버에 전송하는 단계

를 포함하는 액세스 방법.

청구항 71

제68항 내지 제70항 중 어느 한 항에 있어서,

상기 식별자를 허가하는 데이터는 상기 디렉토리 서버에 저장되는 액세스 방법.

청구항 72

제61항 내지 제71항 중 어느 한 항에 있어서,

제2 서비스에 대한 액세스를 요청하는 단계;

상기 식별자에 기초하여 상기 장치가 상기 제2 서비스에 액세스하는 것을 허가하는 단계;

상기 장치가 상기 요청 서버를 통해 상기 제2 서비스에 액세스하게 하는 단계

를 더 포함하는 액세스 방법.

청구항 73

제72항에 있어서,

상기 제2 서비스는 상기 제1 호스트 서버에 위치하는 액세스 방법.

청구항 74

제72항에 있어서,

상기 제2 서비스는 제2 호스트 서버에 위치하는 액세스 방법.

청구항 75

제72항 내지 제74항 중 어느 한 항에 있어서,

상기 장치가 상기 제1 서비스에 액세스하는 것을 허가하는 단계는 제1 디렉토리 서버에서 수행되고,

상기 사용자 장치가 상기 제2 서비스에 액세스하는 것을 허가하는 단계는 제2 디렉토리 서버에서 수행되는 액세스 방법.

청구항 76

제72항 내지 제75항 중 어느 한 항에 있어서,

제3 서비스에 대한 액세스를 요청하는 단계;

상기 식별자에 기초하여 상기 장치가 상기 제3 서비스에 액세스하는 것을 허가하는 단계;

상기 장치가 상기 제3 서비스에 액세스하게 하는 단계

를 더 포함하는 액세스 방법.

청구항 77

제76항에 있어서,

상기 제2 서비스는 상기 제1 호스트 서버, 상기 제2 호스트 서버 또는 제3 호스트 서버에 위치하는 액세스 방법.

청구항 78

제76항 또는 제77항에 있어서,

상기 장치가 상기 제3 서비스에 액세스하는 것을 허가하는 단계는 제3 디렉토리 서버에서 수행되는 액세스 방법.

청구항 79

제61항 내지 제78항 중 어느 한 항에 있어서,

식별자를 제공하는 단계는,

상기 장치가 암호화된 터널(encrypted tunnel)을 통해 상기 요청 서버와 통신하는 단계

를 포함하는 액세스 방법.

청구항 80

제61항 내지 제79항 중 어느 한 항에 있어서,

각각의 개별 서버에서 수신되는 데이터를 캐싱하는 단계

를 더 포함하는 액세스 방법.

청구항 81

제61항 내지 제80항 중 어느 한 항에 있어서,

각각의 호스트 서버는 둘 이상의 서비스를 제공하는 액세스 방법.

청구항 82

제61항 내지 제81항 중 어느 한 항의 방법을 수행하는 장치.

청구항 83

제82항에 있어서,

상기 장치는 개인용 컴퓨터, 스마트 폰, 스마트 태블릿 또는 사물 인터넷이 가능한 장치 중에서 적어도 하나를

포함하는 장치.

청구항 84

실행될 때 컴퓨터 장치가 제61항 내지 제81항 중 어느 한 항의 방법을 수행하게 하는 코드 부분들(code portions)을 포함하는 컴퓨터 판독 가능 매체.

청구항 85

제1 데이터 저장소로부터 제2 데이터 저장소로 제1 데이터를 스위칭하기 위한 요청을 제공하는 단계;

상기 요청에 포함된 식별자에 기초하여 상기 제1 데이터 저장소의 식별자를 디렉토리 서버로부터 결정하는 단계;

상기 제1 데이터 저장소로부터 상기 제2 데이터 저장소로 상기 제1 데이터를 마이그레이션하는 단계

를 포함하는 데이터 마이그레이션 방법.

청구항 86

제85항에 있어서,

상기 마이그레이션하는 단계는, 상기 디렉토리 서버에서;

상기 제2 데이터 저장소에서 상기 데이터에 대한 시작 타임스탬프(start timestamp)를 할당하는 단계; 및

상기 제1 데이터 저장소에서 상기 데이터에 대한 종료 타임스탬프(end timestamp)를 할당하는 단계

를 포함하는 데이터 마이그레이션 방법.

청구항 87

제86항에 있어서,

상기 종료 타임스탬프 이후에 상기 제1 데이터 저장소를 통해 상기 데이터에 액세스하려고 시도하는 요청 서버(requesting server)에 상기 디렉토리 서버를 통해 상기 제2 데이터 저장소에서 상기 사용자를 검색하도록 지시하는 단계

를 더 포함하는 데이터 마이그레이션 방법.

청구항 88

제85항 내지 제87항 중 어느 한 항에 있어서,

상기 제1 데이터 저장소에서의 상기 데이터는 제1 계정 제공자와의 제1 계정 등록을 포함하고,

상기 제2 데이터 저장소에서의 상기 데이터는 새로운 계정 제공자와의 제2 계정 등록을 포함하는 데이터 마이그레이션 방법.

청구항 89

제88항에 있어서,

상기 마이그레이션하는 단계는,

상기 현재 계정 제공자로부터 상기 새로운 계정 제공자로 상기 제1 계정 등록에 관한 정보를 전송하는 단계

를 포함하는 데이터 마이그레이션 방법.

청구항 90

제89항에 있어서,

상기 정보는 등록들(registrations), 잔액들(balances), 컨피규레이션들(configurations) 및/또는 결제 지시들(payment instructions) 중에서 적어도 하나를 포함하는 데이터 마이그레이션 방법.

청구항 91

제88항 내지 제90항 중 어느 한 항에 있어서,

마이그레이션하는 단계는,

상기 제1 등록이 상기 현재 계정 제공자로부터 상기 새로운 계정 제공자로 스위치되어야 함을 나타내는 인증 코드(authentication code)를 확인하는 단계

를 포함하는 데이터 마이그레이션 방법.

청구항 92

제88항 내지 제91항 중 어느 한 항에 있어서,

상기 제1 계정 등록은 제1 사용자 크리덴셜을 포함하고,

상기 제2 계정 등록은 제2 사용자 크리덴셜을 포함하는 데이터 마이그레이션 방법.

청구항 93

제92항에 있어서,

상기 제1 사용자 크리덴셜은 제1 서버에 등록되고,

상기 제2 사용자 크리덴셜은 제2 서버에 등록되는 데이터 마이그레이션 방법.

청구항 94

제93항에 있어서,

상기 제1 계정 제공자에 의해 상기 제1 사용자 크리덴셜을 이용하여 사용자에게 전달되는 통신을 수신하는 단계;

상기 제2 사용자 크리덴셜을 이용하여 상기 통신을 상기 제2 계정 제공자로 라우팅하는 단계

를 더 포함하는 데이터 마이그레이션 방법.

청구항 95

제93항 또는 제94항에 있어서,

상기 제1 크리덴셜을 사용하는 상기 제1 등록 제공자로 만들어진 데이터 트랜잭션을 상기 제2 사용자 크리덴셜을 사용하는 상기 제2 등록 제공자로 반전시키는 단계

를 더 포함하는 데이터 마이그레이션 방법.

청구항 96

제95항에 있어서,

상기 데이터 트랜잰션시 상기 사용자가 상기 제1 사용자 크리덴셜을 사용했다는 것을 결정하는 단계

를 포함하는 데이터 마이그레이션 방법.

청구항 97

제94항 내지 제96항 중 어느 한 항에 있어서,

상기 통신을 전송하는 서버는 상기 제2 사용자 크리덴셜에 액세스하도록 공인되어야 하는 데이터 마이그레이션 방법.

청구항 98

제92항 내지 제97항 중 어느 한 항에 있어서,

상기 제1 사용자 크리덴셜 및 상기 제2 사용자 크리덴셜은 동일한 데이터 마이그레이션 방법.

청구항 99

제85항 내지 제98항 중 어느 한 항의 방법을 수행하는 장치.

청구항 100

제99항에 있어서,

상기 장치는 개인용 컴퓨터, 스마트 폰, 스마트 태블릿 또는 사물 인터넷이 가능한 장치 중에서 적어도 하나를 포함하는 장치.

청구항 101

실행될 때 컴퓨터 장치가 제85항 내지 제98항 중 어느 한 항의 방법을 수행하게 하는 코드 부분들(code portions)을 포함하는 컴퓨터 판독 가능 매체.

청구항 102

제1 엔티티로부터 제2 엔티티로 제1 통신- 상기 제1 통신은 두 개 이상의 데이터 필드들을 포함하고, 각각의 필드는 개별 라벨(respective label)을 포함함 -을 전송하는 단계; 및

상기 제1 엔티티로부터 상기 제2 엔티티로 제2 통신- 상기 제2 통신은 상기 두 개 이상의 데이터 필드들을 포함하고, 상기 제2 통신에서의 상기 필드들의 순서는 상기 제1 통신에서의 상기 필드들의 순서와 다름 -을 전송하는 단계

를 포함하는 통신 방법.

청구항 103

제102항에 있어서,

랜덤 필드를 상기 제2 통신에 추가하는 단계

를 더 포함하는 통신 방법.

청구항 104

제102항 또는 제103항에 있어서,

각각의 필드는 두 개 이상의 특징들을 포함하고,

적어도 하나의 필드에서 특징들의 케이스들을 믹싱하는 단계

를 더 포함하는 통신 방법.

청구항 105

제102항 내지 제104항 중 어느 한 항에 있어서,

상기 제2 통신을 처리하기 전에 상기 제2 엔티티에 의해 상기 제2 통신에서 상기 필드들을 해독 및 순서화하는 단계

를 더 포함하는 통신 방법.

청구항 106

제105항에 있어서,

상기 제2 엔티티에 의해 처리할 수 없는 필드들을 폐기하는 단계

를 더 포함하는 통신 방법.

청구항 107

제102항 내지 제106항 중 어느 한 항에 있어서,

상기 제1 엔티티 및 상기 제2 엔티티 중에서 적어도 하나는 서버를 포함하는 장치.

청구항 108

제102항 내지 제106항 중 어느 한 항에 있어서,

상기 제1 엔티티 및 상기 제2 엔티티 중에서 적어도 하나는 개인용 컴퓨터, 스마트 폰, 스마트 태블릿 또는 사물 인터넷이 가능한 장치를 포함하는 장치.

청구항 109

제102항 내지 제108항 중 어느 한 항의 방법을 수행하는 장치.

청구항 110

제109항에 있어서,

상기 장치는 개인용 컴퓨터, 스마트 폰, 스마트 태블릿 또는 사물 인터넷이 가능한 장치 중에서 적어도 하나를 포함하는 장치.

청구항 111

실행될 때 컴퓨터 장치가 제102항 내지 제108항 중 어느 한 항의 방법을 수행하게 하는 코드 부분들(code portions)을 포함하는 컴퓨터 판독 가능 매체.

청구항 112

USSD(unstructured supplementary service data)를 통한 통신 방법에 있어서,

제1 장치와 제2 장치간의 USSD 세션을 개방하는 단계;

상기 제1 장치에서 상기 세션에서 통신에 대한 사이퍼 텍스트(cypher text)를 생성하는 단계;

상기 제1 장치에서 상기 사이퍼 텍스트를 인코딩하는 단계;

상기 제2 장치에서 해독을 위해 상기 제1 장치로부터 상기 제2 장치로 상기 인코딩된 사이퍼 텍스트를 전송하는 단계

를 포함하는 통신 방법.

청구항 113

제112항에 있어서,

상기 인코딩하는 단계는

상기 사이퍼 텍스트를 7-비트 또는 8-비트 문자 스트링(7-bit or 8-bit character string)으로 인코딩하는 단계

를 포함하는 통신 방법.

청구항 114

제112항 또는 제113항에 있어서,

상기 사이퍼 텍스트의 상기 길이가 상기 USSD 세션에서 상기 허용된 스페이스보다 긴 경우:

상기 사이퍼 텍스트를 두 개 또는 두 개 이상의 파트들로 절단하는 단계; 및

상기 두 개 또는 두 개 이상의 파트들을 개별적으로 전송하는 단계

를 더 포함하는 통신 방법.

청구항 115

제114항에 있어서,

상기 제2 장치에서의 해독을 위해 상기 제2 장치에서 상기 전체 사이퍼 텍스트로 상기 파트들을 리어셈블링하는 단계

를 포함하는 통신 방법.

청구항 116

제112항 내지 제115항 중 어느 한 항에 있어서,

상기 제1 및 제2 장치들을 인증하는 단계

를 더 포함하는 통신 방법.

청구항 117

제116항에 있어서,

인증하는 단계는,

두 개의 통신 컴퓨터 어플리케이션들 간의 프라이버시 및 데이터 무결성을 제공하는 알고리즘을 사용하는 단계

를 포함하는 통신 방법.

청구항 118

제117항에 있어서,

인증하는 단계는,

TLS(transport layer security)를 사용하는 단계

를 포함하는 통신 방법.

청구항 119

제118항에 있어서,

TLS를 사용하는 단계는,

제1 세션 키를 생성하는 단계

를 포함하는 통신 방법.

청구항 120

제119항에 있어서,

제2 세션 키를 생성하기 위해 PAKE 프로토콜 협상(PAKE protocol negotiation)을 암호화하는 상기 제1 세션 키를 사용하는 단계; 및

상기 제2 세션 키를 사용하여 상기 제1 장치와 상기 제2 장치 간의 상기 세션에서 추가 통신들을 암호화하는 단계

를 더 포함하는 통신 방법.

청구항 121

제112항 내지 제120항 중 어느 한 항의 방법을 수행하는 장치.

청구항 122

실행될 때 컴퓨터 장치가 제112항 내지 제120항 중 어느 한 항의 방법을 수행하게 하는 코드 부분들(code portions)을 포함하는 컴퓨터 판독 가능 매체.

청구항 123

제1 엔티티와 관련된 제1 장치와 제2 엔티티와 관련된 제2 장치 간의 통신 방법에 있어서, 상기 제1 장치에서:

제1 공유 비밀(first shared secret)을 사용하여 상기 제1 장치 및 상기 제2 장치 간의 제1 PAKE 세션을 생성하는 단계;

상기 제2 장치로부터 등록 키 및 제2 공유 비밀(second shared secret)을 수신하는 단계;

제2 PAKE 세션을 생성하기 위한 제3 공유 비밀(third shared secret)을 제공하기 위해서 상기 제1 공유 비밀, 상기 등록 키, 및 상기 제2 공유 비밀을 해싱하는 단계

를 포함하는 통신 방법.

청구항 124

제123항에 있어서,

상기 제1 엔티티 및 상기 제2 엔티티를 인증하는 단계

를 더 포함하는 통신 방법.

청구항 125

제124항에 있어서,

인증하는 단계는,

두 개의 통신 컴퓨터 어플리케이션들 간의 프라이버시 및 데이터 무결성을 제공하는 알고리즘을 사용하는 단계

를 포함하는 통신 방법.

청구항 126

제125항에 있어서,

상기 인증하는 단계는,

TLS를 사용하는 단계

를 포함하는 통신 방법.

청구항 127

제123항 내지 제126항 중 어느 한 항에 있어서,

제4 공유 비밀(forth shared secret)을 사용하여 상기 제1 장치 및 제3 장치 간의 제2 PAKE 세션을 생성하는 단계

를 더 포함하는 통신 방법.

청구항 128

제127항에 있어서,

상기 제4 공유 비밀은 상기 제1 장치를 위해 상기 제3 장치에 의해 생성된 인증 코드를 포함하는 통신 방법.

청구항 129

제123항 내지 제128항 중 어느 한 항에 있어서,

상기 제1 공유 비밀은 상기 제1 장치를 위해 상기 제2 장치에 의해 생성된 인증 코드를 포함하는 통신 방법.

청구항 130

제129항에 있어서,

상기 인증 코드는 상기 제1 장치를 위해 식별자와 함께 상기 제1 장치로 전송되는 통신 방법.

청구항 131

제130항에 있어서,

상기 식별자는 상기 제1 장치의 전화 번호 또는 일련 번호를 포함하는 통신 방법.

청구항 132

제123항 내지 제131항 중 어느 한 항에 있어서,

상기 제1 공유 비밀은 상기 제1 엔티티와 연관된 은행 카드의 PAN(personal account number)을 포함하는 통신 방법.

청구항 133

제123항 내지 제131항 중 어느 한 항에 있어서,

상기 제1 공유 비밀은 상기 제1 엔티티와 연관된 은행 카드의 인코딩된 일련 번호를 포함하는 통신 방법.

청구항 134

제123항 내지 제133항 중 어느 한 항의 방법을 수행하는 장치.

청구항 135

제134항에 있어서,

상기 장치는 개인용 컴퓨터, 스마트 폰, 스마트 태블릿 또는 사물 인터넷이 가능한 장치 중에서 적어도 하나를 포함하는 장치.

청구항 136

실행될 때 컴퓨터 장치가 제123항 내지 제133항 중 어느 한 항의 방법을 수행하게 하는 코드 부분들(code portions)을 포함하는 컴퓨터 판독 가능 매체.

청구항 137

서비스에 액세스하는 방법에 있어서,

크리덴셜 및 상기 크리덴셜에 대한 컨텍스트를 제공하는 단계;

상기 크리덴셜 및 상기 컨텍스트에 기초하여 상기 서비스에 대한 액세스를 인증하는 단계

를 포함하는 액세스 방법.

청구항 138

제137항에 있어서,

상기 서비스에 대한 액세스를 인증하는 단계는,

상기 크리덴셜 및/또는 상기 컨텍스트에 기초하여 서비스의 일부에 대한 액세스를 인증하는 단계

를 포함하는 액세스 방법.

청구항 139

제137항 또는 제138항에 있어서,

상기 크리덴셜은 장치 및 상기 장치의 프라이머리 사용자(primary user)와 관련된 제1 크리덴셜을 포함하는 액세스 방법.

청구항 140

제139항에 있어서,

상기 크리덴셜은 장치 및 상기 장치의 세컨더리 사용자(secondary user)와 관련된 제2 크리덴셜을 더 포함하는 액세스 방법.

청구항 141

제140항에 있어서,

상기 크리덴셜에 기초하여 상기 서비스에 대한 액세스를 인증하는 단계는,

상기 제1 크리덴셜 및 상기 제2 크리덴셜 각각에 기초하여 상기 프라이머리 사용자 및 상기 세컨더리 사용자에 대한 상이한 서비스들에 대한 액세스를 인증하는 단계

를 포함하는 액세스 방법.

청구항 142

제141항에 있어서,

상기 장치는 상기 프라이머리 사용자 및 상기 세컨더리 사용자에 대한 상이한 지출 한도인 상기 상이한 서비스

들 및 은행 카드를 포함하는 액세스 방법.

청구항 143

제137항 내지 제142항 중 어느 한 항에 있어서,

상기 크리덴셜은 상기 컨텍스트에 기초하여 선택되는 액세스 방법.

청구항 144

제137항 내지 제143항 중 어느 한 항에 있어서,

상기 서비스는 상기 컨텍스트에 기초하여 선택된 복수의 서비스들을 포함하는 액세스 방법.

청구항 145

제137항 내지 제144항 중 어느 한 항에 있어서,

관리자 또는 사용자는 상기 컨텍스트 또는 크리덴셜을 수정, 추가 또는 취소할 수 있는 액세스 방법.

청구항 146

제137항 내지 제145항 중 어느 한 항에 있어서,

상기 크리덴셜은 패스워드, PIN, 및/또는 다른 직접 인증 크리덴셜(direct authentication credential) 중에서 적어도 하나를 포함하는 액세스 방법.

청구항 147

제137항 내지 제146항 중 어느 한 항에 있어서,

상기 컨텍스트는 상기 크리덴셜을 제공하는 장치, 상기 장치상의 어플리케이션, 상기 장치가 연결된 네트워크, 상기 장치의 지리적 위치, 및/또는 액세스되는 상기 서비스 중에서 적어도 하나를 포함하는 액세스 방법.

청구항 148

제137항 내지 제147항 중 어느 한 항의 방법을 수행하는 장치.

청구항 149

제148항에 있어서,

상기 장치는 개인용 컴퓨터, 스마트 폰, 스마트 태블릿 또는 사물 인터넷이 가능한 장치 중에서 적어도 하나를 포함하는 장치.

청구항 150

실행될 때 컴퓨터 장치가 제137항 내지 제147항 중 어느 한 항의 방법을 수행하게 하는 코드 부분들(code portions)을 포함하는 컴퓨터 판독 가능 매체.

청구항 151

컴퓨터 시스템내 모듈들 간의 통신 방법에 있어서,

제1 모듈로부터 프록시로 공유 메모리 채널을 전달하는 단계;

상기 프록시로부터 제2 모듈로 상기 공유 메모리 채널을 전달하는 단계- 상기 프록시는 상기 컴퓨터 시스템의 상기 커널을 바이패스하여 상기 제1 모듈 및 상기 제2 모듈 간의 데이터를 전송하는 핸드-오프 모듈을 포함함-;

상기 제1 모듈로부터 상기 제2 모듈로 데이터를 전송하는 단계

를 포함하는 통신 방법.

청구항 152

제151항에 있어서,

복수의 요청들을 상기 제1 모듈의 버퍼 메모리에서 배치된 메시지(batched message)로 배칭하는(batching) 단계;

상기 제2 모듈로 전송되는 상기 배치된 메시지를 큐잉하는 단계;

시스템 기능을 허가하는 적어도 하나의 시스템 플래그를 셋팅하는 단계;

상기 제2 모듈에서 상기 적어도 하나의 시스템 플래그를 체크하는 단계; 및

상기 제2 모듈에서 상기 배치된 메시지를 처리하는 단계

를 더 포함하는 통신 방법.

청구항 153

제151항 또는 제152항에 있어서,

상기 제1 모듈 및 상기 제2 모듈 간의 적어도 하나의 공유 메모리 채널을 설정하는 단계

를 더 포함하는 통신 방법.

청구항 154

제153항에 있어서,

상기 적어도 하나의 공유 메모리 채널을 통해 상기 제1 모듈에 응답하는 상기 제2 모듈을 포함하는 통신 방법.

청구항 155

제153항 또는 제154항에 있어서,

상기 적어도 하나의 공유 메모리 채널은 상기 배치된 메시지를 수신 및 어셈블링하고, 상기 제2 모듈로 상기 메모리의 소유권을 넘겨주는 통신 방법.

청구항 156

제155항에 있어서,

상기 적어도 하나의 공유 메모리 채널은 상기 컴퓨터 시스템의 네트워크 스택(network stack)을 통해 배치된 메시지를 수신하는 통신 방법.

청구항 157

제153항 내지 제156항 중 어느 한 항에 있어서,

상기 적어도 하나의 공유 메모리 채널은 HTTP 게이트웨이를 포함하는 통신 방법.

청구항 158

제151항 내지 제157항 중 어느 한 항에 있어서,

HTTP 게이트웨이는 웹 서비스로써 사용되는 통신 방법.

청구항 159

제151항 내지 제158항 중 어느 한 항에 있어서,

통신은 패스워드 인증된 키 교환 프로토콜(password authenticated key exchange protocol)을 사용하는 통신 방법.

청구항 160

제151항 내지 제159항 중 어느 한 항에 있어서,

상기 컴퓨터 시스템의 네트워크 스택에서 제로-카피 네트워킹(zero-copy networking)을 사용하는 단계

를 더 포함하는 통신 방법.

청구항 161

제151항 내지 제160항 중 어느 한 항에 있어서,

상기 컴퓨터 시스템의 네트워크 스택에서 사용자-모드 네트워킹(user-mode networking)을 사용하는 단계

를 더 포함하는 통신 방법.

청구항 162

제151항 내지 제161항 중 어느 한 항에 있어서,

상기 제1 모듈로부터 상기 데이터 전송(data transmission)의 상기 컴포넌트들이 단일 데이터 스트림(single data stream)으로 결합되고, 상기 제1 모듈에서 상기 컴포넌트들로 분리되도록 데이터를 직렬화하는 단계

를 더 포함하는 통신 방법.

청구항 163

제162항에 있어서,

상기 직렬화는 각 모듈의 에지에서 추상화되는 통신 방법.

청구항 164

제151항 내지 제163항 중 어느 한 항에 있어서,

각 모듈의 버퍼 메모리는 구성 가능한 버퍼링 임계값(a configurable threshold of buffering)을 갖는 통신 방법.

청구항 165

제151항 내지 제164항 중 어느 한 항에 있어서,

상기 제1 모듈 및 상기 제2 모듈은 동일한 컴퓨팅 장치 상에 위치하는 통신 방법.

청구항 166

제151항 내지 제164항 중 어느 한 항에 있어서,

상기 제1 모듈 및 상기 제2 모듈은 상이한 컴퓨팅 장치들 상에 위치하는 통신 방법.

청구항 167

제151항 내지 제166항 중 어느 한 항에 있어서,

상기 제1 모듈로부터 상기 제2 모듈로 전송된 데이터는 버전 ID(version ID)를 운반하는 통신 방법.

청구항 168

제167항에 있어서,

상기 버전 ID가 상기 제1 모듈로부터 상기 제2 모듈로 전송된 상기 데이터에 대해 최신인지를 검증하는 단계를 더 포함하는 통신 방법.

청구항 169

제168항에 있어서,

상기 데이터 중에서 임의의 데이터가 업데이트되는 경우, 상기 버전 ID를 현재 버전으로 재검증하는 단계를 더 포함하는 통신 방법.

청구항 170

제169항에 있어서,

상기 버전 ID가 검증되지 않는 경우, 상기 데이터 전송은 실패하는 통신 방법.

청구항 171

제151항 내지 제170항 중 어느 한 항에 있어서,

상기 제1 모듈 및 상기 제2 모듈 중에서 적어도 하나는 적어도 하나의 데이터 서비스 모듈을 포함하고,

상기 컴퓨터 시스템 내의 각각의 데이터 활동은 상기 적어도 하나의 데이터 서비스 모듈을 통해 실행되는 통신 방법.

청구항 172

제171항에 있어서,

상기 적어도 하나의 데이터 서비스 모듈은 코어 데이터베이스 저장소(core database store)에 의해 구현되는 데이터 저장소와 통신하는 통신 방법.

청구항 173

제172항에 있어서,

상기 적어도 하나의 데이터 서비스 모듈은 상기 데이터 저장소에 직접 액세스하는 상기 컴퓨터 시스템의 컴포넌트인 통신 방법.

청구항 174

제173항에 있어서,

상기 코어 데이터베이스 저장소는 적어도 하나의 분산 데이터베이스(distributed database)를 포함하는 통신 방법.

청구항 175

제174항에 있어서,

상기 적어도 하나의 분산 데이터베이스는 별도의 판독(read) 및 기록(write) 액세스 채널들을 갖는 통신 방법.

청구항 176

제173항 내지 제175항 중 어느 한 항에 있어서,

상기 데이터 저장소는 적어도 하나의 이종 데이터베이스(heterogeneous database)에 인터페이스를 제공하는 통신 방법.

청구항 177

제173항 내지 제176항 중 어느 한 항에 있어서,

상기 데이터 저장소는 복수의 인터페이스 타입들을 제공하는 통신 방법.

청구항 178

제177항에 있어서,

상기 복수의 인터페이스 타입들은 적어도 하나의 SQL(Structured Query Language) 인터페이스, 셀 및 칼럼 인터

페이스(cell and column interface), 문서 인터페이스(document interface), 및 상기 코어 데이터베이스 저장소 위에 있는 그래픽 인터페이스(graph interface) 중에서 적어도 하나를 포함하는 통신 방법.

청구항 179

제176항 내지 제178항 중 어느 한 항에 있어서,

상기 데이터 저장소 레이어에 대한 모든 기록들은 하나 또는 하나 이상의 데이터 트랜잭션들의 전부 또는 일부를 제어하는 단일 공유 모듈에 의해 관리되는 통신 방법.

청구항 180

제179항에 있어서,

적어도 하나의 상기 공유 모듈의 리던던트 백업(redundant backup)을 작동시키는 단계

를 더 포함하는 통신 방법.

청구항 181

제179항 또는 제180항에 있어서,

모든 데이터 변경은 시리얼 빠른 시퀀스(serial rapid sequence)로 상기 단일 공유 모듈을 통해 진행되는 통신 방법.

청구항 182

제179항 내지 제181항 중 어느 한 항에 있어서,

상기 단일 공유 모듈은 그 자체를 데이터 트랜잭터 클러스터(data transactor cluster)로 나타내는 핫 백업 리던던시 모델(hot backup redundancy model)을 사용하고,

상기 데이터 트랜잭터 클러스터는 하이어라키(hierarchy)에서 모듈들의 세트이고, 각 모듈은 마스터 모듈(master module)이 실패하는 경우, 데이터 트랜잭션들을 제어하는 통신 방법.

청구항 183

제171항 내지 제182항 중 어느 한 항에 있어서,

도메인에 의해 구성되는 규칙들에 기초하여 모듈들 또는 데이터 저장소들에 걸쳐 데이터를 분할하는 단계

를 더 포함하는 통신 방법.

청구항 184

제183항에 있어서,

데이터 트랜잭션의 레코드 또는 부모 데이터 트랜잭션(parent data transaction)의 레코드의 타겟 데이터를 해싱하는 단계

를 더 포함하는 통신 방법.

청구항 185

제184항에 있어서,

상기 해싱하는 단계는 데이터 파티션들의 수와 동일한 카디널리티(cardinality)를 갖는 통신 방법.

청구항 186

제184항 또는 제185항에 있어서,

열거된 지리적 영역(enumerated geographical area), 성(last name) 및/또는 통화(currency) 중에서 적어도 하나에 의해 타겟 데이터를 해싱하는 단계

를 더 포함하는 통신 방법.

청구항 187

제171항 내지 제186항 중 어느 한 항에 있어서,

다중 데이터 파티션들에 걸쳐 상기 적어도 하나의 데이터 서비스 모듈을 통해 적어도 하나의 데이터 전송을 수행하는 단계

를 더 포함하는 통신 방법.

청구항 188

제171항 내지 제187항 중 어느 한 항에 있어서,

다중 모듈들에 의해 상기 적어도 하나의 데이터 서비스 모듈을 통해 적어도 하나의 데이터 전송을 완료하는 단계

를 더 포함하는 통신 방법.

청구항 189

제171항 내지 제188항 중 어느 한 항에 있어서,

상기 적어도 하나의 데이터 서비스 모듈 상의 적어도 하나의 데이터 전송을 상기 데이터 저장소에서 다중 데이터 스토리지 노드들 상에 유지하는 단계

를 더 포함하는 통신 방법.

청구항 190

제171항 내지 제189항 중 어느 한 항에 있어서,

상기 컴퓨터 시스템은 복수의 데이터 서비스 모듈들을 포함하고,

각각의 데이터 서비스 모듈은 해당 인스턴스에 대한 모든 상기 핫 데이터의 캐시된 표현을 포함하여 인-메모리(in-memory)/인-프로세스(in-process) 데이터베이스 엔진을 호스트하는 통신 방법.

청구항 191

제171항 내지 제189항 중 어느 한 항에 있어서,

상기 컴퓨터 시스템은 복수의 데이터 서비스 모듈들을 포함하고,

각각의 데이터 서비스 모듈은 복수의 이종 또는 동종 데이터베이스 엔진들을 포함하는 통신 방법.

청구항 192

제172항 내지 제191항 중 어느 한 항에 있어서,

모든 데이터 판독들이 일관되고, 대응하는 데이터 기록들을 반영하도록, 상기 데이터 저장소에 대한 액세스의 동시성을 관리하는 MVCC(Multiversion Concurrency Control) 버전 시스템을 사용하는 단계

를 더 포함하는 통신 방법.

청구항 193

제172항 내지 제191항 중 어느 한 항에 있어서,

데이터 레코드가 상기 데이터 저장소에 기록되어야 하고, 임의의 후속 데이터 트랜잭션이 상기 데이터 레코드에 액세스하기 전에 기록된 것으로 확인되어야 하도록, 상기 데이터 저장소에 대한 액세스의 동시성을 관리하는 비관적 일관성(pessimistic consistency)을 사용하는 단계

를 더 포함하는 통신 방법.

청구항 194

제171항 내지 제193항 중 어느 한 항에 있어서,

상기 컴퓨터 시스템은 어플리케이션 레이어를 더 포함하고,

상기 적어도 하나의 데이터 서비스 모듈이 상기 레코드를 기록하고, 상기 데이터 전송을 완료함을 확인할 때까지, 상기 어플리케이션 레이어는 데이터 트랜잭션을 진행할 수 없는 통신 방법.

청구항 195

제151항 내지 제194항 중 어느 한 항의 방법을 수행하는 컴퓨팅 장치.

청구항 196

실행될 때 컴퓨터 장치가 제151항 내지 제194항 중 어느 한 항의 방법을 수행하게 하는 코드 부분들(code portions)을 포함하는 컴퓨터 판독 가능 매체.

발명의 설명

기 술 분 야

[0001] 본 발명은 단일 구현에서 모든 타입들의 트랜잭션들을 대규모로 안전하게 거의 실시간으로 수행하는 방법들 및 시스템들에 관한 것이다.

배 경 기 술

[0002] 트랜잭션 처리는 광범위한 분산 컴퓨터 기반 시스템 및 특히, 결제들(payments)에 관한 트랜잭션들을 수행하는 다중 트랜잭터들(transactors)을 포함할 뿐만 아니라, 다른 금융 자산들 및 상품들(financial assets and instruments), 물리적인 액세스 제어, 데이터에 대한 로지컬 액세스, IoT(Internet of Things)를 구성하는 관

리 및 모니터링 장치들 등에서의 트레이드(trade)와 관련이 있다.

[0003] 최근에 트랜잭션 처리 시스템들을 개발할 때, 엔지니어들은 어려운 트레이드 오프들(trade-offs)을 수행해야 한다. 이것들은 속도 및 탄성(resilience), 처리량(throughput) 및 일관성(consistency), 보안(security) 및 성능(performance), 일관성(consistency) 및 확장성(scalability) 등등 간의 선택을 포함한다. 이러한 트레이드 오프들은 언제나 전체 시스템에 영향을 미치는 타협안(compromises)을 발생 시킨다. 결제 처리 시스템들(Payment processing systems)은 이러한 트레이드 오프들의 효과를 나타낸다. 그것들은 초당 600 및 수만 개의 트랜잭션들을 처리해야 할 수도 있지만, 오직 시스템의 업무량(workload)에서 잠시 동안 추가적인 처리를 위해 그것들을 부분 처리하고, 디테일들을 저장하는 것 만을 수행할 수 있다. 이것은 종종 누락 레코드들을 조정하고, 트랜잭션들을 복제하고, 트랜잭션 시간 및 트랜잭션 처리 시간 사이에 계정들이 초과 인출되는 신용 문제들에 대한 노출 등의 문제들을 발생 시킨다. 그러나, 문제는 페이먼트들로 제안되지 않는다.

[0004] 전체 트랜잭션이 롤백(rolled back)되고(원자성), 데이터베이스를 일관성이 없는 상태로 둘 수 없고(일관성), 서로 간섭할 수 없으며(절연), 심지어 서버들이 다시 시작할 때에도 지속되는(내구성) 경우, ACID(원자성(atomicity), 일관성(consistency), 절연(isolation) 및 내구성(durability))는 각 데이터베이스 트랜잭션이 성공해야 한다는 데이터베이스들에 대한 일관성 모델이다.

[0005] 이 모델은 일반적으로 기존 뱅킹 결제 네트워크들 및 기타 '빅 데이터' 거래 시스템들과 같은 대규모 시스템들의 가용성 및 성능 요구 사항들과 호환되지 않는 것으로 여겨진다. 대신에, 이러한 시스템들은 BASE 일관성(기본 가용성(basic availability), 소프트-상태(soft-state) 및 최종 일관성(eventual consistency))에 의존한다. 이 모델은 데이터베이스가 궁극적으로 일관성있는 상태에 도달하기에 충분하다고 주장한다. 뱅킹 시스템들은 일관성있는 상태에 도달하기 위해 종종 조정 확인들(reconciliation chechs)을 실행하고, 임의의 트랜잭션 처리를 일시 중지해야 하기에, 이 모드에서 작동한다. 대용량 트랜잭션 처리에서 트레이드 오프들이 수행되어야 한다는 이 개념은 기본적인 형태에서 분산 컴퓨터 시스템이 일관성, 가용성 및 파티션 내성(partition tolerance) 세가지 모두를 동시에 제공하는 것이 불가능하다는 점을 나타내는 CAP 정리에 명시되어 있다. 현재 모범 사례 솔류션들(best practice solutions)은 이머징 및 현재 요구 사항들을 충족하기 위한 너무 많은 제한들 및 트레이드 오프들을 포함한다.

[0006] IoT에 의해 생성된 데이터를 조정하는 방법에 대한 문제는 엔지니어들이 네트워크들 및 트랜잭션 처리 시스템들을 구축할 때 그것들이 수행해야한다고 믿는 트레이드 오프들의 효과로 인해 발생하기 시작한다. 그 효과들 중에서 하나는 함께 사물 인터넷을 구성하는 장치들 및 서버들간의 통신에 대한 보안 부족이다. 다른 하나는 장치에 의해 수집된 데이터가 실제로 해당 장치에 의해 검출된 특정 이벤트와 관련이 있음을 보장할 수 없다는 것이다.

[0007] 또한, 클라우드 기반 정보 저장 시스템들은 이러한 트레이드 오프들의 효과를 나타내기에, 종종 궁극적인 일관성만을 보장할 수 있는 엄청난 수의 서버들 및 시스템들이 된다.

[0008] 따라서, 알려진 시스템들에서 BASE 일관성만으로 이익을 얻을 수 있는 대규모 시스템들에 ACID 일관성을 제공할 필요가 있다.

발명의 내용

해결하려는 과제

[0009] 상술한 바와 같이 본 발명은 현재 트레이드 오프들에 의해 고려하거나 제한될 필요가 없는 트랜잭션들을 처리하는 새로운 방법에 관한 것이다. 본 발명은 기존 시스템들에서 가능한 것보다 몇 배 더 큰 비율로 트랜잭션들을 실시간으로 인증 및 처리하고 이러한 트랜잭션들을 실시간으로 결제 또는 처리 및 완료하는 방법을 제공한다.

[0010] 실시간 결제(real-time settlement)는 금융 트랜잭션들에만 적용되는 것이 아니다. 이것들은 즉각적인 인증, 허가, 처리 및 완료의 일부 또는 전체로부터 이익을 얻을 수 있거나 요구되는 임의의 트랜잭션에 적용된다. 이것들은 액세스 제어(access control)로부터 레코드 유효성 검사(records validation), 레코드 및 문서 교환(records and document exchange), 명령 및 제어 지시(command and control instructions) 등에 이르기까지 다양할 수 있다.

[0011] 이 방법은 일곱가지 주요 영역들로 구성된다:

[0012] • 임의의 데이터베이스 제품에 매우 높은 스케일로 ACID 호환 트랜잭션들(ACID compliant transactions, 또는

ACID 준수 트랜잭션들)을 기록하기 위한 방법.

[0013] • 단일 실시간 세션의 경계들(bounds of a single real-time session) 내에 매우 높은 스케일로 완전한 수학적 증명으로 다중 개인 원장들(multiple private ledgers)에 걸쳐 레코드들의 인증을 전달하는 해시 체인 구현.

[0014] • 주요 확장성 문제들을 야기하는 "허브 앤드 스포크(hub and spoke)" 아키텍처를 구현하는 것이 아니라 트랜잭션 서비스 제공자들의 메쉬 네트워크를 지원하는 디렉토리 서비스.

[0015] • 머천트(또는 가맹점) 또는 사용자 장치가 무선(air) 및 하나의 트랜잭션에서 다음으로 트랜잭션을 처리하는데 사용하는 어플리케이션(또는 앱(app))을 업데이트하게 하는 확장 가능한 프레임워크(extensible framework).

[0016] • 다양하고 상이한 트랜잭션 타입들 및 공통 데이터베이스 구조(common database structure, 또는 기본 데이터베이스 구조)를 지원하는 앱들 간의 이동 행렬(translation matrix)로써 역할하는 데이터 서비스 레이어.

[0017] • 서비스 또는 장치가 서비스들 또는 기능들의 세트에 액세스하게 하는 크리덴셜들의 애드혹(ad hoc of credentials) 세트를 어셈블링 및 제안하는 방법.

[0018] • NFC(Near Field Communications) 및 USSD(Unstructured Supplementary Service Data)를 포함하는 임의의 프로토콜에서 보안 실시간 통신(secure real-time communications)을 생성하는 방법.

[0019] 본 발명의 시스템은 처리 방법들 중에서 고유하게 트랜잭션들의 수가 증가함에 따라 제로 증분 코스트(zero incremental cost)로 실시간 트랜잭션 처리 및 완료를 달성하는 방법을 제공한다.

과제의 해결 수단

[0020] 일 실시예에 따르면, 제1 엔티티와 관련된 장치에서 데이터 트랜잭션 레코딩 방법에 있어서, 제1 시드 데이터를 결정하는 단계와, 상기 제1 엔티티 및 제2 엔티티간의 제1 데이터 트랜잭션의 상기 레코드를 생성하는 단계와, 적어도 상기 제1 시드 데이터 및 상기 제1 데이터 트랜잭션의 레코드를 결합하여 제2 시드 데이터를 결정하는 단계와, 상기 제2 시드 데이터를 해싱하여 제1 해시를 생성하는 단계? 상기 제1 해시는 상기 제1 엔티티와 관련된 데이터 트랜잭션들의 히스토리를 포함함 -와, 상기 제1 데이터 트랜잭션의 상기 레코드에 대한 상기 제1 해시를 메모리에 저장하는 단계를 포함하는 데이터 트랜잭션 레코딩 방법이 제공된다. 다른 실시예에 따르면, 상기 방법을 수행하고, 제1 엔티티와 관련된 장치가 제공된다. 다른 실시예에 따르면, 실행될 때 컴퓨터 장치가 상기 방법을 수행하게 하는 코드 부분들(code portions)을 포함하는 컴퓨터 판독 가능 매체가 제공된다.

[0021] 다른 실시예에 따르면, 제1 엔티티와 관련된 장치로부터 제1 해시를 수신하고- 상기 제1 해시는 상기 제1 엔티티에 관련된 데이터 트랜잭션들의 히스토리를 포함함 -, 라이선스 입력을 제공하기 위해서 라이선스 해시와 상기 제1 해시를 결합하고, 상기 라이선스 입력을 해싱하여 제2 라이선스 해시를 생성하고, 및 메모리에 상기 제2 라이선스 해시를 저장하는 라이선스 장치를 제공한다.

[0022] 다른 실시예에 따르면, 제1 엔티티와 관련된 장치로부터 제1 해시를 수신하고- 상기 제1 해시는 상기 제1 엔티티와 관련된 데이터 트랜잭션들의 히스토리를 포함함 -, 디렉토리 입력을 제공하기 위해 디렉토리 해시와 상기 제1 해시를 결합하고, 상기 라이선스 입력을 해싱하여 제2 디렉토리 해시를 생성하고, 및 메모리에 상기 제2 디렉토리 해시를 저장하는 디렉토리 장치를 제공한다.

[0023] 다른 실시예에 따르면, 장치로부터 제1 서비스에 액세스하는 방법에 있어서, 요청 서버에 상기 장치의 식별자를 제공하는 단계와, 상기 식별자에 기초하여 상기 장치가 상기 제1 서비스에 대한 액세스를 요청하는 것을 허가하는 단계와, 상기 장치가 상기 제1 서비스가 위치하는 제1 호스트 서버로부터 상기 제1 서비스에 액세스하게 하는 단계- 상기 액세스는 상기 요청 서버를 통해 이뤄짐 -를 포함하는 액세스 방법이 제공된다. 다른 실시예에 따르면, 상기 방법을 수행하는 장치가 제공된다. 다른 실시예에 따르면, 실행될 때 컴퓨터 장치가 상기 방법을 수행하게 하는 코드 부분들(code portions)을 포함하는 컴퓨터 판독 가능 매체가 제공된다.

[0024] 다른 실시예에 따르면, 제1 데이터 저장소로부터 제2 데이터 저장소로 제1 데이터를 스위칭하기 위한 요청을 제공하는 단계와, 상기 요청에 포함된 식별자에 기초하여 상기 제1 데이터 저장소의 식별자를 디렉토리 서버로부터 결정하는 단계와, 상기 제1 데이터 저장소로부터 상기 제2 데이터 저장소로 상기 제1 데이터를 마이그레이션하는 단계를 포함하는 데이터 마이그레이션 방법이 제공된다. 다른 실시예에 따르면, 상기 방법을 수행하는 장치가 제공된다. 다른 실시예에 따르면, 실행될 때 컴퓨터 장치가 상기 방법을 수행하게 하는 코드 부분들(code

portions)을 포함하는 컴퓨터 판독 가능 매체가 제공된다.

[0025] 다른 실시예에 따르면, 제1 엔티티로부터 제2 엔티티로 제1 통신- 상기 제1 통신은 두 개 이상의 데이터 필드들을 포함하고, 각각의 필드는 개별 라벨(respective label)을 포함함 -을 전송하는 단계와, 및 상기 제1 엔티티로부터 상기 제2 엔티티로 제2 통신- 상기 제2 통신은 상기 두 개 이상의 데이터 필드들을 포함하고, 상기 제2 통신에서의 상기 필드들의 순서는 상기 제1 통신에서의 상기 필드들의 순서와 다름 -을 전송하는 단계를 포함하는 통신 방법이 제공된다. 다른 실시예에 따르면, 상기 방법을 수행하는 장치가 제공된다. 다른 실시예에 따르면, 실행될 때 컴퓨터 장치가 상기 방법을 수행하게 하는 코드 부분들(code portions)을 포함하는 컴퓨터 판독 가능 매체가 제공된다.

[0026] 다른 실시예에 따르면, USSD(unstructured supplementary service data)를 통한 통신 방법에 있어서, 제1 장치와 제2 장치간의 USSD 세션을 개방하는 단계와, 상기 제1 장치에서 상기 세션에서 통신에 대한 사이퍼 텍스트(cypher text)를 생성하는 단계와, 상기 제1 장치에서 상기 사이퍼 텍스트를 인코딩하는 단계와, 상기 제2 장치에서 해독을 위해 상기 제1 장치로부터 상기 제2 장치로 상기 인코딩된 사이퍼 텍스트를 전송하는 단계를 포함하는 통신 방법이 제공된다. 다른 실시예에 따르면, 상기 방법을 수행하는 장치가 제공된다. 다른 실시예에 따르면, 실행될 때 컴퓨터 장치가 상기 방법을 수행하게 하는 코드 부분들(code portions)을 포함하는 컴퓨터 판독 가능 매체가 제공된다.

[0027] 다른 실시예에 따르면, 제1 엔티티와 관련된 제1 장치와 제2 엔티티와 관련된 제2 장치 간의 통신 방법에 있어서, 상기 제1 장치에서 제1 공유 비밀(first shared secret)을 사용하여 상기 제1 장치 및 상기 제2 장치 간의 제1 PAKE 세션을 생성하는 단계와, 상기 제2 장치로부터 등록 키 및 제2 공유 비밀(second shared secret)을 수신하는 단계와, 제2 PAKE 세션을 생성하기 위한 제3 공유 비밀(third shared secret)을 제공하기 위해서 상기 제1 공유 비밀, 상기 등록 키, 및 상기 제2 공유 비밀을 해싱하는 단계를 포함하는 통신 방법이 제공된다. 다른 실시예에 따르면, 상기 방법을 수행하는 장치가 제공된다. 다른 실시예에 따르면, 실행될 때 컴퓨터 장치가 상기 방법을 수행하게 하는 코드 부분들(code portions)을 포함하는 컴퓨터 판독 가능 매체가 제공된다.

[0028] 다른 실시예에 따르면, 서비스에 액세스하는 방법에 있어서, 크리덴셜 및 상기 크리덴셜에 대한 컨텍스트를 제공하는 단계와, 상기 크리덴셜 및 상기 컨텍스트에 기초하여 상기 서비스에 대한 액세스를 인증하는 단계를 포함하는 액세스 방법이 제공된다. 다른 실시예에 따르면, 상기 방법을 수행하는 장치가 제공된다. 다른 실시예에 따르면, 실행될 때 컴퓨터 장치가 상기 방법을 수행하게 하는 코드 부분들(code portions)을 포함하는 컴퓨터 판독 가능 매체가 제공된다.

[0029] 다른 실시예에 따르면, 컴퓨터 시스템내 모듈들 간의 통신 방법에 있어서, 제1 모듈로부터 프록시로 공유 메모리 채널을 전달하는 단계와, 상기 프록시로부터 제2 모듈로 상기 공유 메모리 채널을 전달하는 단계- 상기 프록시는 상기 컴퓨터 시스템의 상기 커널을 바이패스하여 상기 제1 모듈 및 상기 제2 모듈 간의 데이터를 전송하는 핸드-오프 모듈을 포함함 -와, 상기 제1 모듈로부터 상기 제2 모듈로 데이터를 전송하는 단계를 포함하는 통신 방법이 제공된다. 다른 실시예에 따르면, 상기 방법을 수행하는 컴퓨팅 장치가 제공된다. 다른 실시예에 따르면, 실행될 때 컴퓨터 장치가 상기 방법을 수행하게 하는 코드 부분들(code portions)을 포함하는 컴퓨터 판독 가능 매체가 제공된다.

[0030] 상기 제1 시드 데이터는 스타팅 해시(starting hash)를 포함할 수 있다. 상기 스타팅 해시는 상기 제1 엔티티와 관련된 이전 데이터 트랜잭션의 레코드를 해상한 결과일 수 있다. 상기 스타팅 해시는 랜덤 해시(random hash)를 포함할 수 있다. 상기 랜덤 해시는 상기 장치로부터의 서명, 상기 랜덤 해시가 생성된 날짜 및/또는 시간 중에서 적어도 하나를 포함할 수 있다.

[0031] 제2 시드 데이터를 제공하는 단계는 상기 제1 시드 데이터 및 상기 제1 데이터 트랜잭션의 상기 레코드와 제1 영지식 증명(first zero-knowledge proof) 및 제2 영지식 증명(second zero-knowledge proof)을 결합하는 단계를 더 포함할 수 있다. 여기서, 상기 제1 영지식 증명은 상기 스타팅 해시가 상기 제1 엔티티와 관련된 상기 이전 데이터 트랜잭션의 상기 트루 해시를 포함한다는 증명을 포함할 수 있다. 상기 제2 영지식 증명은 제2 해시가 상기 제2 엔티티와 관련된 이전 데이터 트랜잭션의 상기 트루 해시를 포함한다는 증명을 포함할 수 있다. 제2 시드 데이터를 제공하는 단계는 상기 제1 시드 데이터, 상기 제1 데이터 트랜잭션의 상기 레코드, 상기 제1 영지식 증명 및 상기 제2 영지식 증명과 제3 영지식 증명을 결합하는 단계를 더 포함할 수 있다. 상기 제3 영지식 증명은 랜덤 데이터로부터 생성될 수 있다. 상기 제3 영지식 증명은 상기 제1 영지식 증명 또는 상기 제2 영지식 증명의 반복일 수 있다. 상기 제3 영지식 증명은 상기 제2 영지식 증명에 대응하는 상기 제1 데이터 트랜잭션의 제2 레코드를 이용하여 구성될 수 있다.

[0032] 상기 제1 데이터 트랜잭션은 적어도 두 개의 스테이지들을 포함하고, 제2 시드 데이터를 제공하는 단계는 상기 제1 데이터 트랜잭션의 상기 제1 스테이지의 레코드와 상기 제1 영지식 증명을 결합하는 단계와, 및 상기 제1 데이터 트랜잭션의 상기 제2 스테이지의 레코드와 상기 제2 영지식 증명을 결합하는 단계를 포함할 수 있다. 제2 시드 데이터를 제공하는 단계는 상기 제1 데이터 트랜잭션의 상기 제2 스테이지의 레코드로부터 제3 영지식 증명을 구성하는 단계와, 및 상기 제1 데이터 트랜잭션의 상기 제2 스테이지의 레코드와 상기 제2 영지식 증명 및 상기 제3 영지식 증명을 결합하는 단계를 포함할 수 있다. 상기 제1 데이터 트랜잭션은 적어도 세 개의 스테이지들을 포함하고, 제2 시드 데이터를 제공하는 단계는 상기 제1 데이터 트랜잭션의 상기 제3 스테이지의 레코드와 상기 제1 영지식 증명을 결합하는 단계와, 상기 제1 데이터 트랜잭션의 상기 제3 스테이지의 레코드와 상기 제3 영지식 증명을 결합하는 단계를 더 포함할 수 있다.

[0033] 상기 제1 데이터 트랜잭션은 적어도 세 개의 스테이지들을 포함할 수 있고, 제2 시드 데이터를 제공하는 단계는 상기 제1 데이터 트랜잭션의 상기 제3 스테이지의 레코드와 상기 제1 영지식 증명을 결합하는 단계와, 랜덤 데이터와 상기 제3 영지식 증명을 결합하는 단계를 더 포함할 수 있다. 상기 제1 데이터 트랜잭션은 적어도 세 개의 스테이지들을 포함할 수 있고, 제2 시드 데이터를 제공하는 단계는 상기 제1 데이터 트랜잭션의 상기 제3 스테이지의 레코드와 상기 제1 영지식 증명을 결합하는 단계와, 및 상기 제1 데이터 트랜잭션의 제4 스테이지의 레코드와 상기 제2 영지식 증명을 결합하는 단계를 포함할 수 있고, 상기 제1 데이터 트랜잭션의 상기 제4 스테이지는 상기 제1 데이터 트랜잭션의 상기 제3 스테이지의 반복일 수 있다.

[0034] 상기 제1 데이터 트랜잭션은 적어도 세 개의 스테이지들을 포함할 수 있고 제2 시드 데이터를 제공하는 단계는 상기 제1 데이터 트랜잭션의 상기 제3 스테이지의 레코드와 제3 영지식 증명을 결합하는 단계를 더 포함할 수 있다.

[0035] 상기 제1 영지식 증명은 상기 제1 엔티티와 관련된 상기 장치에 의해 구성될 수 있고, 상기 제2 영지식 증명은 상기 제2 엔티티와 관련된 장치에 의해 구성될 수 있다.

[0036] 상기 제1 영지식 증명 및 상기 제2 영지식 증명을 구성하는 단계는 키 교환 알고리즘을 사용하는 단계를 포함할 수 있다. 상기 키 교환 알고리즘은 PAKE 알고리즘을 포함할 수 있다.

[0037] 상기 방법은 상기 제2 엔티티와 관련된 장치에 상기 제1 해시를 전송하는 단계, 상기 제2 엔티티와 관련된 장치로부터 제2 해시를 수신하는 단계- 상기 제2 해시는 상기 제2 엔티티와 관련된 이전 데이터 트랜잭션의 해시를 포함함 -와, 상기 제1 파티(first party) 및 상기 제2 파티(second party)간의 제2 데이터 트랜잭션의 레코드를 생성하는 단계와, 상기 제1 해시 및 상기 제2 해시와 상기 제2 데이터 트랜잭션의 상기 레코드를 결합하여 제3 시드 데이터를 결정하는 단계와, 상기 제3 시드 데이터를 해싱하여 제3 해시를 생성하는 단계- 상기 제3 해시는 상기 제1 엔티티와 관련된 데이터 트랜잭션들의 히스토리 및 상기 제2 엔티티와 관련된 데이터 트랜잭션들의 히스토리를 포함함 ?와, 상기 제2 데이터 트랜잭션의 상기 레코드에 대한 상기 제3 해시를 상기 메모리에 저장하는 단계를 더 포함할 수 있다.

[0038] 제3 시드 데이터를 제공하는 단계는 상기 제2 데이터 트랜잭션의 상기 레코드, 상기 제1 해시 및 상기 제2 해시와 제3 영지식 증명 및 제4 영지식 증명을 결합하는 단계를 더 포함하고, 상기 제3 영지식 증명은 상기 제1 해시가 상기 제1 데이터 트랜잭션의 트루 해시를 포함한다는 증명을 포함하고, 상기 제4 영지식 증명은 상기 제2 해시가 상기 제2 엔티티와 관련된 상기 이전 데이터 트랜잭션의 상기 트루 해시를 포함한다는 증명을 포함할 수 있다. 상기 제2 엔티티와 관련된 상기 이전 데이터 트랜잭션은 상기 제1 데이터 트랜잭션일 수 있다.

[0039] 상기 방법은 상기 제1 엔티티 및/또는 상기 제2 엔티티의 식별자와 상기 해시들 각각을 연관 시키는 단계를 더 포함할 수 있다. 상기 방법은 상기 제1 해시를 재계산하는 단계, 및 매치(match)를 결정하기 위해서 상기 생성된 제1 해시를 상기 재계산된 제2 해시와 비교하는 단계를 더 포함할 수 있다. 상기 방법은 상기 비교가 성공적이지 않는 경우, 추가 데이터 트랜잭션들을 취소하는 단계를 더 포함할 수 있다. 상기 방법은 상기 제1 데이터 트랜잭션에 대응하는 시스템 해시를 시스템 장치에 생성하는 단계를 더 포함할 수 있다.

[0040] 제2 시드 데이터를 제공하는 단계는 상기 제1 시드 데이터 및 상기 제1 데이터 트랜잭션의 상기 레코드와 상기 시스템 해시를 결합하는 단계를 더 포함할 수 있다. 상기 시스템 해시는 상기 시스템 장치상의 이전 데이터 트랜잭션의 레코드를 해싱한 결과일 수 있다.

[0041] 제2 시드 데이터를 제공하는 단계는 라이선스 장치로부터 라이선스 해시를 수신하는 단계와, 상기 제2 시드 데이터를 제공하기 위해서 상기 제1 시드 데이터 및 상기 제1 데이터 트랜잭션의 상기 레코드와 상기 라이선스 해시를 결합하는 단계를 더 포함할 수 있다.

[0042] 상기 방법은, 상기 라이선스 장치에서, 상기 제1 해시를 수신하는 단계와, 라이선스 입력을 제공하기 위해서 상기 라이선스 해시와 상기 제1 해시를 결합하는 단계와, 상기 라이선스 입력을 해싱하여 제2 라이선스 해시를 생성하는 단계를 더 포함할 수 있다.

[0043] 제2 시드 데이터를 제공하는 단계는 디렉토리 장치로부터 디렉토리 해시를 수신하는 단계와, 및 상기 제2 시드 데이터를 제공하기 위해서 상기 제1 시드 데이터 및 상기 제1 데이터 트랜잭션의 상기 레코드와 상기 디렉토리 해시를 결합하는 단계를 더 포함할 수 있다.

[0044] 상기 방법은, 상기 디렉토리 서버에서, 상기 제1 해시를 수신하는 단계와, 디렉토리 입력을 제공하기 위해서 상기 디렉토리 해시와 상기 제1 해시를 결합하는 단계와, 상기 디렉토리 입력을 해싱하여 제2 디렉토리 해시를 생성하는 단계를 더 포함할 수 있다.

[0045] 제2 시드 데이터를 제공하는 단계는 상기 제1 데이터 트랜잭션에 대한 암호화 키로부터 키 해시를 생성하는 단계와, 및 상기 제2 시드 데이터를 제공하기 위해서 상기 제1 시드 데이터 및 상기 제1 데이터 트랜잭션의 상기 레코드와 상기 키 해시를 결합하는 단계를 더 포함할 수 있다. 상기 암호화 키는 공개 키 또는 개인 키를 포함할 수 있다.

[0046] 상기 제1 시드 데이터 및 상기 제1 데이터 트랜잭션의 상기 레코드를 결합하는 단계는 상기 제1 데이터 트랜잭션이 완료되자마자 수행될 수 있다. 상기 메모리는 원격 장치에 위치할 수 있다. 상기 방법은 다른 장치들로부터 수신된 해시들에 대응하는 상기 제1 해시를 상기 원격 장치에서 비교하는 단계를 더 포함할 수 있다. 상기 방법은 상기 장치에 연결된 다른 장치들에 상기 제1 해시를 수신할 것을 예상하도록 통지하는 단계를 더 포함할 수 있다.

[0047] 상기 방법은 상기 메모리에 해시들의 체인을 저장하는 단계를 더 포함할 수 있다. 상기 방법은 전송된 상기 해시 체인들에 대한 액세스를 제한하도록 하는 장치 상에 위치한 제2 메모리에 상기 해시들의 체인을 전송하는 단계를 더 포함할 수 있다. 상기 방법은 상기 해시 체인에서 해시를 수정 또는 삭제하는 단계를 더 포함하고, 상기 해시 체인에서 해시를 수정 또는 삭제하는 단계는 상기 해시 체인에서 서브젝트 해시를 재생성하는 단계와, 상기 레코드가 수정되지 않았는지 여부를 확인하는 단계와, 상기 재생성된 해시를 레코딩하는 단계와, 상기 레코드를 수정 또는 삭제하는 단계와, 상기 서브젝트 해시의 결합 및 상기 수정/삭제된 레코드를 해싱하여 상기 레코드에 대한 새로운 해시를 생성하는 단계와, 상기 새로운 해시를 레코딩하는 단계를 포함할 수 있다. 상기 방법은 상기 새로운 해시를 이용하여 시스템 해시를 생성하는 단계를 더 포함할 수 있다.

[0048] 상기 장치는 서버를 포함할 수 있다. 상기 장치는 유저 장치를 포함할 수 있다. 상기 유저 장치는 개인용 컴퓨터, 스마트폰, 스마트 테블렛 또는 사물 인터넷(IoT) 가능 장치 중에서 적어도 하나를 포함할 수 있다. 상기 유저 장치는 상기 장치상의 메모리에서 상기 제1 해시를 저장할 수 있다. 상기 유저 장치는 해당 서버로부터 오프 라인인 경우에만, 상기 장치상의 메모리에서 상기 제1 해시를 저장할 수 있다. 상기 장치는 상기 제2 엔티티와 관련된 장치에 상기 제1 해시를 전송할 수 있다. 상기 장치는 상기 제1 데이터 트랜잭션의 상기 레코드의 서명되고, 암호화된 카피를 상기 제2 엔티티와 관련된 상기 장치로 전송할 수 있고, 상기 서명은 상기 제1 데이터 트랜잭션의 상기 레코드(that record)에 대한 목적지 서버의 표시(indication)을 포함할 수 있다. 상기 장치는 특정 오프라인 공개 키로 상기 레코드에 서명할 수 있다. 상기 장치는 상기 장치에 속하는 키로 상기 레코드에 서명할수 있다. 상기 목적지 서버만 상기 제1 데이터 트랜잭션의 상기 레코드의 상기 암호화된 카피(encrypted copy)을 해독할 수 있다. 상기 장치가 대응하는 서버와 연결을 회복할 때, 상기 장치는 상기 관련된 해시들 및 그것의 오프라인 데이터 트랜잭션들의 상기 암호화된 레코드들을 대응하는 서버로 전송할 수 있다. 상기 장치는 자신이 보유하는 다른 엔티티들을 포함하는 데이터 트랜잭션들의 레코드들의 복사본들을 상기 다른 엔티티들에 대응하는 서버들로의 전송을 위해 자신에 대응하는 서버에 전송할 수 있다. 상기 전송은 상기 레코드들이 적용되는 모든 서버들에 상기 레코드들을 수신할 것을 기대하도록 통지하는 것을 포함할 수 있다. 상기 장치는 상기 제1 데이터 트랜잭션에서 이것의 부분을 식별하기 위해 고유의 내부 트랜잭션 번호를 생성할 수 있다.

[0049] 상기 허가하는 단계는 상기 식별자에 기초하여 상기 사용자 장치가 상기 제1 서비스에 액세스하도록 허가되는지를 확인하는 단계를 포함할 수 있다. 상기 확인하는 단계는 상기 식별자에 기초하여 상기 사용자가 적어도 하나의 기준(criteria)을 만족하는지 확인하는 단계를 포함할 수 있다. 제1 기준이 상기 제1 호스트 서버 또는 상기 요청 서버에 저장되고, 제2 기준이 다른 서버에 위치할 수 있다. 상기 허가하는 단계는 상기 요청 서버 및 상기 제1 호스트 서버간의 통신에 대한 서명을 검증하는 단계를 포함할 수 있다.

[0050] 상기 허가하는 단계는 상기 요청 서버에서 수행될 수 있다. 상기 허가하는 단계는 상기 요청 서버에서 상기 장치가 상기 제1 서비스에 액세스하도록 이전에 허가되었는지를 결정하는 단계를 포함할 수 있다.

[0051] 상기 허가하는 단계는 디렉토리 서버에서 수행될 수 있다. 상기 허가하는 단계는 상기 요청 서버가 상기 디렉토리 서버로부터 상기 장치에 대한 허가를 요청하는 단계를 포함할 수 있다. 상기 액세스하게 하는 단계는 상기 디렉토리 서버가 상기 제1 호스트 서버에 대한 식별자를 상기 요청 서버에 전송하는 단계를 포함할 수 있다. 상기 식별자를 허가하는 데이터는 상기 디렉토리 서버에 저장될 수 있다.

[0052] 상기 방법은 제2 서비스에 대한 액세스를 요청하는 단계와, 상기 식별자에 기초하여 상기 장치가 상기 제2 서비스에 액세스하는 것을 허가하는 단계와, 상기 장치가 상기 요청 서버를 통해 상기 제2 서비스에 액세스하게 하는 단계를 더 포함할 수 있다. 상기 제2 서비스는 상기 제1 호스트 서버에 위치할 수 있다. 상기 제2 서비스는 제2 호스트 서버에 위치할 수 있다.

[0053] 상기 장치가 상기 제1 서비스에 액세스하는 것을 허가하는 단계는 제1 디렉토리 서버에서 수행될 수 있고, 상기 사용자 장치가 상기 제2 서비스에 액세스하는 것을 허가하는 단계는 제2 디렉토리 서버에서 수행될 수 있다.

[0054] 상기 방법은 제3 서비스에 대한 액세스를 요청하는 단계와, 상기 식별자에 기초하여 상기 장치가 상기 제3 서비스에 액세스하는 것을 허가하는 단계와, 상기 장치가 상기 제3 서비스에 액세스하게 하는 단계를 더 포함할 수 있다.

[0055] 상기 제2 서비스는 상기 제1 호스트 서버, 상기 제2 호스트 서버 또는 제3 호스트 서버에 위치할 수 있다. 상기 장치가 상기 제3 서비스에 액세스하는 것을 허가하는 단계는 제3 디렉토리 서버에서 수행될 수 있다.

[0056] 식별자를 제공하는 단계는 상기 장치가 암호화된 터널(encrypted tunnel)을 통해 상기 요청 서버와 통신하는 단계를 포함할 수 있다. 상기 방법은 각각의 개별 서버에서 수신되는 데이터를 캐싱하는 단계를 더 포함할 수 있다. 각각의 호스트 서버는 둘 이상의 서비스를 제공할 수 있다.

[0057] 상기 장치는 개인용 컴퓨터, 스마트 폰, 스마트 테블렛 또는 사물 인터넷이 가능한 장치 중에서 적어도 하나를 포함할 수 있다.

[0058] 상기 마이그레이션하는 단계는, 상기 디렉토리 서버에서, 상기 제2 데이터 저장소에서 상기 데이터에 대한 시작 타임스탬프(start timestamp)를 할당하는 단계, 및 상기 제1 데이터 저장소에서 상기 데이터에 대한 종료 타임스탬프(end timestamp)를 할당하는 단계를 포함할 수 있다.

[0059] 상기 방법은 상기 종료 타임스탬프 이후에 상기 제1 데이터 저장소를 통해 상기 데이터에 액세스하려고 시도하는 요청 서버(requesting server)에 상기 디렉토리 서버를 통해 상기 제2 데이터 저장소에서 상기 사용자를 검색하도록 지시하는 단계를 더 포함할 수 있다. 상기 제1 데이터 저장소에서의 상기 데이터는 제1 계정 제공자와의 제1 계정 등록을 포함할 수 있고, 상기 제2 데이터 저장소에서의 상기 데이터는 새로운 계정 제공자와의 제2 계정 등록을 포함할 수 있다. 상기 마이그레이션하는 단계는 상기 현재 계정 제공자로부터 상기 새로운 계정 제공자로 상기 제1 계정 등록에 관한 정보를 전송하는 단계를 포함할 수 있다. 상기 정보는 등록들(registrations), 잔액들(balances), 컨피규레이션들(configurations) 및/또는 결제 지시들(payment instructions) 중에서 적어도 하나를 포함할 수 있다. 상기 마이그레이션하는 단계는 상기 제1 등록이 상기 현재 계정 제공자로부터 상기 새로운 계정 제공자로 스위치되어야 함을 나타내는 인증 코드(authentication code)를 확인하는 단계를 포함할 수 있다. 상기 제1 계정 등록은 제1 사용자 크리덴셜을 포함할 수 있고, 상기 제2 계정 등록은 제2 사용자 크리덴셜을 포함할 수 있다. 상기 제1 사용자 크리덴셜은 제1 서버에 등록될 수 있고, 상기 제2 사용자 크리덴셜은 제2 서버에 등록될 수 있다. 상기 방법은 상기 제1 계정 제공자에 의해 상기 제1 사용자 크리덴셜을 이용하여 사용자에게 전달되는 통신을 수신하는 단계와, 상기 제2 사용자 크리덴셜을 이용하여 상기 통신을 상기 제2 계정 제공자로 라우팅하는 단계를 더 포함할 수 있다. 상기 방법은 상기 제1 크리덴셜을 사용하는 상기 제1 등록 제공자로 만들어진 데이터 트랜잭션을 상기 제2 사용자 크리덴셜을 사용하는 상기 제2 등록 제공자로 반전시키는 단계를 더 포함할 수 있다. 상기 방법은 상기 데이터 트랜잰션시 상기 사용자가 상기 제1 사용자 크리덴셜을 사용했다는 것을 결정하는 단계를 더 포함할 수 있다. 상기 통신을 전송하는 서버는 상기 제2 사용자 크리덴셜에 액세스하도록 공인되어야 한다.

[0060] 상기 장치는 개인용 컴퓨터, 스마트 폰, 스마트 테블렛 또는 사물 인터넷이 가능한 장치 중에서 적어도 하나를 포함할 수 있다.

[0061] 상기 방법은 랜덤 필드를 상기 제2 통신에 추가하는 단계를 더 포함할 수 있다. 각각의 필드는 두 개 이상의

특징들을 포함하고, 상기 방법은 적어도 하나의 필드에서 특징들의 케이스들을 믹싱하는 단계를 더 포함할 수 있다.

[0062] 상기 방법은 상기 제2 통신을 처리하기 전에 상기 제2 엔티티에 의해 상기 제2 통신에서 상기 필드들을 해독 및 순서화하는 단계를 더 포함할 수 있다. 상기 방법은 상기 제2 엔티티에 의해 처리할 수 없는 필드들을 폐기하는 단계를 더 포함할 수 있다. 상기 제1 엔티티 및 상기 제2 엔티티 중에서 적어도 하나는 서버를 포함할 수 있다. 상기 제1 엔티티 및 상기 제2 엔티티 중에서 적어도 하나는 개인용 컴퓨터, 스마트 폰, 스마트 테블렛 또는 사물 인터넷이 가능한 장치를 포함할 수 있다. 상기 장치는 개인용 컴퓨터, 스마트 폰, 스마트 테블렛 또는 사물 인터넷이 가능한 장치 중에서 적어도 하나를 포함할 수 있다.

[0063] 상기 인코딩하는 단계는 상기 사이퍼 텍스트를 7-비트 또는 8-비트 문자 스트링(7-bit or 8-bit character string)으로 인코딩하는 단계를 포함할 수 있다. 상기 방법은 상기 사이퍼 텍스트의 상기 길이가 상기 USSD 세션에서 상기 허용된 스페이스보다 긴 경우, 상기 사이퍼 텍스트를 두 개 또는 두 개 이상의 파트들로 절단하는 단계와, 상기 두 개 또는 두 개 이상의 파트들을 개별적으로 전송하는 단계를 더 포함할 수 있다. 상기 해독은 상기 제2 장치에서 상기 전체 사이퍼 텍스트로 상기 파트들을 리어셈블링하는 단계를 더 포함할 수 있다.

[0064] 상기 방법은 상기 제1 및 제2 장치들을 인증하는 단계를 더 포함할 수 있다. 상기 인증하는 단계는 두 개의 통신 컴퓨터 어플리케이션들 간의 프라이버시 및 데이터 무결성을 제공하는 알고리즘을 사용하는 단계를 포함할 수 있다. 상기 인증하는 단계는 TLS(transport layer security)를 사용하는 단계를 포함할 수 있다. TLS를 사용하는 단계는 제1 세션 키를 생성하는 단계를 포함할 수 있다.

[0065] 상기 방법은 제2 세션 키를 생성하기 위해 PAKE 프로토콜 협상(PAKE protocol negotiation)을 암호화하는 상기 제1 세션 키를 사용하는 단계와, 상기 제2 세션 키를 사용하여 상기 제1 장치와 상기 제2 장치 간의 상기 세션에서 추가 통신들을 암호화하는 단계를 더 포함할 수 있다.

[0066] 상기 방법은 상기 제1 엔티티 및 상기 제2 엔티티를 인증하는 단계를 더 포함할 수 있다. 상기 인증하는 단계는 두 개의 통신 컴퓨터 어플리케이션들 간의 프라이버시 및 데이터 무결성을 제공하는 알고리즘을 사용하는 단계를 포함할 수 있다. 상기 인증하는 단계는 TLS를 사용하는 단계를 포함할 수 있다. 상기 방법은 제4 공유 비밀(forth shared secret)을 사용하여 상기 제1 장치 및 제3 장치 간의 제2 PAKE 세션을 생성하는 단계를 더 포함할 수 있다. 상기 제4 공유 비밀은 상기 제1 장치를 위해 상기 제3 장치에 의해 생성된 인증 코드를 포함할 수 있다.

[0067] 상기 제1 공유 비밀은 상기 제1 장치를 위해 상기 제2 장치에 의해 생성된 인증 코드를 포함할 수 있다. 상기 인증 코드는 상기 제1 장치를 위해 식별자와 함께 상기 제1 장치로 전송될 수 있다. 상기 식별자는 상기 제1 장치의 전화 번호 또는 일련 번호를 포함할 수 있다. 상기 제1 공유 비밀은 상기 제1 엔티티와 연관된 은행 카드의 PAN(personal account number)을 포함할 수 있다. 상기 제1 공유 비밀은 상기 제1 엔티티와 연관된 은행 카드의 인코딩된 일련 번호를 포함할 수 있다.

[0068] 상기 장치는 개인용 컴퓨터, 스마트 폰, 스마트 테블렛 또는 사물 인터넷이 가능한 장치 중에서 적어도 하나를 포함할 수 있다.

[0069] 상기 서비스에 대한 액세스를 인증하는 단계는 상기 크리덴셜 및/또는 상기 컨텍스트에 기초하여 서비스의 일부에 대한 액세스를 인증하는 단계를 포함할 수 있다. 상기 크리덴셜은 장치 및 상기 장치의 프라이머리 사용자(primary user) 와 관련된 제1 크리덴셜을 포함할 수 있다. 상기 크리덴셜은 장치 및 상기 장치의 세컨더리 사용자(secondary user)와 관련된 제2 크리덴셜을 더 포함할 수 있다. 상기 크리덴셜에 기초하여 상기 서비스에 대한 액세스를 인증하는 단계는 상기 제1 크리덴셜 및 상기 제2 크리덴셜 각각에 기초하여 상기 프라이머리 사용자 및 상기 세컨더리 사용자에 대한 상이한 서비스들에 대한 액세스를 인증하는 단계를 포함할 수 있다. 상기 장치는 상기 프라이머리 사용자 및 상기 세컨더리 사용자에 대한 상이한 지출 한도인 상기 상이한 서비스들 및 은행 카드를 포함할 수 있다. 상기 크리덴셜은 상기 컨텍스트에 기초하여 선택될 수 있다. 상기 서비스는 상기 컨텍스트에 기초하여 선택된 복수의 서비스들을 포함할 수 있다. 관리자 또는 사용자는 상기 컨텍스트 또는 크리덴셜을 수정, 추가 또는 취소할 수 있다. 상기 크리덴셜은 패스워드, PIN, 및/또는 다른 직접 인증 크리덴셜(direct authentication credential) 중에서 적어도 하나를 포함할 수 있다. 상기 컨텍스트는 상기 크리덴셜을 제공하는 장치, 상기 장치상의 어플리케이션, 상기 장치가 연결된 네트워크, 상기 장치의 지리적 위치, 및/또는 액세스되는 상기 서비스 중에서 적어도 하나를 포함할 수 있다.

[0070] 상기 장치는 개인용 컴퓨터, 스마트 폰, 스마트 테블렛 또는 사물 인터넷이 가능한 장치 중에서 적어도 하나를

포함할 수 있다.

[0071] 상기 방법은 복수의 요청들을 상기 제1 모듈의 버퍼 메모리에서 배치된 메시지(batched message)로 배칭하는(batching) 단계와, 상기 제2 모듈로 전송되는 상기 배치된 메시지를 큐잉하는 단계와, 시스템 기능을 허가하는 적어도 하나의 시스템 플래그를 셋팅하는 단계와, 상기 제2 모듈에서 상기 적어도 하나의 시스템 플래그를 체크하는 단계와, 상기 제2 모듈에서 상기 배치된 메시지를 처리하는 단계를 더 포함할 수 있다.

[0072] 상기 방법은 상기 제1 모듈 및 상기 제2 모듈 간의 적어도 하나의 공유 메모리 채널을 설정하는 단계를 더 포함할 수 있다. 상기 방법은 상기 적어도 하나의 공유 메모리 채널을 통해 상기 제1 모듈에 응답하는 상기 제2 모듈을 포함할 수 있다. 상기 적어도 하나의 공유 메모리 채널은 상기 배치된 메시지를 수신 및 어셈블링하고, 상기 제2 모듈로 상기 메모리의 소유권을 넘겨줄 수 있다. 상기 적어도 하나의 공유 메모리 채널은 상기 컴퓨터 시스템의 네트워크 스택(network stack)을 통해 배치된 메시지를 수신할 수 있다. 상기 적어도 하나의 공유 메모리 채널은 HTTP 게이트웨이를 포함할 수 있다. 상기 HTTP 게이트웨이는 웹 서비스로써 사용될 수 있다.

[0073] 통신은 패스워드 인증된 키 교환 프로토콜(password authenticated key exchange protocol)을 사용할 수 있다. 상기 방법은 상기 컴퓨터 시스템의 네트워크 스택에서 제로-카피 네트워킹(zero-copy networking)을 사용하는 단계를 더 포함할 수 있다. 상기 방법은 상기 컴퓨터 시스템의 네트워크 스택에서 사용자-모드 네트워킹(user-mode networking)을 사용하는 단계를 더 포함할 수 있다.

[0074] 상기 방법은 상기 제1 모듈로부터 상기 데이터 전송(data transmission)의 상기 컴포넌트들이 단일 데이터 스트림(single data stream)으로 결합되고, 상기 제1 모듈에서 상기 컴포넌트들로 분리되도록 데이터를 직렬화하는 단계를 더 포함할 수 있다. 상기 직렬화는 각 모듈의 에지에서 추상화될 수 있다.

[0075] 각 모듈의 버퍼 메모리는 구성 가능한 버퍼링 임계값(a configurable threshold of buffering)을 갖을 수 있다. 상기 제1 모듈 및 상기 제2 모듈은 동일한 컴퓨팅 장치 상에 위치할 수 있다. 상기 제1 모듈 및 상기 제2 모듈은 상이한 컴퓨팅 장치들 상에 위치할 수 있다.

[0076] 상기 제1 모듈로부터 상기 제2 모듈로 전송된 데이터는 버전 ID(version ID)를 운반할 수 있다. 상기 방법은 상기 버전 ID가 상기 제1 모듈로부터 상기 제2 모듈로 전송된 상기 데이터에 대해 최신인지를 검증하는 단계를 더 포함할 수 있다. 상기 방법은 상기 데이터 중에서 임의의 데이터가 업데이트되는 경우 , 상기 버전 ID를 현재 버전으로 재검증하는 단계를 더 포함할 수 있다. 상기 버전 ID가 검증되지 않는 경우, 상기 데이터 전송은 실패할 수 있다.

[0077] 상기 제1 모듈 및 상기 제2 모듈 중에서 적어도 하나는 적어도 하나의 데이터 서비스 모듈을 포함할 수 있고, 상기 컴퓨터 시스템 내의 각각의 데이터 활동은 상기 적어도 하나의 데이터 서비스 모듈을 통해 실행될 수 있다. 상기 적어도 하나의 데이터 서비스 모듈은 코어 데이터베이스 저장소(core database store)에 의해 구현되는 데이터 저장소와 통신할 수 있다. 상기 적어도 하나의 데이터 서비스 모듈은 상기 데이터 저장소에 직접 액세스하는 상기 컴퓨터 시스템의 컴포넌트일 수 있다. 상기 코어 데이터베이스 저장소는 적어도 하나의 분산 데이터베이스(distributed database)를 포함할 수 있다. 상기 적어도 하나의 분산 데이터베이스는 별도의 판독(read) 및 기록(write) 액세스 채널들을 가질 수 있다. 상기 데이터 저장소는 적어도 하나의 이종 데이터베이스(heterogeneous database)에 인터페이스를 제공할 수 있다. 상기 데이터 저장소는 복수의 인터페이스 타입들을 제공할 수 있다. 상기 복수의 인터페이스 타입들은 적어도 하나의 SQL(Structured Query Language) 인터페이스, 셀 및 칼럼 인터페이스(cell and column interface), 문서 인터페이스(document interface), 및 상기 코어 데이터베이스 저장소 위에 있는 그래픽 인터페이스(graph interface) 중에서 적어도 하나를 포함할 수 있다. 상기 데이터 저장소 레이어에 대한 모든 기록들은 하나 또는 하나 이상의 데이터 트랜잭션들의 전부 또는 일부를 제어하는 단일 공유 모듈에 의해 관리될 수 있다.

[0078] 상기 방법은 적어도 하나의 상기 공유 모듈의 리던던트 백업(redundant backup)을 작동시키는 단계를 더 포함할 수 있다. 모든 데이터 변경은 시리얼 빠른 시퀀스(serial rapid sequence) 로 상기 단일 공유 모듈을 통해 진행될 수 있다. 상기 단일 공유 모듈은 그 자체를 데이터 트랜잭터 클러스터(data transactor cluster)로 나타내는 핫 백업 리던던시 모델(hot backup redundancy model)을 사용할 수 있고, 상기 데이터 트랜잭터 클러스터는 하이어라키(hierarchy)에서 모듈들의 세트이고, 각 모듈은 마스터 모듈(master module)이 실패하는 경우, 데이터 트랜잭션들을 제어할 수 있다. 상기 방법은 도메인에 의해 구성되는 규칙들에 기초하여 모듈들 또는 데이터 저장소들에 걸쳐 데이터를 분할하는 단계를 더 포함할 수 있다. 상기 방법은 데이터 트랜잭션의 레코드 또는 부모 데이터 트랜잭션(parent data transaction)의 레코드의 타겟 데이터를 해싱하는 단계를 더 포함할 수

있다. 상기 해싱하는 단계는 데이터 파티션들의 수와 동일한 카디널리티(cardinality)를 가질 수 있다. 상기 방법은 열거된 지리적 영역(enumerated geographical area), 성(last name) 및/또는 통화(currency) 중에서 적어도 하나에 의해 타겟 데이터를 해싱하는 단계를 더 포함할 수 있다.

[0079] 상기 방법은 다중 데이터 파티션들에 걸쳐 상기 적어도 하나의 데이터 서비스 모듈을 통해 적어도 하나의 데이터 전송을 수행하는 단계를 더 포함할 수 있다. 상기 방법은 다중 모듈들에 의해 상기 적어도 하나의 데이터 서비스 모듈을 통해 적어도 하나의 데이터 전송을 완료하는 단계를 더 포함할 수 있다. 상기 방법은 상기 적어도 하나의 데이터 서비스 모듈 상의 적어도 하나의 데이터 전송을 상기 데이터 저장소에서 다중 데이터 스토리지 노드들 상에 유지하는 단계를 더 포함할 수 있다.

[0080] 상기 컴퓨터 시스템은 복수의 데이터 서비스 모듈들을 포함할 수 있고, 각각의 데이터 서비스 모듈은 해당 인스턴스에 대한 모든 상기 핫 데이터의 캐시된 표현을 포함하여 인-메모리(in-memory)/인-프로세스(in-process) 데이터베이스 엔진을 호스트할 수 있다. 상기 컴퓨터 시스템은 복수의 데이터 서비스 모듈들을 포함할 수 있고, 각각의 데이터 서비스 모듈은 복수의 이종 또는 동종 데이터베이스 엔진들을 포함할 수 있다.

[0081] 상기 방법은 정확하게 모든 데이터 판독들이 일관되고, 대응하는 데이터 기록들을 반영하도록, 상기 데이터 저장소에 대한 액세스의 동시성을 관리하는 MVCC(Multiversion Concurrency Control) 버전 시스템을 사용하는 단계를 더 포함할 수 있다. 상기 방법은 데이터 레코드가 상기 데이터 저장소에 기록되어야 하고, 임의의 후속 데이터 트랜잭션이 상기 데이터 레코드에 액세스하기 전에 기록된 것으로 확인되어야 하도록, 상기 데이터 저장소에 대한 액세스의 동시성을 관리하는 비관적 일관성(pessimistic consistency)을 사용하는 단계를 더 포함할 수 있다.

[0082] 상기 컴퓨터 시스템은 어플리케이션 레이어를 더 포함할 수 있고, 상기 적어도 하나의 데이터 서비스 모듈이 상기 레코드를 기록하고, 상기 데이터 전송을 완료함을 확인할 때까지, 상기 어플리케이션 레이어는 데이터 트랜잭션을 진행할 수 없다.

[0083] 제1 실시예 내지 제26 실시예의 모든 선택적인 특징들은 필요한 부분만 약간 수정하여 모든 다른 실시예들과 관련이 있다. 설명된 실시예들의 변형은 고려될 수 있으며, 예를 들어, 모든 개시된 실시예들의 특징들은 임의의 방식으로 결합될 수 있다.

도면의 간단한 설명

[0084] 본 발명의 실시예들은 이제 동일한 부분들을 도시하기 위해 동일한 참조 번호들이 사용된 첨부 도면들을 참조하여 예로서 설명될 것이다.

도면들에서:

도 1은 Tereon의 모듈러 개념을 도시한다;

도 2는 Tereon 시스템 아키텍처의 예를 도시한다;

도 2a는 Tereon 이 서비스들 및 장치들을 기능 영역들 및 컨텍스트들, 장치들, 컴포넌트들 및 프로토콜들로 추상화하는 방법을 도시한다;

도 3은 중개자 프록시를 통한 TLS 연결들을 통해 시작된 통신들을 나타낸다;

도 4는 프록시 메모리로 공유 메모리 및 메시지 전달의 사용을 도시한다;

도 4a는 공유 메모리 및 세마포르 핸드-오버 모듈을 도시한다;

도 5는 네 개의 계정들을 포함하는 해시 체인(hash chain)을 도시한다;

도 6은 동일한 시스템상의 두 개의 계정들을 포함하는 해시 체인을 도시한다;

도 6a는 트랜잭션 스테이지들이 인터리빙하는 동일한 시스템 상의 세 개의 계정들을 포함하는 해시 체인을 도시한다;

도 7은 라이선스 해시들(licence hashes)의 덴드리틱 특성(dendritic nature)을 도시한다;

도 8은 잠시 오프라인 상태가 되는 네 개의 장치들을 포함하는 해시 체인을 도시한다;

도 9는 두 개의 서버들에 구현된 역 룩-업 기능을 도시한다;

도 10은 Tereon 서버들 간의 통신 설정을 도시한다;

도 11은 사용자가 다른 서버로 이동하는 통신들을 도시한다;

도 12는 디렉토리 서비스가 요청 서버를 두 개의 다른 서버들에 연결할 수 있는 방법을 도시한다;

도 13은 다각적인 크리덴셜(multifaceted credential)을 구성하기 위해서 서버가 세 개의 서버들로부터 크리덴셜들을 획득해야 하는 케이스를 도시한다;

도 14는 은행과 사용자의 관계를 도시한다;

도 15는 계정이 이체되는 프로세스를 도시한다;

도 16은 등록된 모바일 번호가 변경되는 프로세스를 도시한다;

도 17은 두 개의 화폐들에 액세스하기 위해서 기 등록된 모바일 번호의 유지를 도시한다;

도 17a는 각각의 통화가 별도의 서버상에 있는 두개의 통화들에 액세스하기 위해서 기 등록된 모바일 번호의 유지를 도시한다;

도 18은 워크 플로우를 도시한다;

도 19는 대안적인 워크 플로우를 도시한다;

도 20은 대안적인 워크 플로우를 도시한다; 그리고,

도 21은 예시적인 컴퓨팅 시스템을 도시한다.

발명을 실시하기 위한 구체적인 내용

[0085] Tereon은 전자 트랜잭션 처리 및 인증 엔징이다. 이것은 모바일 및 전자 결제 처리 시스템으로 구현될 수 있다. 또한, 이것은 IoT 통신 시스템의 일부로 다른 구현들에서 사용될 수 있다.

[0086] Tereon은 임의의 IP(internet protocol) 지원 장치 및 이러한 IP 지원 장치와 상호 작용할 수 있는 임의의 장치들에 대한 트랜잭션 기능을 제공한다. 각 장치는 고유한 ID가 있어야 한다. Tereon의 사용 사례들은 IoT 장치들로부터 의료 기록 액세스 및 관리, 모바일 폰, 결계 단말기 또는 ATM(Automated Teller Machin)과 같은 아주 흔한 결제까지 다양하다. 초기 구현 예에서, Tereon은 모바일 폰들, 카드들, POS(poing-of-sale) 단말기들 및 임의의 고유 참조 ID를 지원한다. Tereon은 소비자들 및 머천트들이 결제하고, 결제받고, 자금을 이체하고, 자금을 받고, 환불하고, 환불받고, 자금을 예치하고, 자금을 인출하고, 계정 데이터(또는 계좌 데이터)를 보고, 과거 트랜잭션들의 미니 명세서들을 볼 수 있게 하는데 필요한 기능을 제공한다. Tereon은 통화간 및 국경간 트랜잭션들을 지원한다. 따라서, 소비자는 하나의 통화로 계정(또는 계좌)를 보유할 수 있지만, 예를 들어, 다른 통화로 이체할 수 있다.

[0087] Tereon의 초기 구현에서, 최종 유저가 특정 트랜잭션을 수행할 수 있는지 여부는 그가 해당 시점에서 사용중인 어플리케이션에 따라 다르다. 머천트들 또는 머천트의 단말기들은 일부 트랜잭션들을 시작할 수 있는 반면, 소비자 장치는 다른 것들을 시작할 수 있다.

[0088] Tereon이 결제를 처리하는데 사용되는 경우, 트랜잭션들은 다음과 같은 모드들로 세분화(또는 분할)될 수 있다: 결제 하기 및 결제 받기, 모바일 소비자 대 모바일 머천트, 모바일 소비자 대 온라인 머천트 포털, 고객이 없는 모바일 소비자 대 모바일 머천트, 계정 포털 내에서 소비자 계정 대 머천트 계정, NFC-Tereon 카드 소비자 대 카드 머천트, NFC 또는 다른 카드 소비자 대 카드 머천트, 자금 이체 및 수령, 계정 포털 내에서 소비자 계정 대 소비자 계정, 모바일 소비자 대 피어-투-피어 모바일 소비자, 모바일 소비자 대 피어-투-피어 카드 소비자, 카드 소비자 대 피어-투-피어 모바일 소비자, 카드 소비자 대 피어-투-피어 카드 소비자, 모바일 소비자 대 피어-투-피어 비 사용자, 카드 소비자 대 피어-투-피어 비 사용자, 비 사용자 대 피어-투-피어 비 사용자, 비 사용자 대 피어-투-피어 모바일 소비자 및 비 사용자 대 피어-투-피어 카드 소비자. 비 사용자는 송금을 받지 않은 수령자와 같이 결제 서비스에 이전에 등록되지 않은 어떤 사람을 나타낼 수 있다.

[0089] 시스템 아키텍처(System Architecture)

[0090] 내부적으로 Tereon 서버는 두 개의 메인 컴포넌트들인 TRE(Tereon Rules Engine) 및 SDASF(Smart Device Application Services Framework)를 포함한다.

[0091] SDASF는 Tereon이 여러 가지 다른 장치들 및 인터페이스들을 관리하게 할 수 있다. 이것은 Tereon이 해당 장치들 및 인터페이스들이 작동하고, Tereon에 연결되는 방식을 정의하기 위해서 일련의 추상화된 레이어들을 사용 및 연결하게 함으로써 그렇게 한다.

[0092] 예를 들어, 모든 은행 카드들은 기본 카드 추상화 레이어(basic card abstraction layer)를 사용할 것이다. 마그네틱 스트라이프 추상화 레이어(magnetic stripe abstraction layer)는 마그네틱 스트라이프가 있는 카드들, NFC 칩이 있는 카드들에 대한 NFC 레이어, 및 칩 컨택트(chip contact)가 있는 카드들에 대한 마이크로프로세서 레이어에 적용될 것이다. 카드가 세 개를 모두 사용하는 경우, Tereon은 메인 카드 추상화 레이어 및 세 개의 인터페이스 레이어들로 그 카드를 정의할 것이다. NFC 레이어 그 자체가 카드들에만 적용되는 것은 아닐 것이다. 이것은 모바일 폰들을 포함하여 NFC를 지원할 수 있는 임의의 장치들에도 적용될 것이다. SDASF는 장치들 또는 인터페이스들 각각에 대한 모듈을 생성하기 위해서 이러한 추상화 레이어들을 사용한다.

[0093] 외부적으로 장치 또는 네트워크에 대한 각 연결(또는 각 접속) 및 각 서비스는 모듈이다. 따라서, 피어-투-피어 결제 서비스, 예금 서비스, 및 미니 명세서들과 같은 서비스들은 모두 모듈들이다. 카드 제조업체들, 은행들, 서비스 제공자들, 단말기들, ATM들 등에 대한 인터페이스들도 마찬가지이다. Tereon의 아키텍처는 여러 가지 모듈들을 지원할 수 있다.

[0094] *모듈러 관점(Modular view).*

[0095] 도 1은 Tereon의 모듈러 개념을 도시한다. 본질적으로, Tereon은 모듈들의 모음(collection)이고, 그것들 자신 대부분은 모듈들을 포함한다. 모듈들은 해당 모듈이 동작하는 컨텍스트들 및 기능 도메인들과 그것들이 수행하는 데 필요한 기능들을 결정하는 비즈니스 로직에 의해 정의된다. 이러한 기능들은 예를 들어, IoT 장치들 간의 동작 및 통신을 관리하고, 전자 또는 디지털 결제를 관리 및 거래하고, 식별 또는 요구에 따른 허가 크리덴셜들을 관리 및 구성하거나, 임의의 다른 형식의 전자 트랜잭션(electronic transaction) 또는 장치를 관리 및 운영하는 것과 같은 임의의 타입의 전자 트랜잭션일 수 있다.

[0096] *Tereon 서버*

[0097] 도 1에 도시된 바와 같이 Tereon 서버(102)를 구성하는 모듈들은 두 개의 레벨들로 볼 수 있다: SDASF(104) 및 규칙 엔진(rules engine; 106). 규칙 엔진(106) 자체는 모듈들(108, 이들 중 일부는 도 1에 도시되어 있음; 이들은 서비스를 정의하는 모듈들, 프로토콜들(미도시), 스마트 장치, 단말들 등을 포함) 각각의 기능 도메인들 및 컨텍스트들을 정의하고, 다음에 이러한 모듈들(108)은 SDASF(104)의 구조를 정의한다. 그런 다음, SDASF(104) 및 이것이 지원하는 결과 서비스들 및 인터페이스들은 Tereon이 이용할 수 있는 시스템 프로토콜들을 정의한다. 그런 다음, 이러한 프로토콜들은 Tereon이 지원할 수 있는 규칙들 및 서비스들(예를 들어, 스마트 장치들, 단말기들 등) - 그것들 자신들은 자체 Tereon이 제공하는 기능 도메인들 및 컨텍스트들을 정의함 - 을 정의한다. 이 순환 또는 반복 접근법(circular or iterative approach)은 모듈들의 정의 및 그것들이 지원하는 기능들 또는 요구 사항들이 서로 일치하는 지를 확인하는 데 사용된다. 이것은 시스템의 동작을 제한하지 않고 원 위치에서 모듈들이 업데이트되고, 업그레이드되고, 교체될 수 있게 한다.

[0098] 블록들 및 모듈들은 추상화된 APIs((application programming interfaces)- 그것들 자신들은 자체 Tereon이 제공하는 기능 도메인들 및 컨텍스트들을 정의함 -를 사용하여 서로 인터페이스한다. 가능하다면, 이들은 공유 메모리를 사용할 수 있는 맞춤형 세마포르 핸드-오프 모듈들(bespoke semaphore hand-off modules)을 사용하여 서로 통신하며, 그 예는 도 4a에 도시되어 있고, 이후에 설명될 것이다. 이러한 방식에서, 블록들 및 모듈들의 내부 동작 및 기능은 전체 시스템의 동작을 손상시키지 않고, 업데이트되거나 교체될 수 있다.

[0099] 프레임 워크 인프라스트럭처 컴포넌트들(Framework infrastructure components)

[0100] 인프라스트럭처 컴포넌트들도 모듈러이다. SDASF의 경우, 이 컴포넌트는 그 자체가 모듈들을 포함한다.

[0101] *다중 인터페이스들(Multiple interfaces)*

[0102] 각 인터페이스는 코어 서버에 연결되는(또는 접속) 별도의 모듈로 구성된다. 따라서, Tereon의 모듈러 구조는 백 오피스들(back offices) 및 코어 시스템들 포함하는 다중 인터페이스들, 카드들, 클리어링 하우스들, 머천트들, 모바일 전화기들, 서비스들, 서비스 제공자들, 스토리지, 단말기들, SMS(short messaga service) 게이트웨이들, HLR(home location register) 게이트웨이들 등을 지원할 수 있다.

[0103] 데이터베이스 인터페이스들은 SQL(structured query language) 엔트리 및 저장된 데이터의 그래프 분석 모두를 지원한다. 또한, 인터페이스들은 데이터베이스들 내에 필드들을 구분하기 위해서 액세스 제어를 지원한다. 다

른 사용자 규칙들 및 허가 레벨들은 정의된 데이터 세트들 및 필드들을 액세스할 수 있다. 액세스는 다양한 보안 수단들에 의해 제어된다. 액세스, 인증, 및 허가는 ACLs(access control lists), LDAP(lightweight directory access protocol), 셀 및 로우 보안(cell and row security)과 같은 커스텀 역할 기반 액세스(custom role-based access), 및 개별 역할들로 제한되는 액세스 인터페이스들을 포함하는 다양한 산업 표준 접근 방법들(industry standard approaches)을 통해 제공될 수 있다.

[0104] *전자 상거래 포털들(E-commerce portals)*

[0105] Tereon은 포털의 운영자가 해당 포털에 대한 플러그 인을 생성할 수 있도록 API를 통해 전자 상거래 포털들을 지원할 수 있다.

[0106] *규칙 엔진(Rules engine)*

[0107] 규칙 엔진(106)은 새로운 서비스들이 트랜잭션을 위해 추상화된 다양한 컴포넌트들을 함께 편성하여 구축되도록 하거나 새로운 장치를 지원하게 할 수 있다. 규칙들은 배포된 서비스들에 대한 비즈니스 논리를 정의하고, 서비스 제공자들은 이러한 서비스들을 개별 사용자들에게 맞출 수 있다.

[0108] 규칙들은 UML(unofied modelling language) 또는 일반 영어(plain english)와 유사한 코드로 정의될 수 있다. 엔진은 규칙들을 구문 분석하고, 추상화된 컴포넌트들로부터 서비스들을 생성할 수 있다.

[0109] 컴포넌트들의 추상화된 특성은 새로운 서비스 또는 장치 모듈들이 신속하게 생성되게 할 수 있다. 이것은 Tereon이 새로운 서비스들 또는 장치들을 필요에 따라 지원하게 할 수 있다.

[0110] Tereon의 내부 인터페이스들은 프로토콜에 영향을 받지 않으므로, 외부 프로토콜 모듈들이 기능에 영향을 미치지 않고 교환될 수 있다. 예를 들어, 뱅킹 코어 시스템에 인터페이스하기 위해서 커스텀 데이터 교환 프로토콜(custom data interchange protocol)은 조직의 한 부분과 ISO 20022 프로토콜 모듈을 다른 부분과 함께 사용될 수 있다.

[0111] SDASF(104)는 Tereon이 다중 스마트 장치들 및 프로토콜들을 지원하게 할 수 있다. SDASF(104)의 아이디어는 엔티티들을 장치 타입들 및 프로토콜들로 추상화하는 것이다. SDASF(104)는 각 장치가 특정 서비스 또는 기능을 위해 요구되는 어느 프로토콜이든 호출한 채로 다중 프로토콜을 정의한다.

[0112] SDASF(104)는 설치의 운영에 영향을 주지 않고, 기존 설치에 새로운 모듈들을 추가하여 확장될 수 있다. 이것은 선호되는 어느 방법이든지 이용하여 모든 서비스들이 백 오피스 서버에 정의되게 할 수 있다. 머천트 단말기들(merchant terminals, 또는 가맹점 단말기들)에 설치되면, Tereon 단말기 어플리케이션들은 소비자에게 서비스들을 제공하기 위하여 SDASF와 통신한다.

[0113] 도 2는 Tereon 시스템 아키텍처(200)를 나타낸다. 여기서 다이어그램 및 묘사(narrative)가 특정 솔루션을 통해 특정 컴포넌트를 가리키는 경우, 이것은 단순하게 이들이 실시예에서 선택된 컴포넌트들 또는 언어들이기 때문이다. 맞춤형(bespoke) 시스템들은 이러한 컴포넌트들을 대체하거나 더 효율적인 것으로 입증될 수 있는 다른 언어들 및 시스템들을 사용하기 위해 구축될 수 있다.

[0114] *Tereon 서버(The Tereon server)*

[0115] Tereon 서비스(202)는 모놀리식 아티팩트(monolithic artefact)로 식별되는 논리적 구조이다. 실제로, 이것은 각각 기능 및 범위에 따라 다를 수 있는 격리된 마이크로서비스들(isolated microservices)의 세트로 존재할 수 있다.

[0116] *통신 레이어(The communications layer)*

[0117] 통신 레이어(204)는 중개자 프록시(intermediary proxy)를 거쳐 TLS(transport layer security) 연결을 통해 개시된다. 또한, 이것은 도 3에 도시된다. TLS는 컴퓨터 네트워크, 일반적으로 TCP/IP(transmission control protocol/internet protocol) 네트워크를 통해 통신 보안을 제공하는 암호화 프로토콜(cryptographic protocol)이다. 각 컴포넌트는 시스템, 오브젝트 또는 서비스에 연결하거나 액세스할 수 있는 사용자들 또는 시스템 프로세스들을 명시하는 ACL(access control list)이 있다. 이것은 중개자가 들어오는 원본 연결(incoming, original connection)을 설정하고, 본질적인 보안을 강화하며, 위험 프로파일을 줄일 수 있게 할 수 있다. 이 예에서, 프록시는 전문화된 Tereon 맞춤화(specialized Tereon customizations)와 함께 종래 기술에서 공지된 HTTP 게이트 웨이 플랫폼을 사용한다.

[0118] *개인 DNS 네트워크(Private DNS network)*

[0119] DNS(206)는 디렉토리 서비스(216)의 기반으로 사용된다. 디렉토리 서비스(216)는 고도로 중복되고, 지리적 위치들에 걸쳐 복제된다. 그러나, 이 구조와 능력은 하기에 설명된 대로 기존 DNS 서비스들이 제공할 수 있는 것보다 훨씬 크다.

[0120] *추상화(Abstractions)*

[0121] 도 2a는 Tereon이 이 서비스들 및 장치들을 소비자 또는 소비자 활동들 및 규칙들, 머천트 활동들 및 규칙들, 은행 활동들 및 규칙들, 이체 활동들 및 규칙들, 장치 기능 및 규칙들 등과 같은 기능 도메인들 및 컨텍스트들로 추상화하는 방법을 도시한다. 도 1은 컴포넌트들 및 시스템의 서비스들을 기능 블록들 또는 모듈들로 추상화하여 Tereon이 이러한 추상화에 어떻게 영상을 미치는지 도시한다.

[0122] Tereon 모듈들은 이러한 추상화로부터 구성된다. 각 장치, 각 인터페이스 및 각 트랜잭션 타입은 이 도메인들 및 컨텍스트들로 추상화된다. 이러한 추상화는 재사용이 가능하고, 의미가 있거나 허용되는 경우, 다른 것들에 인터페이스할 수 있다. 예를 들어, 청구 카드(charge card, 또는 고객 카드), 신용 카드(credit card), 직불 카드(debit card) 및 료열티 카드 모듈들(loyalty card modules)은 많은 공통 추상화(common abstractions, 또는 기본 추상화)를 각각 사용할 것이다. 결제 및 자금 이체 모듈들도 마찬가지일 것이다.

[0123] *프로토콜들(Protocols)*

[0124] Tereon이 지원하는 프로토콜들(204 및 212) 각각은 그 자체가 모듈로 구현된다. Tereon은 이러한 모듈들을 필요로하는 서비스들 또는 컴포넌트들에 이러한 모듈들을 이용할 수 있게 한다.

[0125] 레거시 시스템들은 그것들이 하드웨어를 추가해야 하기 전에 100s 또는 1000s에 동시에 일어나는 트랜잭션들을 처리하는데 어려움을 격는다. 그것들의 시스템들을 업데이트하는 대신, 은행들은 조정 계정들(reconciliation accounts, 또는 조정 계좌들) 및 결제 포인트까지 신용을 커버하기 위한 고 비용들이 요구되는 주기적인 결제 시스템들(periodic settlement systems)에 의존해 왔다. Tereon은 신용 노출(credit exposure) 및 이러한 계정들에 대한 필요성과 떨어져 있다. 이것은 초당 100,000건의 트랜잭션들을 처리하도록 요청받는 매우 저렴한 시스템들을 제공한다. Tereon은 탄력성이 구축되고, 서버당 초당 1,000,000건의 트랜잭션들을 지원하고, 값 비싼 하드웨어에 의존하기 보다 하이엔드 상품 하드웨어에서 작동하도록 설계되었다. 또한, Tereon은 ACID 보증들 및 이것의 실시간 성능을 손상시키지 않으면서 거의 선형 방식(near-linear fashion)으로 수평 및 수직 스케일링을 지원한다.

[0126] *라이센싱 서브시스템(The licensing subsystem)*

[0127] Tereon 라이센싱 서버(210)는 시스템의 컴포넌트들이 단일 배포된 인스턴스(single deployed instance) - 단일 인스턴스의 마이크로서비스들은 기계가 예를 들어, 물리적 기계(physical machine), 논리적 기계(logical machine), 가상 기계(virtual machine), 컨테이너이거나 실행 가능한 코드를 포함하기 위한 기타 일반적으로 사용되는 매커니즘, 및 모든 임의의 번호(across any number) 또는 기계들의 타입인지 여부에 관계없이 단일 기계상의 프로세스 간 통신(inter-process communications)에 결합됨- 내에서나 배포 인스턴스들 전체(across deployment instances, 예를 들어 서로 통신하는 개별 소비자 플랫폼들) 내에서 합법적이고, 인증되고, 인가된 피어 시스템들(peer systems)과 통신하고 있음을 보장한다. 라이센싱 플랫폼(licensing platform)은 당 업계에 공지된 인증 기관 구조(certificate authority structure)를 통해 구현된다.

[0128] 컴포넌트들이 시스템에 설치될 때, 그것들은 규정되고 구성 가능한 간격들로, 안전하고 인증된 연결을 통해 라이선스 서버(licence server)에 인증서 서명 요청(certificate signing request)과 함께 그것들의 설치 세부사항(조직(organization), 컴포넌트 타입 및 세부사항, 라이선스 키 등)을 전달한다.

[0129] 인증서 서버(certificate server)는 그것들의 세부사항(또는 세부정보)들을 허가된 컴포넌트 디렉토리(authorised component directory)와 비교하고, 일치하는 경우, 설치 요청을 개시하는 장치에 내부 인증 기관 하이어라키(internal certificate authority hierarchy)에서 격리된 보안 서명 키(isolated, secured signing key, 일반적으로 하드웨어 보안 모듈을 통해)로 서명되고, 규정된 기간(예를 들어, 1 개월) 동안 사용가능한 새로운 인증서를 승인한다. 연결된 시스템들에서 모든 클럭들(clocks)은 동기화된다.

[0130] 그런 다음, 호출자(caller, 또는 발신자)는 다른 모듈들과 통신을 개시할 때 클라이언트 인증서로써 인증서를 사용하고, 연결의 수신자로 역할을 할 때 서버 인증서로써 인증서를 사용할 수 있다. 개인 키를 수신한 적 없는 라이선스 서버는 손상되는 경우에도 임의의 다른 당사자(any other party)가 이 인증서를 가장할 수 있도록

하는 세부사항(또는 세부정보)들을 가지고 있지 않는다. 선호하는 경우, 호출자는 두 개의 인증서들-클라이언트 인증서 및 서버 인증서-를 라이선스 서버로부터 요청할 수 있다.

[0131] 각 컴포넌트는 서버 및 클라이언트 인증서들이 신뢰되고 인증된 인증 기관의 에이전트에 의해 서명되었는지 검사하고, 그것들이 중간자(man-in-the-middle) 공격 또는 감시에 대한 대상이 아니며, 상대방(counter-party)이 말하는 누구라는 상당한 확신을 가지고 통신할 수 있다. 각 인증서는 각 모듈 자체-예를 들어, 특정 조직에 대한 룩업 서버(lookup server)로-를 표시할 수 있는 방법을 제한하는 사용 코드 메타데이터(usage code metadata)로 승인된다. 조직은 모든 당사자가 인가되고(또는 라이선스가 있고), 합법적으로 유효한 인스턴스들을 운영함을 확신한다.

[0132] 대부분의 인증서는 간단하게 만료되고 고정된 기간(fixed term) 동안 갱신되지 않고, 승인되지 않는다. 그러나, 드물게 인증서가 손상되거나 라이선스가 종료 또는 일시 중단되는 경우, 해지 목록(revocation list)은 사용되고 필요에 따라 프록시 서비스들(proxy services)에 비동기적으로 분배된다. 액티브 인증서 디렉토리(active certificate directory)는 항상 유지되고, 주기적인 감사(periodic auditing)에 사용할 수 있다.

[0133] 양방향 유효성 검사의 이점들(two-way validation benefits) 외에도(클라이언트는 자신이 말하는 사람이고, 각 연결에서 서버는 보고하는 자임), 이 구현은 컴포넌트들이 원격 라이선스 서버들과의 통신이 필요한 각 연결 구축없이 안전하게 서로 통신하게 하여 플랫폼의 전반적인 신뢰성(reliability)을 잠재적으로 감소시키지 않으면서 안전하게 통신하게 한다.

[0134] *사이트 간 통신(Site to site communications)*

[0135] 사이트 간 통신은 식별되고 노출된 HTTP 게이트웨이 인스턴스(HTTP gateway instance; 212)를 통해 용이하게 되고, 커스텀 제로-카피(custom zero-copy) 및 선택적인 사용자-모드 기능(optional user-mode functionality)을 실행한다. 이것은 모바일 장치들, 단말기들, 및 다른 외부 당사자들이 사이트 간 연결은 물론 인스턴스들과 통신하는데 사용되는 플랫폼이다. 이것은 산업 표준 침입 검침(industry standard intrusion detection), 속도 제한(rate limiting) 및 DDOS(distributed denial-of-service) 공격 보호, 하드웨어 암호화 오프로딩(hardware encryption offloading) 등을 수용한다. 이것은 기능적으로 논리적 인스턴스 프록시 메커니즘(logical instance proxy mechanism)이 크고, 모든 동일한 기능-클라이언트/서버 인증서들 및 유효성 검사(validation, 또는 확인)을 포함함-을 지원하는 동시에 외부에서 인정된 인증 기관을 외부 당사자에 사용한다.

[0136] *Tereon 데이터 서비스(The Tereon data service)*

[0137] Tereon 시스템의 핵심 특징들 중에서 하나는 이전 시스템들 보다 상당히 많은 트랜잭션들을 (처리량 면에서) 처리할 수 있다는 것이다. 이것은 데이터 및 트랜잭션들을 처리할 수 있는 고도의 동시성, 신속성 및 확장성이 뛰어난 처리 네트워크, 매우 효율적인 데이터 서비스 레이어(data services layer) 뿐만 아니라 처리 오버헤드(processing overhead)를 최소화하는 알고리즘들 및 맞춤형 모듈들(bespoke modules)을 구현하는 고유한 디자인 때문이다.

[0138] 설명된 성능 특성들(performance characteristics)은 컴퓨팅 하드웨어의 특정 부분에서 더 많이 수행되는 규모 확장(scaling up)에 주로 타켓되므로, 운영 코스트들(running costs) 및 전력 소비(power consumption)에서 상당한 감소를 야기한다. 그러나, 디자인은 단일 시스템에 한정되지 않는다; Tereon 시스템은 다수의 장치들 상에 동시에 실행할 수 있는 각 서비스를 이용하여 수직 및 수평적으로 엄청나게 스케일 아웃(scaling out)할 수 있다.

[0139] 단일 시스템 또는 서버 상에 높은 레벨의 성능을 얻기 위해서, 시스템은 불필요한 직렬화(serializations)를 피하고, 불필요한 스트림 처리(stream processing)을 피하고, 불필요한 메모리 카피들(memory copies)을 피하고, 사용자로부터 커널 모드(kernel mode)로의 불필요한 전환(transitions)을 피하고, 프로세스들 간의 컨텍스트 스위치를 피하고, 랜덤 또는 불필요한 I/O를 피함으로써, 이것의 처리 오버헤드(processing overhead)를 가급적이면 최소화한다. 시스템이 올바르게 작동할 때, 이것은 해당 시스템 상에 매우 높은 레벨의 트랜잭션 성능(transactional performance)을 얻을 수 있다.

[0140] 기존 모델에서 서버 A가 요청을 수신한다. 그런 다음, 이것은 서버 B에 대한 쿼리(query)를 구축하고 직렬화하며 즉시 서버 B로 해당 쿼리를 전송한다. 그런 다음, 서버 B는 (필요한 경우) 해당 쿼리를 해독하고, 역질렬화하고(deserialize), 해석한다. 그런 다음, 이것은 응답을 생성하고, 직렬화하고, 필요한 경우, 해당 응답을 암호화한 후 해당 응답을 서버 A 또는 다른 서버로 다시 전송한다. 커널 및 프로세스 컨텍스트 스위치들(kernel and process context switches)은 메시지 당 수십 개씩 발생하며, 단일 메시지는 다양한 형태로 여러 번 캐스팅

되고, 메모리는 많은 작업 버퍼들 사이에 복사된다. 이러한 커널 및 프로세스 컨텍스트 스위치는 처리되는 메시지 당 대규모 처리 오버헤드를 부과한다.

[0141] *통신 아키텍처(Communications architecture)*

[0142] Tereon은 시스템에 의해 처리되는 기존 방식 데이터(traditional way data) 및 통신을 재구성하여 이것의 처리량을 달성한다. 가능한 경우, Tereon은 커널에 의해 부과되는 처리 오버 헤드를 피하고, 표준 데이터 관리 모델들에서 종종 발생하는 보안 문제들을 피하기 위해서 운영 시스템 커널(operating system kernel)을 바이패스한다.

[0143] 시스템에서 각 데이터 활동은 데이터 서비스 인스턴스(214)를 통해 실행된다. 이것은 직접 데이터 플랫폼 액세스를 갖는(또는 직접 데이터 플랫폼 액세스가 가능한) 시스템의 유일한 컴포넌트인 규모가 축소된 서비스-지향 데이터 서비스 레이어(scaled out service-oriented data service layer)이다. 따라서, 시스템상의 모든 데이터 활동들은 반드시 이것을 통과해야 한다.

[0144] 데이터 서비스 레이어(214)는 별도의 전용 판독 및 기록 액세스 채널들(226)을 통해 데이터 저장 레이어(220)와 통신한다. 데이터 저장 레이어(220)는 그 자체가 적어도 하나의 분산 데이터베이스를 포함하는 코어 데이터베이스 스토어(224)를 통해 구현된다. 이러한 데이터베이스들은 ACID 보증을 제공할 필요가 없다; 이것은 데이터 저장 레이어에 의해 관리된다.

[0145] 데이터 저장 레이어(220)에 대한 모든 기록들은 모든 데이터 변경이 인과 관계를 유지하기 위해 시리얼 빠른 시퀀스(serial rapid sequence)로 진행되면서 단일 공유 트랜잭터(single shared transactor)에 의해 관리되며, 이를 통해 모든 데이터 변경이 인과 관계를 유지하기 위해 시리얼 빠른 시퀀스(serial rapid sequence)로 진행된다. 트랜잭터 디자인(transactor design)은 데이터 트랜잭터 클러스터(data transactor cluster; 222)로서 자신을 제시하는 핫 백업 리던던시 모델(hot backup redundancy model)을 사용한다. 하나의 트랜잭터가 임의의 이유로 실패하거나 멈추는 경우, 다른 트랜잭터들 중에서 어느 하나는 즉시 인계 받을 것이다.

[0146] 데이터 플랫폼은 모든 데이터 도메인들에 대한 분할(partitioning)을 지원하지만, 그 지원은 도면(figure)에 도시되지 않았다. 임의의 케이스에서 단일 데이터 저장 레이어(무제한 데이터 노드들에 의해 백업됨(또는 지원됨))가 금지되거나 그렇게 해야하는 규제 이유들(regulatory reasons)이 있는 경우, 데이터는 서로 다른 트랜잭션들을 이용하여 서로 다른 데이터 클러스터들에 저장하기 위해 명령적이거나 선언적인 방법들을 통해 분할될 수 있다. 예를 들어, 사이트는 네 개의 데이터 플랫폼들(four data platforms)-지리적 또는 사법적 기준(geographic or jurisdictional criteria)에 따라 또는 1-5로 시작하는 계정은 하나로 6-0은 다른 계정으로 고객들을 분할함-있다(a site may have four data platforms, partitioning customers by geographic or jurisdictional criteria, or for accounts starting with 1-5 to go in one, 6-0 in another). 이것에 대한 처리 결과들이 있지만, 이것은 플랫폼에 의해 지원된다.

[0147] 도 3은 데이터 서비스 레이어(214)로 및 그로부터 통신을 라우트하는 통신 레이어(204)를 통한 통신을 도시한다. 모듈(350)이 다른 모듈(360)과 통신할 필요가 있는 경우, 이것은 먼저 프록시(370)과 연결을 개시하고, 단계 302에서 클라이언트 인증서(client certificate)를 인증하고, 단계 304에서 프록시 인증서(proxy certificate)가 구축시에 유효하고 신뢰되는지 체크한다. 모듈(350)은 단계 306에서 메시지를 프록시(370)에 전달한다. 프록시(370)는 단계 308에서 타겟 모듈(360)과 상관 연결(correlating connection)을 설정한다; 이것은 먼저 단계 308에서 자신을 인증하고, 단계 310에서 모듈의 인증서(module's certificate)가 유효하고 신뢰되는지 검증한다. 프록시(370)는 단계 314에서 모듈의 응답을 수신하기 전에, 단계 312에서 이니시에이터(initiator; 모듈(350))의 확인된 세부사항(또는 세부정보)들을 전달한다. 프록시(370)는 단계 316에서 타겟(모듈(360))의 세부사항(또는 세부정보)들 및 이것의 응답을 반환한다. 이것은 프록시(370)를 통해 모듈(350) 및 모듈(360)간의 통신 채널을 확립하고, 두 모듈 모두 높은 신뢰도로 서로 인증되고 식별되며, 필요한 경우 모든 통신 및 데이터가 암호화된다. 프록시(370)는 단계 318에서 모듈(350)으로부터 메시지를 단계 320에서 타겟 모듈(360)로 중계하고, 단계 322에서 타겟 모듈의 응답을 단계 324에서 모듈(350)로 중계한다.

[0148] 이러한 연결들은 발신자 및 수신자의 인증서의 세부사항(또는 세부정보)들에 기초하여 세션 공유 및 연결 유지(keep-alive)를 사용한다(예를 들어, 모듈(350)은 프록시(370)을 통해 타겟 모듈(360)에 대한 연결을 "클로즈(close)"할 수 있고, 실제로 새로운 종단간 연결(end-to-end connection)을 구축하지 않고 리오픈(reopen)할 수 있음. 연결은 임의의 다른 회로에서 절대 공유되지 않음). 통신 프록시(370)는 HTTP 게이트 웨이 또는 일부 다른 적절한 모듈 또는 컴포넌트일 수 있다.

[0149] 이러한 아키텍처는 주로 대량의 메모리 사용과 함께 상당한 성능 코스트(performance cost)를 발생시킨다. 모듈(350)이 타겟 모듈(360)과 통신하기 위해서 전통적으로 이것은 타겟 모듈(360)에 이것을 전달하기 전에 페이로드를 직렬화하고, 페이로드를 암호화하고, 이것을 프록시(370)에 스트림하고(여기서 프록시(370)는 페이로드을 해독할 수 있음), 컨텐츠를 역직렬화 및 해석하고, 페이로드를 재직렬화하고, 타겟 모듈(360)에 대해 이것을 암호화할 필요가 있다. 타겟 모듈(360)은 컨텐츠를 해독하고, 역직렬화하고 해석할 수 있다.

[0150] Tereon은 평균 및 최대 레이턴시(latency)를 줄이고, 메모리 로딩을 줄이고, 상용 하드웨어상 단일 플랫폼 성능을 향상시키기 위해서 여러 가지 기술들을 사용한다. 이것은 마이크로 서비스의 배포 이점들(deployment benefits), 유지 보수, 및 보안 모두를 유지하면서 모놀리식(monolithic) 인-프로세스 성능(in-process performance)을 달성한다. 이것은 이러한 시스템이 제공해야하는 높은 레벨의 보안 및 제어를 손상시키지 않고 수행한다.

[0151] Tereon은 도 3에 도시된 바와 같이 통신 레이어를 통해 배치된 메시징 모델(batched messaging model, 일괄처리된 메시징 모델)을 사용할 수 있다. 단계 306에서 모듈(350)으로부터 프록시(370)로 전달된 메시지와 같은 전달된 각 메시지는 메시지 배치(a batch of messages)일 수 있다. 그러나, Tereon은 이보다 훨씬 더 나아갈 수 있다.

[0152] 배치된 메시징(일괄처리된 메시징) 외에도, 도 4는 두개의 모듈들의 서버들이 그것들 간의 공유 메모리 채널을 협상하기 위해서 프록시 모듈(맞춤형 핸드-오버 모듈)을 통해 서로 통신하는 지를 도시한다. 단계 402 내지 412는 도 3에서 단계 302 내지 312와 유사하며, 필요한 경우, 서비스의 속성들은 그것들이 클라이언트 요청과 매치하는지 확인하기 위해 체크되고, 이는 단계 302 내지 312에서 발생할 수도 있다.

[0153] 모듈(450) 내지 모듈(460) 인스턴스는 TLS, 또는 전통적인 TLS HTTPS 뿐만 아니라 호출자 트랜잭션들에 대한 HTTP 게이트웨이의 사용자-모드 및 제로 카피를 함께 최적으로 사용할 수 있다.

[0154] 소스 모듈(450) 및 목적지 모듈(460)이 로컬인 경우, 단계 402 내지 412로부터의 프록시(470)을 통해 연결을 설정한 후, 발신자 및 수신자는 공유 메모리를 통해 서로 다이렉트 연결을 선택적으로 요청하고, 이 선택적인 요청으로 이 방법은 도 3에 설정된 방법과 다른 것이다. 발신자 및 수신자가 서로 다이렉트 연결을 요청하는 경우, 협상 후에, 공유 채널은 단계 414에서 모듈(460)으로부터 프록시(470)로, 단계 416에서 프록시로부터 모듈(450)로 전달되고, 이 시점부터 두 개의 모듈들은 세마포르들 및 공유 메모리를 다시 사용하는 직접 처리 메커니즘(direct process mechanism)을 사용한다. 이는 단계 418, 420, 422 등에서 모듈(450) 및 모듈(460) 간의 메시지들에 의해 예시된다.

[0155] Tereon 모델에서 서버(450)는 태스크에 대해 최적으로 기본 메모리 버퍼들(native memory buffers) 내 복수의 요청들을 일괄처리하고(batch), 서버(460)에 대한 메시지를 큐잉하고(queues), 세마포르(semaphore)를 트립한다(trip). 서버(460)는 플래그들(flags)을 체크하고, 직접적으로 공유된 메모리를 처리하고, 공유된 메모리에 응답한다. 연결은 발신자 및 수신자의 인증서의 세부사항(또는 세부정보)들과 통신을 위한 세마포르들(semaphores) 및 공유된 메모리에 기초하여 연결 유지 및 공유된 메모리(keep-alive and shared memory)를 사용한다.

[0156] 상술한 방법을 사용하여, 통신은 단일 발신자 목적지(single-caller destination), ACL-제어(ACL-controlled), 보안에 대한 직렬화 및 스트리밍의 오버헤드(기계 내에 이것이 포함되는 경우)를 회피할 수 있다. 이것은 암호화가 필요하지 않다; 연결은 유효성이 검증되고, 인증되고, 셋업 시 허가되고, 강탈될 수 없으며, 적절한 경우, 프로세스들은 적절한 경우 대규모의 독점적 메모리 구조(wholesale, proprietary memory structures)를 공유할 수 있다.

[0157] 프록시(470) 및 Tereon 코드 모듈들(450 및 460) 모두는 가능한 경우, 제로 카피 네트워킹(zero-copy networking) 및 사용자-모드 네트워킹(user-mode networking)을 지원한다(필수 RCP/IP 라이브러리로 컴파일될 때, HTTP 프록시는 네트워크 패킷들에 대한 커널 컨텍스트 스위치들의 상당한 코스트(the significant cost of kernel context switches)를 피하는 솔류션을 제공함). 이것은 프록시(470) 및 Tereon 코드 모듈들이 사용할 수 있는 네트워크 드라이버 특정 코드를 통해 용이하게 된다. 이것은 작은 패킷 요청 및 응답에 대한 메모리 사용을 최소화한다; 이것들은 엄청나게 방대한 Tereon 동작들- 여기서 대부분의 동작들은 단일 TCP 패킷에 알맞을 수 있음 -을 구성한다.

[0158] 도 4a는 Tereon 시스템이 Tereon 시스템의 임의의 두 개의 컴포넌트들(예를 들어, Tereon 내의 기능을 제공하는 HTTP 게이트웨이(406a) 및 마이크로서비스들(410a)) 사이에서 데이터를 효과적으로 교환하는데 사용되는 공유

메모리를 사용할 수 있는 맞춤형 세마포르 핸드-오프 모듈들(408a)의 세트를 구현하는 방법을 도시한다. 도 4a에서 데이터 서비스 레이어(214)는 마이크로서비스들(410a)에 의해 구현된다. 그러나, 마이크로서비스들은 모든 종류의 서비스 모듈을 나타낼 수 있다.

[0159] 네트워크 스택(404a)(루프백 가상 장치를 포함함)은 연결 서버(connecting server; 402a)로부터 요청을 수신 및 어셈블하고, 이것을 사용자-모드 타켓 메모리로 복사하는 대신에, 수신자- 이 경우에는 HTTP 게이트웨이(406a)-에게로 메모리 승인의 소유권(ownership of the memory grant)을 단순하게 건넨다. 이것은 메모리 대역폭 포화(memory bandwidth saturation)가 발생하기 시작하는 매우 큰 로드(예를 들어, 초당 수백만개의 요청)에서 주로 유용하다.

[0160] 커스텀 Tereon 업스트림 HTTP 게이트웨이 모듈(custom Tereon upstream HTTP gateway module; 406a)은 로컬 인스턴스들(HTTP 게이트웨이 인스턴스와 관련됨, 일반적으로 각 컨테이너 또는 각 물리적, 논리적, 또는 가상 기계 상에 HTTP 게이트웨이 인스턴스가 있음)이 게이트웨이로부터 모듈로 프록시 메모리에 대한 메모리 전달 및 공유 메모리를 사용하기 위한 옵션을 허용하고, 그 반대의 경우도 해당 업스트림 연결(또는 접속)을 허용한다. HTTP 게이트웨이(406a)가 기존 메커니즘을 통해 요청을 직렬화하고 이것을 전달하는 대신에, 공유 메모리 업스트림 제공자(shared memory upstream provider)를 위해 구성될 때 HTTP 게이트웨이(406a)는 수신자에게로 전달하는 공유 메모리를 사용한다.

[0161] 이 경우에, 공유 메모리는 다른 HTTP 게이트웨이, HTTP 게이트웨이 인스턴스 또는 다른 요소를 프록시로써 사용하여 설정되었을 수 있다. HTTP 게이트웨이를 사용하는 것은 특히 효율적일 수 있다.

[0162] 운영 시스템 커널에 의해 제공되는 통신 훅들(communications hooks)을 사용하는 대신에, 각 데이터 교환 모듈은 커널을 바이패스(또는 우회)한다. 이것은 커널 오버헤드를 피하여 시스템의 처리량(throughput)을 증가시키고, 데이터가 커널에 의해 제공되는 서비스들로/로부터 전달될 때 발생할 수 있는 불안정 영역들(areas of insecurity)을 다룬다. 예를 들어, Tereon 내의 모듈은 시스템 컴포넌트로부터 직접 데이터 서비스 레이어(214)로, 및 데이터 서비스 레이어(214)로부터 시스템 컴포넌트로 데이터를 효과적으로 교환하기 위해 사용된다.

[0163] 이 아키텍처가 가져오는 이점의 다른 예는 HTTP 게이트웨이(406a)의 향상된 효율- HTTP 게이트웨이(406a)가 데이터 서비스 레이어(214) 또는 다른 컴포넌트들과 같은 마이크로서비스들(410a)에 대한 모든 입력 데이터(all of the incoming data), 및 마이크로서비스(410a)들 또는 데이터 서비스 레이어(214)로부터의 모든 출력 데이터(all of outgoing data)를 HTTP 게이트웨이(406a)로 핸드오버하게 하는 핸드-오프 모듈(408a)을 사용하여 달성됨 -이다. 기본 HTTP 게이트웨이들의 데이터 및 메시징 핸드-오프를 사용하는 대신에, 그 자체가 효율적이고, 공유 메모리를 또한 사용할 수 있는 세마포르 핸드오프 모듈은 데이터가 커널을 우회하여 데이터 레이서(214)로부터 HTTP 게이트웨이(406a)로, 및 데이터 레이어(214)로 직접 전달되게 한다. 이것은 시스템의 처리량을 증가시킬 뿐만 아니라; HTTP 게이트웨이들을 사용하는 시스템들에서 공통적인 취약점 영역들 중에서 하나를 보호한다는 점에서 추가 이점이 있다.

[0164] 공유 메모리 채널을 제공하는 모듈 또는 공유 메모리 채널과 통신하는 모듈은 요청을 배치(일괄처리)하고 직렬화하거나 역직렬화하고 분리할 수 있다. 해당 태스크를 수행하는 모듈은 해당 모듈의 기능 및 모듈이 정상적으로 작동할 때 발생하는 처리 오버헤드에 도달하게 된다. 예를 들어, 한 경우에, 자신이 많은 수의 메시지들(요청이거나 아닐 수 있음)을 수신하는 모듈은 수신자 모듈에 대해 그것들의 메시지들을 배치(일괄처리)하고 직렬화할 공유 메모리 모듈로 그것의 메시지들을 전달하는데, 배치(일괄처리) 및 직렬화의 오버헤드가 효율적이고 로드(load)에서 메시지들을 처리하는 다른 방법으로부터 해당 모듈을 막기 때문이다. 다른 경우에, 모듈은 공유 메모리 채널을 통해 해당 수신자에게로 배치(일괄처리)를 전달하기 전에 그것의 메시지들을 특정 수신자에게로 배치(일괄처리)하고 직렬화할 수 있다.

[0165] 또 다른 경우에, 메시지들을 수신자 모듈에게로 전달하는 모듈은 메시지들을 배치(일괄처리)하고 직렬화하기 위해서 공유 메모리 채널을 제공하는 모듈에 의존할 수 있지만, 배치된 메시지들(일괄처리된 메시지들)을 수신하는 모듈은 그 자신이 메시지들을 역직렬화 및 분리할 수 있다. 어떤 모듈이 배치(일괄처리) 및 직렬화, 또는 역직렬화 및 분리에 대한 태스크를 수행하는 지에 대한 질문은 어떤 선택이 모듈들이 수행하는 기능들을 위한 최적 성능 레벨을 제공하는 지로 요약된다. 배치 및 직렬화의 순서는 메시지 타입 및 통신 모듈들에 의해 제공되는 기능들에 의존할 것이다.

[0166] Tereon 은 웹 서비스로 가장하기 위해서 HTTP 게이트웨이(406a)를 사용하므로 네트워크 운영자들이 비표준 서비

스들을 차단하는 잠재적 문제들을 피한다. Tereon은 물론, 필요한 경우, 임의의 다른 서비스로 가장할 수 있으므로, 잘 알려진 네트워크 보안 구성(network security configurations)을 쉽게 작업할 수 있다.

[0167] 이러한 디자인에 따라, 시스템- 시스템은 사용 가능한 자원들을 이용하도록 디자인된 모듈들을 사용함 -은 전체 아키텍처에 걸쳐 이 모듈러 접근 방식을 수행하고, 가능한 경우 오버헤드를 피한다. 추가적인 예는 네트워크 스택(404a)에서 제로-카피 네트워킹 또는 사용자-모드 네트워킹을 지원하는 모듈들의 네트워킹 시스템- 가능한 경우 Tereon이 사용함 -이다. 이것은 네트워킹에 대한 커널을 사용하는 많은 오버헤드(heavy overhead)를 피한다. 또한, 모듈러 디자인은 Tereon이 시스템들의 다중 타입들- 유사 맞춤형 모듈들이 유사 기능을 제공하고, 각 운영 시스템 또는 하드웨어 구성에 맞게 맞춤화됨 -에서 동작하게 한다.

[0168] 도 3 및 도 4에 도시된 방식으로 중개자(intermediary)를 사용하는 것은 내부-기계 또는 외부-기계의 모든 통신들에 대한 중앙 집중식 제어 지점을 허용한다. 이것은 속도(rate) 및 보안 제어들, 모니터링 및 감사, 및 특수 규칙들 또는 재지시들에 대한 단일 제어 지점이다. 가동 중단 또는 심각한 위험들을 초래하지 않으면서, 이것은 심지어 시스템을 운영하는 동안에도 시스템을 유연하게 전개할 수 있다. 또한, 이것은 클라이언트 인식 또는 복잡성(any client awareness or complexity)없이 쉽게 로드-밸런싱(load-balancing) 및 리던던시들(redundancies)을 용이하게 한다.

[0169] 도 3의 모듈(350)가 타겟 모듈(360)에 통신하기를 원하는 경우, 중개자의 사용은 타겟 모듈(360)이 "n" 기계들에 걸쳐 로드-밸런싱되게 하고, 중개자를 단순히 재구성하는 대신에 모든 잠재적 클라이언트들을 재구성하지 않고 임의의 수 또는 기계들의 타입에 걸쳐 이동되게 한다.

[0170] 시스템은 두 개의 통신 당사자들이 서로의 키 교환을 상호 인증하는 기능을 제공하기 위해 생성된 PAKE(password authenticated key exchange) 프로토콜을 사용한다. 이것은 다른 잘 알려진 공용 키 교환 프로토콜들- 예를 들어, Diffie-Hellman 키 교환 프로토콜과 같음, 중간자 공격(man-in-the-middle attack)에 프로토콜들을 취약하게 만듦 -에 대해선 불가능하다. 올바르게 사용되는 경우, PAKE 프로토콜은 중간자 공격(man-in-the-middle attack)에 면역된다.

[0171] Tereon이 외부 장치 또는 서버와 같은 외부 시스템들과 통신하는 경우, 이것은 통신 시스템에 추가적인 레이어를 추가한다. 많은 키-교환 프로토콜들은 중간자 공격들에 이론적으로 취약하다. 통신이 두 개의 알려진 엔티티들 간에 있음을 확인하기 위해서 인증서 및 서명된 메시지들을 사용하여 연결(또는 접속)이 설정되면, 시스템은 제2 보안 세션 키를 설정하기 위해 PAKE 프로토콜을 사용하므로, 통신이 중간자 공격에 영향을 받지 않게 한다. 따라서, 통신은 TLS 세션 키를 사용하고, 그 다음 모든 통신을 암호화하기 위해서 PAKE 프로토콜의 세션 키를 사용할 것이다.

[0172] 침범할 수 없는 식별 스트링(inviolable identity string)을 가지는 장치들과 통신하는 경우에는, 필요한 경우 TLS는 생략될 수 있고(TLS can be dispensed with if necessary), PAKE 프로토콜이 메인 세션 키 프로토콜로 대신 사용된다. 예를 들어, 장치들이 사물 인터넷의 컴포넌트들의 세트를 형성하는 소형 하드웨어 센서들인 경우, 이것은 발생한다.

[0173] *통신 방법들(Communication methods)*

[0174] Tereon 데이터 서비스(214)는 조정 트랜잭터(coordinating transactor, 하나 이상의 트랜잰션들의 전체 또는 일부를 수행, 관리 또는 제어하는 장치 또는 모듈)를 통해 완전한 ACID 보증들을 제공하고, n+1 또는 더 큰 리던던시 및 선택적 다중-사이트 복제를 제공하는 그래프 기능을 갖는 키-값 저장소에 기반한다. 데이터 서비스(214)는 공유 메모리 기능 이외에, 제로-카피 기능 및 무제한 판독 스케일링, 인-메모리 캐싱(in-memory caching), 및 매우 높은 레벨들의 기록 성능을 제공하는 데이터-도메인 서비스에 캡슐화된다. 이것은 메모리 캐싱이 큰 가변 크기의 데이터 클러스터(variable sized data cluster)에서 유지된다. 매우 독특한 상황에서, 데이터 서비스는 키-값 저장소들을 직접 사용하기 위해 우회(또는 회피)될 수 있다.

[0175] 데이터 서비스(214)는 고 성능 기본 SQL 스타일 기능과 함께, 화폐 흐름 분석과 같은 기능을 지원하는 그래프 처리 기능을 제공한다. 매우 높은 성능 모듈 통신 아키텍처(플랫폼의 효율성 및 성능을 제공함)에 결합된 데이터 서비스(214)는 범용 서버 하드웨어(본딩된(bonded) 10 Gbps 네트워킹을 갖음)에 대한 테스트에서 초당 280만 트랜잭션들을 초과하는 매우 효율적인 디자인을 제공한다.

[0176] 다음과 같은 아키텍처 우선 순위들을 구현함으로써, 시스템은 시스템 내 및 시스템들 간에 전송되는 메시지들을 처리하기 위해 필요한 커널 및 처리 컨텍스트 스위치들의 수를 크게 줄일 수 있다.

[0177] a) 제로-카피 네트워킹은 네트워크 에지(network edge)로부터 서비스들로 전송 비용들을 최소화할 수 있다.

[0178] b) 사용자-모드 네트워킹은 네트워크 에지로부터 서비스들로 전송 비용들을 최소화할 수 있다.

[0179] c) 직렬화가 필요한 경우(주로 기계 또는 서버 경계들을 넘을 때), 고효율 직렬화는 SOAP(Simple Object Access Protocol)와 같은 높은 오버헤드 직렬화와는 대조적으로 프로토콜 버퍼들 또는 Avro로 사용된다. 이는 각 서버의 에지에서 추상화되어 성능 및 효율성 레벨이 낮음에도 불구하고 특정 서버가 인터넷을 통해 다른 대륙(continent)의 피어 서버(peer server)로 쉽게 통신할 수 있다.

[0180] d) 처리 컨텍스트 스위치들을 최소화하고, 임의의 특정 서버에 대한 캐시 일관성을 최대화하기 위한 요청을 배치(일괄처리)하려고 시도할 서버들은 구성 가능한 버퍼링 임계값(configurable threshold of buffering)이 있다. 예를 들어, 서버 A에 10,000 개의 요청들이 20 ms 기간 내에 도달하고, 플랫폼이 20 ms 버퍼 위도우를 목표로 하고, 해당 10,000 개의 요청들을 위해 서버 B의 지원이 필요한 경우, 10,000 개의 요청들을 단일 요청으로 수집한 다음, 세마포르를 프래깅(flagging)하여 서버 B에 대한 비동기 메시지를 대기시킨다. 그런 다음, 서버 B는 서버 A로 단일 응답을 제공하여 10000 개의 요청들을 신속하게 처리할 수 있다. 이는 최적 효율성 대 최대 응답 시간에 기반하여 구성할 수 있다.

[0181] 실제로, 커널 및 프로세스 컨텍스트 스위치들의 수를 줄이는 것은 플랫폼의 성능 레벨에 엄청난 개선을 초래했다. 메시지 당 많은 커널 및 프로세스 컨텍스트 스위치들이 발생하는 대신에, Tereon 모델은 통신되는 중인 메시지들의 배치(일괄처리)로 인해, 메시지들의 블록당 많은 커널 및 프로세스 컨텍스트 스위치들을 발생시킨다. 테스트는 이 모델을 사용하여 기존 모델 및 Tereon 모델 간의 성능 차이가 많은 작업 부하들에 대해 1 : 1000 및 그 이상이 임을 나타낸다.

[0182] 그러나, 모듈들 및 그것들의 이점은 단일 시스템들로 제한되지 않는다. 예를 들어, 서버 A 및 서버 B가 별개의 기계들에 존재하는 경우에도, Tereon 시스템은 효율적인 직렬화 및 배치(일괄처리)를 여전히 사용할 것이다. 이는 선택적인 제로-카피 또는 사용자-모드 네트워킹과 결합되는지 여부에 관계없이, Tereon 모델은 네트워크 및 처리 성능을 크게 향상시킨다.

[0183] 테스트는 이러한 디자인 요소들이 초고속 네트워크 와이어(예를 들어, 바운딩된 10 Gps)를 통해 초 당 수백만개 및 초 당 수천만 개의 메시지 요청 및 응답 왕복(배치(일괄처리), 공유 메모리 모드에서)으로 로컬 서버 대 서버 운영들을 시연했음을 나타내었다.

[0184] 이러한 모든 트랜잭션들이 실시간으로 처리되고, 즉시 조정될 수 있기 때문에, 특히 뱅킹, IoT, 의료, ID 관리, 운송, 및 정확한 데이터 처리가 필요한 다른 환경들에 많은 장점들이 있다. 특히, 이러한 시스템들은 현재 실시간으로 트랜잭션들을 조정하지 않는다. 그 대신에, 트랜잭션들은 일정 기간 이후에, 때로는 일괄적으로(또는 배치(batch)되어) 조정된다. 예를 들어, 그것은 보통 금융 트랜잭션들이 일괄적으로(또는 배치되어) 처리되는데, 별도의 조정 프로세스들이 시간들 이후에 실행하기 때문이다. Tereon 시스템을 사용하여, 이것은 은행들이 이전에 불가능했던 방식에서 실시간으로 모든 금융 트랜잭션들을 조정할 수 있다는 것이다. 따라서, 이것은 은행들이 정의에 따라 모든 트랜잭션들이 그것들이 처리될 때 조정되었기에 정확하게 조정될 수 없거나 아직 조정되지 않는 금융 트랜잭션들을 커버하기 위해서 조정 계정들(reconciliation accounts, 또는 조정 계좌들)을 보유할 필요가 없다는 것이다.

[0185] *트랜잭션들 및 데이터 분할*

[0186] Tereon 시스템에서 모든 원자 활동들(atomic activities)은 트랜잭션들이다 - 트랜잭션들에 대한 ACID 보증들을 백킹하는 임의의 시스템의 기본적이 요구 사항이므로, 그것들은 전체적으로 실패하거나, 전체적으로 성공한다. 이 섹션은 이것이 어떻게 수행되는지 간단하게 설명하고, Tereon이 트랜잭션들에 대한 ACID 보증들을 달성하는데 있어 분할의 영향을 완화하기 위해서 트랜잭션들 및 데이터 분할에 취한 접근 방식의 세부사항(또는 세부정보)들을 설명한다.

[0187] 상술한 바와 같이, Tereon 플랫폼 내 각 데이터 활동은 Tereon 데이터 서비스 인스턴스(214)- 그 자체가 마이크로서비스들(410a)의 세트로써 동작할 수 있음 -를 통해 실행된다. 이는 직접 데이터 플랫폼 액세스가 있는 유일한 시스템의 컴포넌트인 확장된 서비스-지향 시스템(scaled out service-oriented system)이므로, 모든 데이터 활동들은 이것을 통과해야 한다. 이러한 데이터 서비스들은 항상 일관된 판독 데이터를 가지는 인스턴스 캐시된 데이터 MVCC(Multiversion Concurrency Control)를 이용하여 상이한 데이터 서비스 인스턴스들을 통해 시스템 내의 병렬 트랜잭션들이 수행될 수 있도록 확장된다.

[0188] 데이터 활동들은 원자 메시지들을 통해 데이터 서비스 인스턴스로 전체 데이터 잡(job)을 포함하는 메시지와 함께 발생한다; 예를 들어, 잡은 여러 개의 레코드들 및 속성들을 판독하거나, 종속 데이터(dependent data)에 따라 데이터를 업데이트 또는 삽입하거나, 태스크들의 조합을 포함할 수 있다. 데이터 서비스 인스턴스는 모든 백킹(backin), 트랜잭션 데이터 저장소들(all backing, transactional data stores)에 걸쳐 2-단계 커밋 트랜잭션(two-phased commit transaction)으로 잡을 실행한다.

[0189] Tereon 모델은 다음의 기술들을 통해 데이터 일관성을 보증한다:

[0190] a) 임의의 판독 데이터의 세트는 버전 ID를 운반(또는 전달)한다.

[0191] 모든 기록들(업데이트하고 종속 삽입함)이 버전 ID가 낙관적인 트랜잭션과 같은 모든 관련 데이터에 대해 최신인지 검증한다. 이는 다양한 계정 속성들(예를 들어 허가들(permissions), 잔액(balance, 또는 밸런스), 및 통화 데이터(currency data))을 획득하기 위해서 소스가 3 개의 레코드들을 판독하는 경우, 이 데이터의 클러스터가 일관된(또는 통일된)버전 ID가 있다는 것을 의미한다. 그러한 값들 중에서 어느 것이라도 업데이트되거나, 종속 데이터가 기록되는(예를 들어, 금융 이체(financial transfer) 경우, 버전 ID는 현재 버전(또는 최신)으로 다시 확인되고, 기록이 다른 경우- 통화 가정(currency assumptions)이 변경되거나, 환율이 수정되었다고 언급 - 기록은 전체적으로 완전히 실패한다. 적절한 경우, 다운스트림 서비스는 데이터가 임의의 중요한 방식으로 트랜잭션을 변경하는지 여부를 재판독, 및 평가한다. 그렇지 않으면 트랜잭션은 다시 제출된다. 다시 트랜잭션이 실패하는 경우, 이것은 구성 가능한 재시도 횟수가 초과되고, 심각한 실패가 발생할 때까지 반복된다. 심각한 실패는 정상적인 상황에서 극히 드물다.

[0192] 실제 시나리오들의 대다수에서, 심지어 엄청난 트랜잭션 용량들 및 계정 다이버시티(account diversity)에 걸쳐서도, 실패된 낙관적인 트랜잭션은 결코 발생하지 않는다. 드문 경우지만, 데이터는 결코 손상되지 않고, 최소한의 처리 오버헤드가 있다. 이 MVCC/낙관적인 모델은 사용된 플랫폼이 (예외적인 상황에서 요구될 수 있는 규제 삭제 이상의)영구 히스토리 데이터베이스(perpetual history database)인 경우, 삭제된 레코드들에 대해서도 완전하게 보호한다.

[0193] b) 주어진 데이터 파티션(이는 데이터 서비스의 수평 확장과는 별도의 개념임)을 위해 플랫폼으로의 기록.

[0194] 많은 데이터 서비스 인스턴스들은 하나의 데이터 파티션으로부터 판독 및 하나의 데이터 파티션으로 기록할 수 있고, 단일 데이터 서비스 인스턴스는 다중 데이터 파티션들로부터 판독 및 다중 데이터 파티션들로 모두 저장할 수 있다. 모든 판독들 및 기록들은 필요에 따라 하나 이상의 리던던트 운영 백업(redundant operating backup)과 함께 단일 마스터 트랜잭터 인스턴스(single master transactor instance; 222)를 통해 발생한다. 그러나, 단일 인스턴스만은 항상 활성화한다. 이는 거래적(transactional) 및 인과적(causal) 유효성이 모든 상황들(예를 들어, 네트워크 분열 중에, 또는 짧은 통신 지연 중에 예를 들어, 스큐(skew)가 발생하지 않음) 하에서 유지된다는 것을 보장한다. 이 트랜잭터는 모든 낙관적인 트랜잭션들이 유효한지 여부를 확인하고, 해당 인스턴스에 대해 문맥적으로 중요한 만큼 업데이트된 최신 정보로 데이터 서비스 인스턴스들에서 캐시 관리자들을 지속적으로 리프레시한다.

[0195] c) 선택적인 데이터 분할

[0196] 단일 트랜잭터로 제약되는 것은 매우 큰 Tereon 인스턴스들에 대한 확장성을 잠재적으로 제한할 수 있다(단일 조직이 지역별 등으로 다중 Tereon 인스턴스들을 관리할 수 있다는 것을 이해). 데이터 분할은 Tereon 데이터 서비스 클러스터가 도메인별로 구성된 Tereon 규칙들에 따라 트랜잭터들(222) 또는 데이터 저장소들(224)에 걸쳐 데이터를 분할할 수 있다는 개념이다. Tereon 플랫폼은 이종, 다중-컴포넌트 해싱 전략으로 다음의 분할 규칙들을 현재 지원한다:

[0197] i) 주어진 요소 또는 임의의 상위 요소의 데이터를 대상으로 해싱(예를 들어, 상위 레코드(parent record)에 따라 세부사항(또는 세부정보) 해시). 고성능 해싱은 카디널리티(cardinality)가 파티션들의 수와 같다.

[0198] 시스템은 재조정(rebalancing)을 현재 제공하지 않기 때문에, 미래 구현에서 재조정이 제공될지라도, 현재 구현에서 해싱은 전면에 있어야 한다(원 날짜 및 시간에 의한 해싱을 포함하는 다중-부분 규칙(multi-part rule)을 사용하여 파티션들이 현재 여전히 추가될 수 있지만).

[0199] ii) 주어진 요소 또는 임의의 상위 요소(예를 들어, 열거된 지리적 지역별로. A-K 또는 L-Z 등 성별로. 통화별로. 기타 등.)의 타겟된 데이터의 해싱이 구성된 데이터.

[0200] 알파뉴메릭(alphanumeric), 유니코드(unicode), 및 다른 특징 코드 범위들(other character code ranges, 또는

다른 문자 코드 범위들), 정수 범위들(integer ranges), 부동 소수점 범위들(floating point ranges), 및 열거된 세트들을 지원하는 해싱이 타겟된 데이터.

[0201] iii) 상기의 조합들(Combinations of the above)

[0202] 구현 예에서, 두 문자들, A 및 B는 전체 지리적 지역에 걸쳐 해당 지역의 두 개의 부분들을 나타내는 숫자들 1 및 2로 공통된 두 개의 개별 데이터 세트들을 나타낼 수 있다. 예를 들어, 단일 파티션 규칙은 지리적 지역과 같은 최상위 레벨 파티션들 1AB 및 2AB 사이의 분할을 지원하고, 그런 다음 계정 번호 해시를 통해 A 및 B 서브-파티션들 간의 분할을 더 지원할 수 있다.

[0203] d) 단일 데이터 서비스 인스턴스를 통해 수행되는 단일 잡은 다중 데이터 파티션들을 걸쳐 다중 트랜잭터들에 의해 완료되고, 많은 수의 데이터 스토리지 노드들에서 지속될 수 있다.

[0204] 이것은 명백한 데이터 무결성 복잡성들(data integrity complexities)을 나타낸다. 그러나, 데이터의 무결성은 트랜잭션의 모든 구성 요소들이 단일 2-단계 커밋 랩퍼(single two-phased commit wrapper)에 바인드됨에 따라 보장된다. 모든 영구적인 노드들 및 액터들에 대한 트랜잭션 전체는 전체적으로 완료하거나 실패하고, 모든 동일한 버전의 보증들을 제공한다.

[0205] 아키텍처 디자인들의 이러한 컨플루언스(confluence, 또는 융합)의 최종 결과는 시스템이 완전히 트랜잭션적으로 안전하고, 매우 리던던트하고(또는 중복되고), 수평적으로나 수직적으로 매우 확장이 가능하다는 것이다. 기록 트랜잭션들(대부분의 시나리오들에서 작은 퍼센트의 활동을 포함함)은 파티션당 단일 트랜잭터의 트랜잭션적인 필요성에 의해 제한될 수 있지만, 규칙 기반 분할의 부가- 특히 우수한 데이터 요소들 -는 심지어 두 갈래로 나뉘는 인스턴스들을 고려하기 전에, 개념적으로 무제한 수준으로 시스템을 확장할 수 있는 엄청난 유연성을 제공한다.

[0206] *Tereon 데이터 저장소 구현*

[0207] Tereon 인프라스트럭쳐는 초당 1,000,000 건 이상의 ACIS 보장 트랜잭션들을 처리할 수 있다. 이것은 개별 판독 및 기록 액세스 채널들(226)과 함께 스토리지 레이어에 대한 고성능 키/값 분산된 데이터베이스를 사용하여 분산된 데이터베이스 또는 데이터베이스들(224)의 최상부 상에 데이터 저장소 레이어(220)를 추상화 또는 다른 방법으로 구현함으로써 달성된다(이것은 Tereon 데이터 서비스를 통해 추상화로부터 스토리지 레이어에 대한 직접 데이터베이스 사용에 이르기까지 모든 깊이 레벨일 수 있음). 데이터 저장소의 Tereon의 사용 및 구성은 독특하다.

[0208] 데이터 서비스 레이어는 이것의 맞춤화 데이터 교환 모듈들을 통해 데이터 저장소 레이어와 통신한다. 데이터베이스들 그 자체는 전혀 임의의 ACID 보증들을 제공할 필요가 없다- 이는 데이터 저장소 레이어(220)에 의해 처리됨 -. 이것들이 기록 프로세스들을 상당히 늦추므로, 그래프 기능들을 제공할 필요도 없다. 데이터 저장소 레이어(220)는 이종 데이터 레이어들에 대한 인터페이스를 제공하고, 시스템의 다른 부분들이 필요로 하는 인터페이스 기능을 제공한다. 따라서, 기록 기능은 빠른 셀 및 열 구조를 제공하는 반면, 판독 인터페이스는 그래픽 인터페이스를 제공하여 이것이 분산된 데이터 저장소를 마이크로 초로 탐색하게 한다.

[0209] 데이터 저장소 레이어는 코어 데이터 저장소 데이터베이스들 상부의 SQL 인터페이스 및 그래픽 인터페이스 레이어를 제공하고, Tereon을 구분하는(또는 구별하는) 많은 중요한 아키텍처 장점들을 제공한다. 각각의 클라이언트 인스턴스(Tereon 데이터 서비스 인스턴스(214))는 해당 인스턴스에 대한 모든 핫 데이터의 캐싱된 표현들을 포함한 인-메모리/인-프로세스 데이터베이스 엔진을 호스트한다. 결과적으로, 인스턴스는 데이터베이스 엔진 및 모든 현재 트랜잭션적인 데이터의 캐싱된 표현들, 각 현재 트랜잭션의 상태, 및 해당 인스턴스가 동작중인 기계 또는 기계들의 다른 빠른 메모리 또는 이것의 RAM의 부분 내의 인스턴스의 현재 상태와 관련된 모든 다른 정보를 호스트한다.

[0210] 이것은 Tereon 데이터 서비스가 엄청난 레이트(enormous rate)로 대부분의 판독-지향적 작업들을 용이하게 하고(초당, 인스턴스당 수백만개의 분리된 쿼리들, 여기서 핫 관련 데이터는 로컬로 캐시됨), 달성될 수 있는 성능 레벨들 이상의 중요도(magnitudes, 또는 규모)는 외부 데이터베이스 시스템들에 대한 외부 또는 오프-기계 요청들을 직렬화하고 수행하는 것이었다. 데이터가 인-프로세스 캐쉬에 없을 때, 이것은 키 값 저장소로부터 검색되다.

[0211] MVCC 버전 시스템은 동시성을 관리하는데 사용되고, 데이터 레이어의 속성은 데이터가 절대 삭제되지 않는다는 것이다(규정 준수를 위한 강제 삭제들 이외에)- 시스템은 데이터 시스템의 수명 동안 모든 레코드 변경의 전체

히스토리를 보유함 -. 이것은 “as of” 쿼리, 및 모든 플랫폼 변경 감사와 같은 간단한 동작들을 수행한다.

[0212] 데이터 레이어의 기록 구현은 모든 데이터 변경을 일련의 빠른 시퀀스로 처리해야하는 단일 공유 트랜잭터를 사용한다. 이것은 트랜잭션들이 유효하고, 일관되며 대부분의 데이터베이스 플랫폼들에서 부담되는 가중치인 변경 동시성 오버헤드를 최소화하게 한다. 트랜잭터 디자인은 핫 백업 리던던시 모델을 사용한다. 트랜잭터 프로세스들이 변경되면, 이것은 모든 활성 쿼리 엔진들(이러한 경우, Tereon 데이터 서비스에 존재함)을 공지하고, 그것들은 적절한 그것들의 인-메모리 캐쉬들을 업데이트한다.

[0213] 디자인은 데이터 저장소의 사이즈에 관계없이 판독들, 기록들 및 검색들에 대한 마이크로-초 레이턴시를 제공한다. 또한, 이것은 이것의 동작에 영향을 주지 않고 컴포넌트들이 업데이트 및 교체될 수 있는 모듈러 구성을 제공한다. 이 데이터 저장소는 기본 구현으로부터 추상화되고, Tereon 데이터 서비스에서 다른 저장소들로 대체될 수 있다.

[0214] 데이터 저장소 레이어가 비관적인 ACID 보증들(226)과 함께 동작하도록 설정된 경우, 그것은 다음 트랜잭션으로 넘어가기(또는 이동하기) 전에 레코드를 기록했는지 확인하기 위한 추가 단계를 수행하고, 그런 다음 이것은 짧은 지연을 추가하지만, ACID 일관성 및 데이터 무결성의 절대적인 보증을 제공한다.

[0215] 이 디자인의 장점은 이것이 ACID 보증들을 제공한다는 것인데, 데이터 레이어가 레코드를 기록하고 트랜잭션을 완료했음을 확인할 때까지 어플리케이션 레이어가 진행할 수 없기 때문이다.

[0216] 이것은 예를 들어, 뱅킹, 결제, 및 인과관계를 보호해야하는(또는 유지해야하는) 다른 트랜잭션 타입들에서 최종 일관성에 의해 발생된 문제들이 제거되는 것을 의미한다. 또한, ACID 보증들로 디자인함으로써, 은행 시스템들이 불일치되는 프로세스들을 발견할 때 모든 부족분을 커버하기 위한 조정 계정들에 대한 필요성은 제거된다. 실시간 처리는 최종 일관성 시스템들 상에서 조정 프로세스들이 발생하는 시간-지연도 제거된다는 것을 의미한다.

[0217] 이 플랫폼의 디자인은 범용 하드웨어 상의 매우 높은 레벨들의 리던던시 및 안정성, 및 뛰어난 확장성(수직 및 수평적으로 모두)을 제공한다. 트랜잭터 시스템의 가능한 한계들(또는 제한들)에 대한 이론적인 우려들은 이것들의 한계들을 극복하기 위해서 데이터 서비스에서 분할 플랫폼을 구축되었으나, 대다수의 시나리오들 하에서는 해당 플랫폼을 사용할 필요가 없다.

[0218] *룩업/디렉토리 서비스*

[0219] Tereon 시스템은 사용자 또는 장치(218)가 어떤 서버에 등록되는지, 또는 특정 기능, 리소스, 설비, 트랜잭션 타입, 또는 다른 타입의 서비스를 제공하는 서버를 식별하는 시스템에서 크리덴셜들 및 정보의 디렉토리인 디렉토리 서비스(216)를 가진다. 디렉토리 서비스는 사용자(218)의 다중 인증 방법들이 가능하게 하는데, 이것은 특정 사용자와 관련된 다양한 상이한 타입들의 크리덴셜들을 저장하기 때문이다. 예를 들어, 사용자(218)는 자신의 모바일 번호, 이메일 주소, 지리적 위치, PANs(primary account numbers) 등을 사용하여 인증될 수 있고, 매번 인증할 필요가 없도록 데이터를 캐시한다.

[0220] 디렉토리 서비스(216)은 기본 서비스들(underlying services), 서버들, 및 실제 사용자 계정들로부터 사용자의 인증 ID를 분리하는 추상화 레이어를 제공한다. 이것은 사용자(218) 또는 머천트가 서비스에 액세스하기 위해 사용하는 크리덴셜들 및 Tereon이 서비스 자체를 수행하기 위해 필요한 정보 간의 추상화를 제공한다. 예를 들어, 결제 서비스에서 디렉토리 서비스(216)은 서버 주소와 함께 아마도 통화 코드(currency code), 및 모바일 번호와 같은 인증 ID를 간단하게 링크할 것이다. 사용자(218)가 은행 계정(또는 은행 계좌)을 보유하는지 여부, 또는 사용자(218)가 은행과 뱅킹하는지 여부를 결정하는 방법은 전혀 없다.

[0221] 시스템 아키텍처는 Tereon이 기존 시스템들의 범위를 간단하게 넘어서는 다양한 새로운 서비스들 또는 기능을 제공하게 한다.

[0222] Tereon 시스템 아키텍처는 유용한데, 확장 가능한 리던던트 시스템들(또는 확장 가능하고, 중복되는 시스템들)을 허용하기 때문이다. 은행 코어 시스템들은 예를 들어, 카드 관리, 전자 상거래, 모바일 결제와 같이 개별 채널들 전용인 모듈들을 제공하는 경향이 있다. 이것은 사일로(silos)를 강화하고, 그것들의 IT 시스템들의 복잡성을 증가시킨다. 이러한 복잡성은 은행이 자신의 서비스들 및 시스템들을 정기적으로 업데이트하지 못하는 이유들 중에 하나이다.

[0223] Tereon은 고도의 구성 및 맞춤화가 가능한 모듈러 아키텍처로 모든 장치들 및 모든 사용 케이스들을 지원하도록 디자인된다. 이것의 핵심은 상술한 SDASF(104), 및 고도의 추상화와 같은 비즈니스 규칙 엔진(106)이다. 이것

은 Tereon의 유연성을 가능하게 하는 확장 가능한 프레임워크의 조합이다.

[0224] Tereon은 운영자(또는 오퍼레이터)가 표준 운반-등급 시스템들(standard carrier-grade systems)을 사용하게 하고, 다양한 트랜잭션 타입들을 제공 및 지원하게 한다. Tereon은 트랜잭션이 인증을 필요로 하는지 여부에 관계없이 모든 트랜잭션들을 지원할 것이다.

[0225] *스페셜 프로세스들*

[0226] 스페셜 프로세스들(208)은 데이터 서비스들의 기능을 이상적으로 레버리지한다(leverage). 그러나, 독특한 요구 사항이 코어 데이터 서비스를 변경 또는 확장하는 것을 정당화하지 않는 인스턴스들이 있을 수 있으므로, 데이터 라이브러리는 데이터로부터 직접 가져오기 위해서 스페셜 프로세스 내에 레버리지된다. 예를 들어, 이것은 AML(anti-money laundering), CRM(customer relationship management), 또는 ERP(enterprise resource planning) 기능들과 같은 그래픽-기능 프로세스들을 포함할 수 있다.

[0227] *다중 서비스들*

[0228] 각 서비스가 모듈이기 때문에, Tereon의 모듈러 구조는 여러 타입들의 서비스들 및 장치들을 지원할 수 있다. 예를 들어, 결제에서 이 구조는 Tereon이 은행들, 청구 카드들, 신용 서비스들, 신용 조합들, 직불 서비스들, 직원 제도들(employee schemes), ePurse, 로열티 제도들(loyalty schemes), 멤버쉽 제도들(membership schemes), 소액 금융(microfinance), 선불 결제(prepayment), 학생 서비스들, 발권(ticketing), SMS 알림들(SMS notifications), HLR 룩업들(HLR lookups, 또는 HLR 조회들) 등을 포함하여 복수의 결제 타입들 및 장치들을 지원하게 할 수 있다.

[0229] *다중 종단점 장치들(Multiple end-point devices)*

[0230] Tereon 모듈러 구조는 마그네틱 스트라이프 카드들, 스마트 카드들, 피처 폰들, 스마트 폰들, 태블릿들, 카드 단말기들, POS(point of sales) 단말기들, ATM들, PC들, 디스플레이 스크린들, 전자 액세스 제어들(electronic access controls), 전자 상거래 포털들, 손목 밴드들 및 기타 웨어러블들 등을 포함하여 직접적이거나 간접적으로 통신할 수 있는 거의 모든 종단점 장치(end-point device)를 지원할 수 있다.

[0231] *다중 데이터베이스들(Multiple databases)*

[0232] 모듈러 구조는 시스템이 하나의 데이터베이스에 제한되지 않는다는 다른 이점이 있다. 대신에, 여러 데이터베이스들은 문제의 데이터베이스에 특정된 모듈로 각각 연결될 수 있으므로, 특정 목적을 위해 특정 데이터베이스들을 사용하거나, 다중 이종 데이터베이스들에 걸쳐 데이터 레코드들의 조합을 사용한다.

[0233] 라이센싱 서브시스템(210)의 구현은 이것이 제공하는 허가 및 인증 혜택 뿐만 아니라 라이센싱 목적들을 위한 인증서 기관들의 사용에서 신규한 것이다. 각 모듈은 서로의 주장들을 신뢰하거나 공유 데이터베이스에서 간단한 인증을 사용하거나, 또는 각 연결 구축시 개별 라이선스 서버로 끊임없이 위임하는 대신에(수반되는 성능 및 안정성 오버헤드가 있음), 이러한 분산되고, 모듈 기반 시스템들에 대한 가장 일반적인 구현 패턴들이다. Tereon에서, 라이센싱 서브시스템은 모듈들 간의 연결들이 본질적으로 안전하고, 최소한의 성능 및 안정성 오버헤드로 액터들에 관한 신뢰되고, 검증된 메타데이터를 보유하게 한다.

[0234] 또한, 구현은 라이선스 서버 손상(licence server compromise)의 인스턴스 내 잠재적 취약성의 범위를 제한한다: 전통적인 배치에서 이런 손상은 모든 컴포넌트들의 초토화(scorched-earth) 재건에 가치가 있다. Tereon 모델에서는 새로운 중개자 서명 인증서를 요구하는(하드웨어 보안 모듈에 의해 보호되지 않는 경우) 시간-기반 노출이 있다. 사전 손상이 부여된 기존의 모든 인증서들은 아주 오래될 수 있고, 정상적인 스케줄로 갱신될 수 있다. 새로운 인증서들은 새로운 권한으로 부여될 것이고, 다른 모든 불량 인증서들은 손상으로서 거절될 것이다. 이 노출 윈도우 제어(exposure window control)는 최악의 시나리오들(worst-case scenarios)에 도움이 된다. 라이선스 서버가 보유한 데이터는 하드웨어 보안 모듈에 이상적으로 보관되는 서명 인증서 개인 키의 외부에서 완전하게 권한이 없는 정보이다.

[0235] 또한, Tereon의 디자인은 다른 Tereon 서버들과 이러한 서버들의 네트워크의 일부로써 통신할 소형 Tereon 서버를 사용하는 모바일 폰 또는 IoT 장치와 같은, 종단점 장치를 결합하는(또는 조합하는) 옵션을 발생시킨다. 그것들은 데이터를 수집하고(또는 분석하고), 활동들을 조정하기 위해서 Tereon 라이선스 서버(210), 및 아마도 하나 이사의 운영자가 운영하는 Tereon 서버들과 통신할 것이다. 그럼에도 불구하고, 종단점 장치 및 Tereon 서버 사이의 차이- 모든 차이는 장치들 및 서버들이 놓여있는 사용-케이스들에만 기반함 -는 추상적인 것일 수

있다.

[0236] 해시 체인들

[0237] 블록체인의 큰 단점 중 하나는 블록체인이 이전의 모든 트랜잭션들에 대한 감사(audit)를 저장한다는 것이다(즉, 인증 목적들로 사용되는 블록체인에서 트랜잭션 히스토리를 결정할 수 있다). 이것은 블록체인의 사이즈가 결국 너무 커져서 현실적인 시간 프레임에서 관리할 수 없기 때문에, 블록체인 접근 방식이 무한하게 확장할 수 없지만, 각 블록의 사이즈가 블록체인이 등록할 수 있는 초당 최대 트랜잭션들을 제한함을 의미한다.

[0238] 두 번째 단점은 거래 히스토리가 블록체인에 액세스할 수 있는 모든 사람이 사용할 수 있으므로, 거래 당사자들(또는 거래 파티들)이 누구인지를 확인하는 기능을 제공한다는 것이다. 이것은 프라이버시 및/또는 기밀성(confidentiality)이 가장 중요한 요구 사항들인 의미있는 모든 활동에서 블록체인을 사용하는데 있어 중요한 프라이버시 및 규제 문제들(regulatory challenges)을 나타낸다.

[0239] 다른 단점은 블록체인이 트랜잭션의 결과 또는 최종 레코드만을 해시할 수 있고, 실제 프로세스들 또는 트랜잭션 그 자체의 단계들을 유효하게 할 수 없다는 것이다.

[0240] 여기에 개시된 해시 체인은 거래하는(또는 트랜잭션을 처리하는) 당사자들(또는 파티들) 간의 레코드들을 비공개로 유지하고, Tereon의 모든 사용자들- 그것들이 공개 또는 비공개 네트워크들 상에서 동작하는지 여부에 관계없이 -을 포함하는 분산된 인증 네트워크를 제공하기 위해 특정 해싱 접근 방법을 사용하여 이러한 문제들을 극복하고자 한다.

[0241] 이것은 임의의 제3 당사자(또는 제3 파티)에게로 기본 통신의 컨텐츠들을 공개하지 않고, 공개 및 비공개 네트워크들에 걸쳐 실시간으로 동작하는 해시들의 분산된 체인의 지속적인 구성에 의해 달성된다. 이것은 분산된 해시 또는 원장의 표준 모델과 직접적으로 대조된다- 여기서, 모든 당사자(또는 모든 파티)는 모든 통신의 컨텐츠를 그것들이 해당 통신에 대한 당사자(또는 파티)인지 여부에 관계없이 보고, 수용해야 함 -.

[0242] 해시 체인이 영지식 증명(zero knowledge proof)을 포함하는 프로토콜을 사용할 때, 이것은 트랜잭션의 단계들 및 정보 또는 트랜잭션의 단계들에 의해 생성된 결과(outcome)를 각각 인증할 수 있다.

[0243] 구현은 동일한 중간 해시(intermediate hash)를 생성하는 통신에 대한 당사자들(또는 파티들), 또는 동일한 통신에 대해 고유한 중간 해시들을 생성하는 그것들이 발생할 수 있다. 또한, 구조는 해시 체인의 무결성에 영향을 주지 않고, 기존 알고리즘들이 사용되지 않으므로 당사자들(또는 파티들)이 새로운 해시 알고리즘을 마이그레이션하게 한다. 이것은 블록체인과 같은 기존 라이브 솔류션들에서 사용되는 알고리즘들을 업데이트 또는 업그레이드하는 어려움과 직접적으로 대조된다.

[0244] Tereon은 트랜잭션의 각 측면(계정)에 대한 해시 감사 체인(hash audit chain)을 생성한다. 여기서:

[0245] • Tereon은 레코드와 관련된 해시를 생성하고, 해당 레코드에 대한 해시를 저장한다. Tereon은 레코드를 생성하는 액션이 완료되면, 해당 해시를 생성할 것인데, 레코드를 생성하는 단계들과 해당 단계들로부터 발생하는 정보 또는 결과를 사용하기 때문이다.

[0246] • Tereon은 현재 레코드에 대한 데이터의 일부로 이전 레코드(previous record)에 대한 해시를 사용한다; 및

[0247] • 모든 레코드 체인에서 제1 해시(첫번째 해시)는 서버의 서명, Tereon이 해당 해시를 생성하는 날짜 및 시간, 및 필요한 경우 랜덤 번호를 가지는 랜덤 해시일 것이다.

[0248] 레코드가 둘 이상의 당사자들(또는 파티들)을 포함하는 액션이고, 각 당사자(또는 각 파티)가 이것의 액션의 사이드의 레코드를 보유해야 하는 경우, Tereon은 액션에서 당사자들(또는 파티들) 각각을 위해 다음을 수행할 것이다:

[0249] • 다른 당사자 또는 당사자들과 각 당사자의 레코드의 해시를 공유함;

[0250] • 레코드 해시를 생성할 Tereon에 대한 수신 당사자의 레코드의 일부를 형성하는 해시를 사용함;

[0251] • 다른 당사자 또는 당사자들로부터의 해시를 포함하는 레코드의 중간 해시를 생성함.

[0252] • 각 당사자가 각 액션에서 다른 당사자들의 일부를 캡슐화하는 해시를 가지도록 해당 중간 해시를 다른 당사자

또는 당사자들과 공유함;

[0253] • 액션의 레코드에 중간 해시를 포함함;

[0254] • 액션에 대해 이것이 저장할 최종 해시를 생성하고, 다음 레코드의 일부로 사용함; 및

[0255] • 전송된 해시들 각각, 또는 영지식 증명을 사용하여 프로토콜로 생성된 중간 해시를 전송자의 ID 또는 Tereon 번호와 연관시킴.

[0256] Tereon은 아래에 설명된 바와 같이, 이것에 필요한 ACID 보증들 및 실시간 세션 트랜잭션 및 처리 속도를 제공할 수 있다. 또한, 블록체인의 보급은 이 분야(또는 지역)의 개발(또는 발전)이 고려되지 않았다는 것을 의미한다.

[0257] 블록체인은 트랜잭션이 완료되면 트랜잭션의 레코드를 해시할 수 있다. 블록체인에 전달된 레코드가 실제로 트랜잭션 그 자체의 진짜(또는 실제) 레코드라는 보장은 없다. 블록체인은 이것의 기본 해시 구조가 동적 및 실시간 트랜잭션들이 아닌 정적인 데이터의 컬렉션들을 위해 디자인되고 정직하게 행동하는 이것의 운영자들의 대다수에 의존하기 때문에, 이 방식으로 제한된다. 블록체인 그 자체는 이것이 결국 일관성을 제공할 수 있다는 점에서 아직 더 한계가 있다; 트랜잭션들의 연대 순서에 따라 결정되는 ACID 일관성이 아니라, 약간 다른 트랜잭션 세트들을 포함하는 두 개 이상의 블록들이 거의 동시에 많거나 적게 발견될 때, 그것들의 트랜잭션들이 블록들로 통합되는 순서 및 블록체인에서 포트들(forks)을 관리하는 합의 모델에 의해 결정됨.

[0258] 도 5는 네 개의 계정들(502, 504, 506 및 508)을 포함하는 해시 체인의 덴드리틱 특성(dendritic nature)을 도시한다. 계정들은 동일한 서버에 있거나, 개별 서버들에 있을 수 있다. 각 시스템은 하나 이상의 서버를 지원할 수 있고, 각 서버는 하나 이상의 계정들을 지원할 수 있다. 계정들이 있는 위치는 무관하다. 또한, 도 5는 계정들의 쌍들 사이에 발생하는 5 개의 트랜잭션들을 도시한다. 계정들(502 및 504) 사이에 발생하는 두 개의 트랜잭션들, 계정들 502 및 506 사이에 발생하는 두 개의 트랜잭션들, 및 계정(506 및 508) 사이에 발생하는 하나의 트랜잭션이 있다. 도면에서, 각 박스는 열이 최상위에 있는 계정과 관련된 단계이다. 각 단계는 해당 계정 내에서의 검색, 또는 해당 계정 및 다른 보이지 않는 계정 또는 시스템과 같은 보이지 않는 액션 또는 트랜잭션을 포함한다. 이것들의 트랜잭션들 또는 액션들이 무엇인지는 무관하다. 중요한 것은 그것들이 이것의 감사에 Tereon 시스템 레코드들을 무언가를 포함한다는 것이다.

[0259] 단계 510에서, Tereon 시스템은 이 계정에 대한 이전 해시 h(502)를 획득한다. 상술한 바와 같이, 제1 해시(첫 번째 해시)는 서버의 서명, Tereon이 해시를 생성한 날짜 및 시간, 및 필요한 경우, 랜덤 번호가 있는 랜덤 해시다. Tereon은 해시를 단계 51-에서 발생하는 트랜잭션 또는 액션에 대한 레코드에 추가하고, 그런 다음 이 트랜잭션에 대한 해시 h(512)를 계산하는 시드로 이것을 사용한다. 이 단계(stage, 또는 스테이지)에서 레코드는 h(502) 및 h(512)를 포함한다.

[0260] 단계 512에서, 시스템은 계정(504)을 보유하는 서버와 해시 h(510)를 교환한다. 이것은 계정(504)에 대한 이 트랜잭션에 대한 해시 h(504)를 레코드에 추가하고, 중간 해시 h(512i)를 생성하고, 이것의 레코드에 이것을 추가하고, 그런 다음 계정(504)로부터 중간 해시 h(514i)(후술되는 바와 같이, 단계 514에서 생성됨)로 이것을 교환한다. 그런 다음, 이것은 이것의 레코드에 이 해시를 추가하고 해시 h(512)를 생성한다.

[0261] 이 해시 h(512)는 단계 512 까지의 계정(502), 및 단계 514의 중간 스테이지까지의 계정(504)에 대한 해시들의 체인을 유효하게 하는 정보를 이제 포함한다. 레코드는 h(510), h(512i), h(514i), h(504), 및 h(512)를 포함한다.

[0262] 단계 514에서, 시스템은 계정(502)를 보유하는 서버와 해시 h(504)를 교환한다. 이것은 계정(502)로부터의 해시 h(510)를 레코드에 추가하고, 중간 해시 h(514i)를 생성하고, 이것을 이것의 레코드에 추가하고, 그런 다음 이것을 계정(502)로부터의 중간 해시 h(512i)로 교환한다. 그런 다음, 이것은 이것의 레코드에 이 해시를 추가하고, 해시 h(514)를 생성한다.

[0263] 이 체인은 단계 512까지의 계정(502) 및 단계 514까지의 계정(504)에서 해시들의 체인을 유효하게 하는 정보를 이제 포함한다.

[0264] 이 절차는 상술한 바와 같이 정확하게 동일한 방식으로 트랜잭션에 대한 해시들을 생성하기 위해서 계정들 502, 504, 506 및 508 간의 추가 트랜잭션들을 위해 수행된다. 예를 들어, 단계 534에서, 시스템은 단계 528에서 생성된 계정(502)에 대한 이전 해시 h(528)을 획득하고, 감사 레코드를 발생시키는 트랜잭션 또는 액션(보이지 않

는)에 대한 레코드에 이것을 추가하고, 이 트랜잭션에 대한 해시 h(534)를 생성한다. 이 체인은 단계 534까지의 계정(502), 단계 526까지의 계정(504), 단계 530까지의 계정(506), 단계 530에서 h(530)을 생성하는데 사용되었던 계정(508)로부터의 중간 해시까지의 계정(508)에 대한 해시들의 체인을 유효하게 하는 정보를 이제 포함한다. 레코드 h(534), 및 h(528)를 포함한다. Tereon은 단계 530에서 h(524)로부터 생성된 h(530i)를 포함하는 레코드로부터 단계 528에서 해시 h(528)를 생성한다. 해시 h(524)는 단계 524에서 h(524)를 생성하는데 사용되었던 계정(508)로부터 중간 해시까지의 계정(508)를 유효하게 했던 정보를 포함한다.

[0265] *조정(Reconciliation)*

[0266] 사기꾼이 이전 트랜잭션의 레코드를 변경한 경우 트랜잭션이 수행할 수 없게 하기 위해서, 조정은 먼저 마지막 'N' 트랜잭션들에 걸쳐 수행될 수 있다. 따라서, 예를 들어, Tereon이 단계 522에 의해 제시되는 트랜잭션을 수행하기 전에, 이것은 계정(502)에 대한 이전 'N' 트랜잭션들까지 단계 516, 및 아마도 단계 512 등에 대한 해시들을 먼저 재계산할 수 있다. 감사 추적(audit trail)은 트랜잭션들을 위해 최종 해시 값들을 다시 계산하기 위한 충분한 정보를 가질 것이다. 마찬가지로, 계정(504)을 보유한 시스템은 단계 526, 단계 520 등에 대한 해시들을 다시 계산할 수 있다. Tereon은 단계 522 트랜잭션에 대한 계정(506)에 대한 모든 해시들을 다시 계산할 필요가 없다.

[0267] 해시 체인에서, 레코드된 해시들 중에서 어느 하나가 다시 계산된 해시들과 일치하지 않는 경우, 이것은 레코드가 인증없이 변경되었음을 의미하고, 운영자는 즉시 문제를 조사하거나 추가 트랜잭션들을 차단할 수 있다.

[0268] *시스템 해시 체인*

[0269] 또한, 시스템 해시는 각 레코드에 추가될 수 있다. 이것은 액션이 해싱된 레코드가 속한 계정과 관련이 있는지 여부에 관계없이, 레코드의 해시일 것이고, 여기서 시드는 시스템 상에 이전 액션의 해시일 것이다. 그런 다음, 시스템 해시가 각 계정 내의 해시 체인에 추가되는 경우, 시스템 전체의 해시 체인은 제공된다.

[0270] 도 6은 동일한 시스템 상의 두 개의 계정들(602 및 604)를 포함하는 해시 체인의 덴드리틱 특성을 도시하며, 모든 시스템 이벤트들을 기록하는 '시스템 계정(system account)'은 606이다. 시스템은 레코드가 상주하는 위치에 관계없이, 레코드를 발생시키는 모든 액션에 대한 레코드의 새로운 해시를 생성한다. h(606), h(608), h(612) 등 시스템 해시들이 있다.

[0271] 도 6은 동일한 시스템 상의 두 개의 계정들(602 및 604)를 포함하는 해시 체인의 덴드리틱 특성을 도시하며, 모든 시스템 이벤트들을 기록하는 '시스템 계정(system account)'은 606이다. 시스템은 레코드가 상주하는 위치에 관계없이, 레코드를 발생시키는 모든 액션에 대한 레코드의 새로운 해시를 생성한다. h(606), h(608), h(612) 등 시스템 헤쉬들이 있다.

[0272] 단계 608에서, Tereon은 시스템의 감사 레코드에서 엔트리를 트리거하는 계정(602)에서 보이지 않는 액션 또는 트랜잭션의 레코드의 해시를 생성하고(계정(602)에 대한 레코드는 해당 계정에 대한 이전 레코드 해시, 해시 h(602)를 포함함), 새로운 시스템 해시 h(602)에 대한 h(606)을 사용한다. 그런 다음, 시스템은 트랜잭션에 대한 레코드에 대해 이 해시를 기록하고, 단계 610에서 계정(602)에 대한 해시 h(610)을 계산한다.

[0273] 시스템의 컴퓨팅 성능이 허용하는 경우, 이것은 계정 해시들의 동작을 반영하는 시스템 해시들에 대한 강력한 변형을 사용할 수 있다.

[0274] 단계 610에서, Tereon은 시스템 계정(606)과 해시 h(602)를 해시 h(606)로 교환한다. 이것은 시스템 계정(606)으로부터의 해시 h(606)를 그것의 레코드에 추가하고, 중간 해시 h(610i)를 생성한다. 이것은 시스템의 감사 레코드에서 엔트리를 트리거하고 이것의 레코드에 해시를 추가하는 계정(602)에서 보이지 않는 액션 또는 트랜잭션을 완료한 후에 이것을 생성한다. 그런 다음, Tereon은 중간 시스템 해시 h(608i)로 이 중간 해시를 교환한다. 그런 다음, 이것은 이것과 h(608)을 이것의 레코드에 추가하고 새로운 계정 해시 h(610)을 생성한다.

[0275] 단계 612에서, Tereon은 계정들(602 및 604)과 단계 608에서 생성된 해시 h(608)를 교환한다. 이것은 단계 610에서 생성된 h(610), 및 h(604)를 이것의 레코드에 추가하고 중간 해시 h(612i)를 생성한다. 이것은 계정들(602 및 604)과 이것을 그것들의 중간 계정 시스템 해시들 h(614si) 및 h(616si), 및 계정 (602)에 대응하는 중간 해시들 h(614i) 및 계정(604)에 대응하는 h(616i)로 교환한다. 그런 다음 이것은 새로운 시스템 해시 h(612)를 생성한다. 그런 다음, 시스템은 이 해시를 기록한다.

[0276] 단계 614에서, Tereon은 시스템 계정(606)과 단계 610에서 생성된 해시 h(610)을 교환한다. 이것은 시스템 계

정(606)으로부터의 해시 h(608)- 단계 608에서 생성됨 -를 이것의 레코드에 추가하고, 중간 계정 시스템 해시 h(614si)를 생성한다. 이것은 계정(604)를 사용하여 트랜잭션을 완료한 후에 이 해시를 생성하고(중간 트랜잭션 해시들 h(614i) 및 h(616i)를 교환함), 이것을 이것의 레코드에 추가하고, 그런 다음 이것을 중간 시스템 해시 h(612i)로 교환한다. 그런 다음 이것은 이것 및 h(618)을 이것의 레코드에 추가하고 계정 해시 h(614)를 생성한다.

[0277] 단계 616에서, Tereon은 시스템 계정(606)과 해시 h(604)를 교환한다. 이것은 시스템 계정으로부터의 해시 h(608)을 이것의 레코드에 추가하고, 중간 계정 시스템 해시 h(616si)를 생성한다. 이것은 계정(602)을 사용하여 트랜잭션을 완료한 후에 이것을 생성하고(그리고 중간 트랜잭션 해시들 h(614i) 및 h(616i)를 교환함), 해시를 이것의 레코드에 추가하고, 그런 다음 이것을 중간 시스템 해시 h(612i)로 교환한다. 그런 다음 이것은 이것 및 h(608)을 이것의 레코드에 추가하고 계정 해시 h(616)를 생성한다.

[0278] 단계 612에서, 하나의 옵션은 시스템이 계정(604)로 중간 시스템 해시 h(614si), 및 계정(602)로 중간 시스템 해시 h(616si)를 전송하는 것이다. 이것은 이것들의 계정들에 대한 최종 레코드 해시들을 의미하며, h(614) 및 h(616)은 3 개의 중간 시스템 해시들 h(614si), h(614si), 및 h(612i)의 레코드들을 포함하므로 확실성의 추가 레이어를 제공한다.

[0279] 시스템 해시 체인은 각 개별 트랜잭션의 양측의 해시들 뿐만 아니라 트랜잭션들이 전체적으로 이제 포함하므로, 해시 체인을 상당하게 강화한다.

[0280] Tereon이 다른 시스템 상에 계정들 간의 트랜잭션을 관리하는 경우, 프로세스는 이 시스템들 각각에 대한 단계들 608 및 610과 같다.

[0281] *라이선스 서버 해시들(Licence server hashes)*

[0282] 해시들은 개별 Tereon 시스템들 사이에 생성된 이것들과 관련이 있다. 이러한 시스템들이 서로 상호 작용할 때, 그것들은 모든 그것들의 시스템들 상에서 트랜잭션들을 유효하게 하는 정보를 포함하는 해시 트리에 결국 참여할 것이다. 그러나, 이것은 이러한 시스템들이 서로 상호 작용하는 레이트로만 증가한다. 시스템은 더욱 발전할 수 있고, 각 서버가 즉시 글로벌 해시 트리에 참여할 수 있는 다른 레이어를 구축할 수 있다. 이것은 블록체인으로부터 완전하게 해시 체인을 구별할 수 있다.

[0283] 블록체인 운영자가 프라이빗 블록체인을 설정하면, 해당 블록체인은 다른 것들 모두와 격리되어 작동한다. 전반적인 처리 속도가 향상하는 것은 다른 방법으로 제공할 수 있는 임의의 보안에서 상실되는데, 사용자가 트랜잭션의 유효성을 검사하기 위해 대규모 블록체인들의 네트워크에 의존할 수 없기 때문이다. 보안에 대한 블록체인의 주장들 중에서 하나는 공격자가 이것의 보안을 손상시키기 위해서 많은 블록체인 네트워크의 노드들을 손상시켜야 한다는 것이다(노드들의 25-33% 정도의 손상은 블록체인을 손상시키기에 충분할 수 있음). 단일 프라이빗 블록체인은 정의에 따라 해당 번호를 하나로 줄인다.

[0284] 해시 체인을 사용하면, 프라이빗 Tereon 서버 또는 네트워크는 심지어 퍼블릭 Tereon 서버들 및 네트워크들에 의해 생성된 해시 체인들로부터의 이점을 얻을 수 있다. 프라이빗 Tereon 서버 또는 네트워크를 운영하는 것은 운영자가 Tereon 시스템의 인증 강도에 대해 타협해야 한다는 것을 의미하지 않는데, 해당 시스템이 여전히 글로벌 해시 체인의 멤버일 것이기 때문이다. 이것은 단순하게 이것의 트랜잰션들- 라이선스 서버와 관련된 그것들 이외에 -이 해당 시스템에 대해 완전하게 프라이빗으로 유지될 것이라는 것이다.

[0285] 이것을 달성하기 위해서, 모든 서버는 다른 Tereon 서버들과 상호 작용하는지 여부에 관계없이, 라이선스 서버와 상호 작용해야 한다. Tereon 서버가 폐-루프 서버에서 동작하고, 해당 루프가 두 개 이상의 서버로 구성되는 경우에만 이것은 해당 루프 내에서 다른 Tereon 서버들과만 상호 작용할 것이다.

[0286] 라이선스 서버 해시를 추가하면, 모든 서버는 이것이 라이선스 서버와 상호 작용하자마자- 이것은 기본적으로 매일 수행해야함 - 글로벌 서버 해시 체인에 참여할 것이다. 라이선스 서버 해시들은 Tereon 서버 및 라이선스 서버 간의 2 개의 파티 트랜잭션(two-party transaction)에 의해 기본적으로 생성된다. 라이선스 서버 트랜잭션들은 Tereon 서버들 간의 임의의 기본 데이터 트랜잭션들에 영향을 미치지 않는다는 사실 이외에도, 각 서버를 위한 시스템 해시들이 라이선스 서버 해시들로부터 파생된 정보를 이제 포함할 것이며, 그반대의 경우도 마찬가지이다.

[0287] 도 7은 라이선스 해시들의 덴드리틱 특성을 도시한다. 이 간단한 예에서, 시스템 서버(702)는 시스템 서버들(704 및 706)이 상호 접속하는 폐 루프 시스템이다. 3 개 모두는 라이선스 서버(708)와 주기적으로 상호 작용

해야 한다.

[0288] 라이선스 서버(708)와 이것의 최초 질문에서 각 서버는 이것의 공개 키, 서버가 처음 라이선스가 부여된 날짜 및 시간, 및 데이터의 랜덤 세트로부터 이것의 제1 해시(첫 번째 해시)를 생성한다.

[0289] 단계 710에서, Tereon은 중간 라이선스 해시 h(710i)를 생성하기 위해 이것의 해시 h(708)을 사용하고, 이것을 이것의 레코드에 추가하고, 이것을 서버(702)로부터의 중간 시스템 해시 h(712i)로 교환한다. 그런 다음, 이것은 이 해시를 이것의 레코드에 추가하고, 그런 다음 이것의 레코드에 추가하는 라이선스 해시 h(710)을 생성한다.

[0290] 단계 712에서, Tereon은 중간 시스템 해시 h(712i)를 생성하기 위해 이것의 해시 h(702)를 사용하고, 이것을 이것의 레코드에 추가하고, 이것을 라이선스 서버(708)로부터의 중간 라이선스 해시 h(710i)로 교환한다. 다음, 이것은 이 해시를 이것의 레코드에 추가하고, 이것의 레코드에 추가하는 시스템 해시 h(712)를 생성한다.

[0291] 단계 712에서, Tereon은 중간 시스템 해시 h(712i)를 생성하기 위해 이것의 해시 h(702)를 사용하고, 이것을 이것의 레코드에 추가하고, 이것을 라이선스 서버(708)로부터의 중간 라이선스 해시 h(710i)로 교환한다. 다음, 이것은 이 해시를 이것의 레코드에 추가하고, 이것의 레코드에 추가하는 시스템 해시 h(712)를 생성한다.

[0292] 단계 716에서, Tereon은 중간 시스템 해시 h(716i)를 생성하기 위해 이것의 해시 h(704)를 사용하고, 이것을 라이선스 서버(708)로부터의 중간 라이선스 해시 h(714i)로 교환한다. 그런 다음, 이것은 이 해시를 이것의 레코드에 추가하고 이것의 레코드에 추가하는 시스템 해시 h(716)을 생성한다.

[0293] 단계 718에서, Tereon은 중간 라이선스 해시 h(718i)를 생성하고, 이것을 이것의 레코드에 추가하고, 이것을 서버(706)로부터의 중간 시스템 해시 h(720i)로 교환한다. 그런 다음, 이것은 이 해시를 이것의 레코드에 추가하고 이것의 레코드에 추가하는 라이선스 해시 h(718)을 생성한다.

[0294] 단계 720에서, Tereon은 중간 시스템 해시 h(720i)를 생성하기 위해 이것의 해시 h(706)을 사용하고, 이것을 이것의 레코드에 추가하고, 이것을 라이선스 서버(708)로부터의 중간 라이선스 해시 h(718i)로 교환한다. 그런 다음, 이것은 이 해시를 이것의 레코드에 추가하고, 이것의 레코드에 추가하는 시스템 해시 h(720)를 생성한다.

[0295] Tereon 서버 트랜잭션들에 대한 이러한 세 개의 라이선스 서버는 다음의 결과들을 발생시키게 된다:

[0296] • 단계 712에서 생성된 해시 h(712)는 다음의 상태를 유효하게 하는 정보를 포함한다:

[0297] • 중간 해시 h(710i)까지 라이선스 서버(708)의 해시 체인; 및

[0298] • 해시 h(712)까지 서버(702)의 해시 체인.

[0299] • 단계 716에서 생성된 해시 h(716)은 다음의 상태를 유효하게 하는 정보를 포함한다:

[0300] • 중간 해시 h(714i)까지 라이선스 서버(708)의 해시 체인;

[0301] • 중간 해시 h(702ii)까지 서버(702)의 해시 체인; 및

[0302] • 해시 h(716)까지 서버(704)의 해시 체인.

[0303] • 단계 720에서 생성된 해시 h(720)은 다음의 상태를 유효하게 하는 정보를 포함한다:

[0304] • 중간 해시 h(718i)까지 라이선스 서버(708)의 해시 체인;

[0305] • 중간 해시 h(k702ii)까지 서버(702)의 해시 체인;

[0306] • 중간 해시 h(716i)까지 서버(704)의 해시 체인; 및

[0307] • 해시 h(720)까지 서버(70)의 해시 체인.

[0308] • 단계 718에서 생성된 해시 h(718)는 다음의 상태를 유효하게 하는 정보를 포함한다:

[0309] • 해시 h(718)까지 라이선스 서버(708)의 해시 체인;

[0310] • 중간 해시 h(k702ii)까지 서버(702)의 해시 체인;

[0311] • 해시 h(k704i)까지 서버(704)의 해시 체인; 및

[0312] • 해시 h(720)까지 서버(706)의 해시 체인.

[0313] 따라서, 라이선스 및 시스템 해시들은 그것들이 네트워크에서 모든 서버상에 트랜잭션들을 검증하게 하는 정보를 그것들의 서버들이 상호 연결되었는지 또는 폐 루프로 동작하는지 여부에 관계없이 포함한다.

[0314] 따라서, 라이선스 및 시스템 해시들은 그것들이 네트워크에서 모든 서버상에 트랜잭션들을 확인하게 하는 정보를 그것들의 서버들이 상호 연결되었는지 또는 폐 루프로 동작하는지 여부에 관계없이 포함한다.

[0315] *오프-라인 트랜잭션들(Off-line transactions)*

[0316] 이 접근 방식을 사용하여, 오프-라인 트랜잭션들은 장치들 및 그것들의 서버들 간의 지속적인 통신 링크들이 제거되어야 하므로, 온-라인 트랜잭션들과 동일한 유효성을 이제 가질 수 있다. 따라서, 센서들, 휴대 가능한 결제 단말기들, 및 기타 등등과 같은 장치들은 서로 통신할 수 있고, 데이터를 다운로드 및 업로드하기 위해서 일정한 간격들로 그것들의 서버들과 연결할 수 있다. 시스템은 연결되고, 연결되지 않은 환경들 간에 원활하게 동작할 수 있다.

[0317] 해시 체인은 장치들이 그것들 각각의 서버들과 통신할 수 없는 동안에도 그것들이 오프-라인 트랜잭션들에 관여할 수 있는지 여부를 결정하기 위해 비즈니스 규칙들을 사용하여, 그것들 간의 트랜잭션들을 유효하게 하고 감사하게 할 수 있다. 장치들은 그것들이 다시 그것들의 서버들과 연결될 때 그것들의 서버들과 그것들의 감사 및 트랜잭션 레코드들을 단순히 조정한다.

[0318] 도 8은 네 개의 장치들 각각의 Tereon 서버들로부터 일시적으로 오프-라인되는 네 개의 장치들을 포함하는 해시 체인의 예를 도시한다. 이것들 중에서 세 개(802, 804 및 806)은 시각화된다(네번째 장치(808)는 단계 828에서 체인과 상호 작용함).

[0319] 장치들 간의 오프-라인 트랜잭션들을 지원하기 위해서, 장치들 그 자신들은 그것들이 참여하는 각 트랜잭션의 해시를 생성할 것이다. 장치가 다시 온-라인이고 이것의 서버와 통신할 때, 해당 장치는 이것의 서버로 해당 트랜잭션에 대한 해시를 전송할 것이다.

[0320] 트랜잭션을 개시하는 장치가 오프-라인이면, 이것은 이것의 트랜잭션에 대한 해시를 생성하고, 해당 해시를 저장할 것이다. 또한, 이것은 이것의 상대방 장치(또는 카운터파티 장치)(이것이 트랜잭션 중인 장치)로 해당 해시를 전송할 것이고, 해당 상대방 장치는 제1 장치로 이것의 해시를 전송할 것이다. 이것은 상술한 해시 체인들과 동일한 방식으로 달성된다. 장치들은 블루투스, NFC, 로컬 Wi-Fi 등과 같은 임의의 양-방향 채널을 통해 서로 통신할 수 있다. 그것들은 심지어 다른 것들이 판독하도록 각 트랜잭션 스테이지(transaction stage)에 대한 바코드들을 그것들의 스크린들 상에 게시할 수 있다. 또한, 각 장치는 이것의 트랜잭션 레코드의 서면되고, 암호화된 카피를 다른 장치로 전송할 수 있는데, 서명은 해당 레코드를 위한 목적지 서버(destination server)도 포함할 것이다. 목적지 서버만 해당 레코드를 해독할 수 있다.

[0321] 장치가 이것의 Tereon 서버와 통신을 회복하면, 해당 장치는 이것의 오프-라인 트랜잭션들 및 그것들과 관련된 해시들의 암호화된 레코드들을 서버로 전송할 것이다. 또한, 이것은 이것의 상대방들로부터의 레코드들과 같은 이것이 보유하는 다른 트랜잭션들의 카피들을 해당 서버로 전송할 것이고, 그런 다음 서버는 그것들의 레코드들 및 그것들과 관련된 해시들을 그것의 상대방 장치들이 등록된 서버로 전송할 것이다. 각 장치는 트랜잭션에서 이것의 부분을 식별할 이것 자신의 고유한 내부 트랜잭션 번호(예를들어, 모노토닉 카운터에 의해 생성된 것과 같은)를 생성할 것이다. 또한, 트랜잭션이 온-라인인 경우, 장치에 연결된 서버는 장치 및 서버 모두가 사용할 고유한 트랜잭션 번호를 생성할 것이다.

[0322] 장치들은 그것들의 각 트랜잭션의 인과 관계를 유지하기 위해서 내부 트랜잭션 번호들과 시간 및 날짜 스탬프들, 장치들 클럭 스큐(devices clock skew)에 관한 정보 및 다른 정보를 결합할 수 있다. 그것들 각각의 서버들이 트랜잭션 정보를 수신하면, 그것들은 트랜잭션들의 순서를 재구성할 수 있으므로, 모든 장치들에 대한 온-라인 및 오프-라인 트랜잭션들 모두의 인과 관계를 유지한다.

[0323] 도 8을 참조하면, 단계 812에서, 장치(802)는 h(812)를 생성하기 위해서 서버(810)로부터의 해시 h(802), 이전 레코드 해시, 및 해시 h(810)를 포함하는 트랜잭션의 레코드를 해싱한다. 그런 다음, 이것은 이 해시를 서버(810)에 전달하는데, 여기서 해당 해시는 단계 814에서 h(814)를 계산하는데 사용되는 레코드의 일부를 형성한다. 장치(802)는 이것의 Tereon 서버(810)에 연결된다는 것을 의미하는 이 때 온-라인이다. 단계 814에서, Tereon은 서버(810)에 대한 이전 해시 h(810)을 사용하고, 이것 및 h(812)를 레코드에 추가하고, 그런 다음 h(814)를 계산한다. 레코드는 h(810), h(812), 및 h(814)를 포함한다.

[0324] 운영자가 시스템 해시를 포함하기 위해 Tereon을 구성한 경우, 상술한 바와 같이, 해시 h(814)를 계산하기 전에 이것을 레코드에 추가한다. 그런 다음, 레코드는 h(812), h(810), 관련된 경우 중간 시스템 해시, 및 h(814)를 포함할 수 있다.

[0325] 단계 816에서, 장치(802)는 지금 오프-라인인데, 이것은 서버(810)에 연결될 수 없기 때문이다. 이것은 장치(804)와 트랜잭션하는데, 장치(804)는 또한 이것의 각 Tereon 서버로부터 오프-라인이다. 장치들(802 및 804)는 단계 818에서 장치(802)로부터의 중간 해시 h(816), 장치(804)로부터의 중간 해시 h(818), 장치(802)로부터의 해시 h(816), 및 장치(804)로부터의 해시 h(818)을 생성하기 위해서 상술한 해싱 절차를 따른다. 장치들(802 및 804)는 그것들의 오프-라인 공개 키들로 그것들의 해시들에 이제 서명하고, 다른 장치로 이것을 해당 트랜잭션에 대한 레코드의 안호화된 카피와 함께 전달한다. 이것은 장치(802)의 제1 오프-라인 트랜잭션(또는 첫번째 오프-라인 트랜잭션)이고- 이것은 서버(810)과 접촉이 없어 졌기 때문 -, 장치(804)의 제1 오프-라인 트랜잭션(또는 첫번째 오프-라인 트랜잭션)이다- 이것의 서버와 접촉이 없어 졌기 때문 -. 관리자는 어플리케이션이 이것의 마지막 n 트랜잭션들까지를 오프-라인으로 트랜잭션하는 각 고유한 장치로 전송할 수 있도록 시스템을 설정할 수 있다.

[0326] 이 절차는 장치(802) 및 장치(804) 사이, 및 장치(804) 및 장치(806) 사이 체인에서의 추가 트랜잭션들을 위해 반복된다. 이러한 트랜잭션들에서, 장치들(802 및 804)는 이것들의 해시 및 레코드를 이전 트랜잭션들로 교환할 필요가 없는데, 그것들 각각이 이미 카피를 보유하고 있기 때문이다.

[0327] 장치(802)는 단계 830에서 이것의 서버(810)과 접촉을 재설정할 때까지 이 방식으로 계속 동작할 것이다. 장치(802)는 이것의 오프-라인 트랜잭션들 및 그것들과 관련된 해시들- 이 예에서, 단계 816, 822, 및 826에서 각각 생성된 h(816), h(822), 및 h(826) -의 암호화된 레코드들 모두를 이제 업로드한다. 또한, 이것은 장치(804, 806, 및 808)에 대해 이것이 보유하는 암호화된 트랜잭션 레코드들 및 해시들을 업로드한다. 서버는 이들을 저장하고, 장치(804, 806 및808)에 대응하는 서버들로 그것들을 각각 업로드한다. 서버(810)는 이 업로드를 트랜잭션으로 등록하고, 단계 832에서 해시 h(832)를 생성한다. 장치(802)는 장치들(804, 806, 및 808), 및 각각의 트랜잭션 레코드들로부터의 이것의 해시들의 레코드를 클리어하고, 단계 830에서 해시 h(830)을 생성한다.

[0328] 장치(802)는 단계 820에서 해시 h(820) 및 h(808)을 발생시키는 장치들(806 및 808) 간의 트랜잭션에 대한 해시 및 암호화된 레코드를 보유한다. 이 예에서, h(808)은 여기에서 해당 트랜잭션을 위해 생성된 장치(808)에 대한 해시를 나타내는데 사용되는데, 이것은 얼마나 많은 오프-라인 트랜잭션들이 발생했는지 알 수 없기 때문이다.

[0329] 서버(810)은 장치(802)로부터 수신한 오프-라인 레코드들을 장치들(804, 806, 및 808)로부터 수신한 그것들, 및 그것들의 트랜잭션들을 포함하는 임의의 다른 서버와 조정할 것이다. 서버(810)는 어떤 서버들로부터의 레코드들을 수신할 것인지를 알 것인데, 이들은 장치(802)와 관련된 트랜잭션들에 대한 레코드들이 전송된 서버들에 해당할 것이기 때문이다. 장치(802)는 장치(808)로부터의 레코드들을 수신하는 것을 기대하지 않을 것인데, 장치(802)가 장치(808)와 트랜잭션하지 않았기 때문이다. 장치들(804 또는 806)이 다른 서버들에 소속된 오프-라인 장치들과 트랜잭션했던 경우, 서버(810)는 이것들의 다른 서버들로부터 추가적인 레코드들을 수신할 것이다.

[0330] 서버(810)은 트랜잭션 레코드들 및 서명들에 대한 시간 및 날짜 스탬프를 해당 트랜잭션들을 주문 및 번호를 매기기 위해 사용하고, 그것들을 오프-라인 트랜잭션들로 표시할 것이다.

[0331] 오프-라인 모드는 여러 가지 변형들을 제시한다. 첫 번째는 중간 오프-라인 해시들 없이 수행하고, 각 장치의 이전 트랜잭션에 대한 해시들을 간단히 사용하는 것이다. 이것은 계층의 확실성을 상실하지만, 잘 작동한다. 두 번째는 오프-라임 트랜잭션들을 위해서만 장치 해시들을 생성하는 것이다. 이것은 온-라인 트랜잭션들을 약간 단순화하지만, 계층의 확실성을 다시 상실한다. 세 번째 변형은 특정 오프-라인 공개 키로 오프-라인 트랜잭션들에 대한 레코드들에 서명하는 것이 아니라, 장치의 키로 모든 레코드에 간단하게 서명하는 것이다. 서버 및 장치 모두는 어떤 트랜잭션들이 온-라인 및 오프-라인인지 알 수 있는데, 이것들이 계정 감사 추적에 기록될

것이기 때문이다. 그러나, 장치를 위한 개별 키 및 일련의 트랜잭션 번호들을 실행하면, 이것은 오프-라인 대 온-라인 트랜잭션들을 보여주기가 쉽다.

[0332] 네 번째 변형은 각 서버가- 이것이 이것의 연결된 장치들로부터의 오프-라인 트랜잭션들의 레코드들을 수신할 때 - 그것들의 레코드들이 적용되는 모든 서버들이 그것들의 서버들로부터의 레코드들을 예상하도록 통지하는 것이다. 예를 들어, 도 8에서 도시된 오프-라인 다이어그램에서, 이것은 장치(804)가 이것의 서버에 나중에 연결되고, 장치(806)가 다른 장치(미도시)와 트랜잭션된다고 가정한다. 장치(804)가 이것의 서버와 연결하면, 해당 서버는 장치(802)에 관한 레코드들을 서버(810)로 전송할 것이다. 장치(804)는 임의의 다른 장치와 오프-라인으로 트랜잭션하지 않았고 임의의 다른 장치를 위한 오프-라인 레코드들을 보유하지 않는다. 반면에, 서버(810)는 장치(804)에 대한 이것의 레코드들을 장치(804)에 대응하는 서버로 전송하고, 해당 서버에 장치(806)로부터 동일한 레코드들의 카피들을 수신함을 예상할 수 있도록 통지한다- 단계 826 및 828에서의 트랜잭션 동안 장치(802)는 이것들을 장치(806)로 전달했음 -. 마찬가지로, 장치(806)가 이것의 서버에 연결(또는 접속)하면, 해당 서버는 장치(802)에 대한 이것의 레코드들을 서버(810)로, 장치(804)에 대한 것을 장치(804)에 대응하는 서버로, 장치(808)에 대한 것을 장치(808)에 대응하는 서버로, 다른 장치에 대한 것을 이것의 각 서버로 전송할 것이다. 또한, 이것은 장치(802(서버810) 및 804)에 대응하는 서버들 모두에게 다른 장치에 대응하는 서버로부터의 레코드들을 예상하도록 통지할 것이다.

[0333] 해시 체인을 사용하는 것은 Tereon 상의 부담의 지속적 증가를 부과하지 않는다. 액션은 두 개 이상의 당사자들을 포함하는 경우가 거의 없고, 그 액션이 있는 곳에서는, 해당 액션은 일반적으로 일대다(one-to-many) 전송일 것이고, 그 자체가 그저 일대일(one-to-one) 전송들의 컬렉션이다. 또한, 다대일(many-to-one) 전송은 일반적으로 일련의 일대일 전송들일 수 있으며, 이는 그저 두 개의 당사자 액션들의 컬렉션이다.

[0334] *레코드 수정(Amending record)*

[0335] 사용자가 레코드를 수정하는 경우, Tereon은 원본 레코드(original recor)를 덮어 쓰지 않을 것이다. 대신에, Tereon은 수정된 레코드를 포함하는 새로운 레코드를 그저 생성하고, 이것은 Tereon이 해당 레코드가 다시 수정될 때까지 나타내는 버전일 것이다. 수정(amendment)은 액션이다. 이것은 이전 트랜잭션의 결과를 효과적으로 수정하는 모든 금융 및 트랜잭션 레코드들- 결제와 같은 트랜잭션들의 결과 -에서 발생할 것이다. 또한, 이것은 운영자가 이메일들, 의료 레코드들 등과 같은 다른 레코드 타입들을 관리하기 위해 Tereon의 서브 세트를 사용하는 경우에도 발생할 것이다. 이렇게 하면, Tereon은 레코드의 각 버전의 카피를 보유할 것이다.

[0336] 법원(court of law), 또는 법의 일반적의 운영이 운영자가 레코드를 완전하게 지우거나, 원본 레코드를 수정하도록 요구하는 상황들이 있을 수 있다. 이러한 상황들에서, Tereon은 원본 레코드의 컨텐츠들, 및 아마도 관련된 레코드들의 컨텐츠를 삭제 또는 수정할 것이다. Tereon은 후속 해시들을 무효화하지 않고도 이것을 달성할 수 있다.

[0337] Tereon이 히스토리컬 레코드를 삭제 또는 수정하는 경우, 이것은 다음과 같을 것이다:

[0338] • Tereon이 레코드를 삭제 또는 수정하기 전에 해당 레코드를 수정 또는 변경되는지 확인하기 위해서 해당 레코드의 해시를 재생성하고, 재생성된 해시를 기록함.

[0339] • 삭제 또는 수정된 레코드의 컨텐츠들, 및 삭제 또는 수정에 대한 이유들을 원본 레코드에서의 새로운 필드에 기록함.

[0340] • 기록에서 관련 필드들 삭제 또는 수정, 및 해당 삭제 또는 수정 날짜 및 시간 추가함.

[0341] • 해당 레코드에 해단 새로운 해시 생성함; 및

[0342] • 새로운 해시 기록함.

[0343] 이 절차를 따르면, Tereon은 어떤 방식으로든 해시 체인을 수정할 필요가 없다. 삭제 또는 수정된 레코드의 원본 해시로부터 생성된 유효한 레코드들을 위한 모든 해시들은 유효할 것이다. 시스템 해시는 삭제 또는 수정이 액션이므로 삭제 또는 수정된 레코드의 새로운 해시를 포함할 것이다. 이러한 방식으로 사기 행위는 재계산된 해시들과 일치하지 않는 모든 기록된 해시들을 찾아 내어 쉽게 인식될 것이다.

[0344] *영지식 증명들을 갖는 해시 체인(Hash chain with zero knowledge proofs)*

[0345] 해시 체인은 트랜잭션의 양측이 해시들과 관련된 레코드들을 해시했다는 것을 다른 것에게 증명하게 하는 추가된 레이어를 제공한다. 이것은 레코드의 해시가 해당 레코드의 실제 해시라는 것을 하나의 파티가 제2 파티(검증자(verifier))에게 증명하게 하는 해시 체인 내에 키 교환 알고리즘을 포함하여 이루어 진다.

[0346] 두 당사자들이 공용 키를 협상하게 하는 임의의 알고리즘은 여기서 사용될 수 있고, 영지식 증명을 사용할 필요가 없다. 그러나, 영지식 증명들을 사용하는 PAKE(password authenticated key exchange) 알고리즘들은 여기에서 사용하는 것이 가장 효율적이다. 중간 스테이지에서 올바른 PAKE 프로토콜 및 영지식 증명을 사용하는 것은 당사자들이 동일한 중간 해시들을 생성할 것이므로, 해시들을 교환할 필요가 없다.

[0347] 양측이 영지식 증명을 사용하여 동일한 해시를 생성하게 하는 PAKE 알고리즘과 같은 알고리즘을 사용하면, 당사자들은 더 멀리 갈 수 있다. '증명'을 생성하기 위해서 트랜잭션을 구성하는 정보를 포함 및 사용할 수 있는 영지식 증명을 사용하여, 당사자들은 모두 동일한 중간 해시를 생성할 수 있다. 이것은 그것들이 중간 해시를 서로 교환할 필요가 없다. 또한, 이것은 레코드 및 정보를 생성하는 단계들과 그것들의 단계들로부터 발생하는 결과가 해시 체인 프로세스의 컴포넌트들이 된다는 것을 의미한다. 둘 이상의 당사자들이 참여하는 경우, Tereon은 모든 당사자들이 공용 해시를 생성하게 하도록 프로토콜 및 영지식 증명의 그룹 변형을 사용할 수 있다.

[0348] 당사자들이 동일한 해시를 생성하게 하는 PAKE 알고리즘들은 그것들이 중간 해시를 생성할 수 있기 전에 일반적으로 당사자들 간의 2 또는 3회 정보 전달을 수행한다. 트랜잭션이 완료하는데 2 개의 스테이지들만을 필요로 하는 경우(예를 들어, 요청 및 수락/검증), 당사자들은 하나의 중간 해시만을 생성할 것이다. 트랜잭션이 3 개의 스테이지들을 필요로 하고, 알고리즘이 2 개의 패스들(passes)에서 해시를 생성하는 경우, 당사자들은 3개의 스테이지를 두 번 반복하여 4 세트들의 정보를 교환하고, 2 개의 해시들- 트랜잭션에서 첫 번째 두 개의 단계들 후에 제1 해시(또는 첫 번째 해시), 그런 다음 세 번째 단계를 반복한 후에 제2 해시(또는 두 번째 해시) -을 생성할 것이다.

[0349] 이러한 영지식 증명의 예는 Schnorr NIZK 증명이다. 이 영지식 증명은 Schnorr NIZK 증명에 대한 명세서 문서에 개시된 바와 같이, 증명의 일부로 전송되는 정보 및 증명의 일부인 해시를 생성하는데 사용되는 정보 모두에 추가 정보를 추가하여 간단하게 확장될 수 있다.

[0350] 또한, SPEKE(simple password exponential key exchange) 프로토콜에서 공용 키를 생성하는 방법을 채택하는 것과 같은 다른 방법은 사용될 수 있고, 그렇게 하는 방식은 위의 사항을 고려할 때 사소한 것이다.

[0351] 또한, 이것은 당사자들이 트랜잭션 데이터에 기초하여 공용 키를 생성하게 하도록 키 교환 프로토콜들을 확장할 수 있는 간단한 방법이다. 다시 말하면, 간결함의 목적들을 위해 여기에 단순하게 도시되어 있지 않다.

[0352] 공용 키를 생성하기 위해서, 당사자들은 공용 키의 해시를 간단하게 생성한다. 해시는 트랜잭션 정보를 유효하게 할 수 있는 정보를 포함할 것인데, 해당 정보가 공용 키, 및 해시를 생성하는 프로세서에서 사용되었기 때문이다.

[0353] *두 개의 스테이지들에서 트랜잭션(Transaction in two stage)*

[0354] 이 동작 방법을 도시하는 예는 4 개의 계정들(502, 504, 506 및 508)을 포함하는 해시 체인의 덴드리틱 특성을 도시한 도 5에 다시 나타낼 것이다. 계정들은 동일한 시스템 상에 있을 수도 있고, 그것들은 개별 시스템들 상에 있을 수도 있다. 계정들이 있는 위치는 무관하다. 단계들 512 및 514에서의 이 트랜잭션은 2 개의 스테이지들을 필요로 한다.

[0355] *2 개의 패스 PAKE(Two pass PAKE)*

[0356] 단계 512에서의 제1 패스(또는 첫 번째 패스)에서 계정(502)은 단계 510에서 생성된 이 계정을 위한 이전 해시 h(510)을 취하고, 이것을 제1 스테이지(또는 첫 번째 스테이지)의 트랜잭션적인 정보에 추가하고, 제1 영지식 증명(또는 첫 번째 영지식 증명)을 구성하고, 이것을 계정(504)으로 전달한다. 영지식 증명은 제1 스테이지의 트랜잭션적인 정보 및 해시 h를 구성하는 정보를 수반한다.

[0357] 제2 패스(또는 두 번째 패스)에서, 계정(504)는 해당 계정을 위한 이전 해시 h(504)를 취하고, 이것을 제2 스테이지(또는 두 번째 스테이지)의 트랜잭션적인 정보에 추가하고, 제2 영지식 증명(또는 두 번째 영지식 증명)을 구성하고, 이것을 계정(502)으로 전달한다. 제2 영지식 증명은 제2 스테이지의 트랜잭션적인 정보 및 해시 h(504)를 구성하는 정보를 수반한다.

[0358] 계정들(502 및 504)는 두 계정들을 위한 중간 해시인 해시 h(512i514i)를 이제 독립적으로 구성한다. 두 계정들(502 및 504)는 이 해시를 그것들의 레코드들에 추가한다. 계정(502)는 단계 512에서 이것의 트랜잭선의 레코드의 해시 h(512)를 생성하고, 계정(504)는 단계 514에서 이것의 트랜잭션의 레코드의 해시 h(514)를 생성한다.

[0359] *3 개의 패스 PAKE(Three pass PAKE)*

[0360] 이 예에서, 단계들 512 및 514에서 트랜잭션은 당사자들이 3 개의 패스들 후에 공통 해시(common hash)를 구성하게 하는 PAKE 알고리즘을 사용하여 2 개의 스테이지들을 취한다.

[0361] 제1 패스 및 제2 패스는 상기와 같이 수행된다. 제3 패스(또는 세 번째 패스)에서, 계정(502)은 계정(504)가 제2 패스에서 전송했던 정보를 취하여 해당 정보로 제3 영지식 증명(또는 세 번째 영지식 증명)을 구성하고, 이것을 계정(504)으로 전송한다. 또한, 제3 영지식 증명은 제2 스테이지의 트랜잭션적인 정보 및 해시 h(504)를 구성하는 정보를 수반한다.

[0362] 계정들(502 및 504)는 해시 h(512i514i)를 이제 독립적으로 구성한다. 두 계정들(502 및 504)는 이 해시를 그것들의 레코드들에 추가한다. 2 개의 패스 PAKE 접근 방식에서와 같이 계정(502)은 단계 512에서 이것의 트랜잭션의 레코드의 해시 h(512)를 생성하고, 계정(504)은 단계 514에서 이것의 트랜잭션의 레코드의 해시 h(514)를 생성한다.

[0363] 두 가지 경우에서, 체인은 단계 512까지의 계정(502)에서, 단계 514까지의 계정(504)를 위한 해시들의 체인을 유효하게 하는 정보를 이제 구성한다. 두 계정들(502 및 504)는 중간 해시 h(512i514i) 뿐만 아니라 그것들의 레코드들을 위한 그것들의 해시들을 보유한다. 그러나, 여기서 중간 해시는 영지식 증명들을 사용하지 않은 이전 예들에서의 시스템들 사이에 교환되었던 중간 해시들과 약간 다르다. 여기서 중간 해시는 계정들(502 및 504) 간의 트랜잭션 해시이므로, 계정들(502 및 504) 모두에 공용이다. 해시는 트랜잭션의 해시고, 트랜잭션의 일부로 생성되었다. 이것은 트랜잭션에서 동시에 존재한다. 계정(504)의 해시 h(514)가 이것의 트랜잭션의 레코드의 이것의 해시인 반면, 해시 h(512)는 계정(502)의 트랜잭션의 이것의 레코드의 해시이다- 이것에 개인 정보를 포함할 것임 -. 따라서, 계정들(502 및 504)는 그것들 사이의 트랜잭션에서 실제 단계들, 및 이것들의 해당 트랜잭션의 레코드들 모두를 증명할 수 있다.

[0364] *3 개의 스테이지들에서 트랜잭션(Transaction in three stages)*

[0365] 도 5를 사용한 다른 예로서, 단계들 528 및 530에서 트랜잭션이 2 개 보다는 3 개의 개별 스테이지들을 포함한다고 가정한다.

[0366] *2 개의 패스 PAKE(Two pass PAKE)*

[0367] 제1 패스(first pass, 첫번째 패스)에서, 계정(502)는 단계 522에서 생성된 이 계정을 위한 이전 해시 h(522)를 취하고, 이것을 제1 스테이지의 트랜잭션적인 정보에 추가하고, 제1 영지식 증명을 구성하고, 이것을 계정(506)으로 전송한다. 영지식 증명은 제1 스테이지의 트랜잭션적인 정보 및 해시 h(522)를 구성하는 정보를 수반한다.

[0368] 제2 패스에서, 계정(506)은 단계 524에서 생성된 해당 계정을 위한 이전 해시 h(524)를 취하고, 이것을 제2 스테이지의 트랜잭션적인 정보에 추가하고, 제2 영지식 증명을 구성하고, 이것을 계정(502)으로 전송한다. 제2 영지식 증명은 제2 스테이지의 트랜잭션적인 정보 및 해시 h(524)를 구성하는 정보를 수반한다.

[0369] 계정들(502 및 506)은 공용 해시를 해시 h(528i530i)를 이제 독립적으로 구성할 수 있는데, PAKE 알고리즘이 2 개의 패스들 후에 당사자들이 공용 해시를 구성하게 하기 때문이다. 그러나, 트랜잭션은 여전히 수행할 제3 스테이지(또는 세 번째 스테이지)가 있다.

[0370] 이 예에서, 시스템은 PAKE 알고리즘을 사용하여 제2 패스 세트(second set of passes)를 통해 간단하게 실행한다- 트랜잭션의 제3 스테이지로 시작함- . 제2 패스 세트의 제2 패스(second pass, 두번째 패스)는 간단하게 랜덤 데이터를 사용할 수 있다. 또한, 이것은 2 개의-스테이지 트랜잭션과 함께 3 개의 패스 PAKE를 사용하는 것과 유사한 마지막 스테이지를 반복할 수 있다.

[0371] 후자의 경우에서, 제3 패스(새로운 PAKE 알고리즘의 제1 패스)는 수행되고- 계정(502)은 서명된 h(528i530i)를 취함 -, 이것을 제3 스테이지의 트랜잭션적인 정보에 추가하고, 정보로 제3 영지식 증명을 구성하고, 이것을 계정(506)으로 전송한다. 제4 패스(또는 네 번째 패스)(새로운 PAKE 알고리즘의 제2 패스)는 수행되고- 계정

(506)은 서명된 h(528i530i)를 취함 -, 이것을 계정(502)이 전송한 제3 스테이지의 트랜잭션적인 정보에 추가하고, 정보로 제4 영지식 증명(또는 네 번째 영지식 증명)을 구성하고, 이것을 계정(502)으로 전송한다. 계정들(502 및 506)은 해시 h(528i2530i2)를 이제 독립적으로 구성할 수 있다. 이것은 이 트랜잭션에서 생성된 제2 공통 해시(또는 두 번째 공용 해시)이고, 이것이 트랜잭션의 3 개의 스테이지들 모두를 포함하므로 이제 계정들(502 및 506) 간의 트랜잭션의 해시다. 계정들(502 및 506) 모두는 이 해시를 그것들의 레코드들에 추가한다. 계정(502)은 단계 528에서 이것의 트랜잭션의 레코드의 해시 h(528)을 생성하고, 계정(506)은 단계 (530)에서 이것의 트랜잭션의 레코드의 해시 h(530)을 생성한다.

[0372] 이 절차는 상술한 바와 같이 정확하게 동일한 방식으로 각 트랜잭션에 대한 해시들을 생성하기 위해서 계정들(502, 504, 506 및 508) 간의 추가 트랜잭션들을 위해 수행된다.

[0373] *3 개의 패스 PAKE(Three pass PAKE)*

[0374] 제1 패스 및 제2 패스는 상기와 같이 수행된다. 제3 패스에서, 계정(502)은 제3 스테이지의 트랜잭션적인 정보를 구성하는 정보로 제3 영 지식 증명을 구성하고, 이것을 계정(506)으로 전송한다. 영지식 증명은 제3 스테이지의 트랜잭션적인 정보를 구성하는 정보를 수반한다.

[0375] 계정들(502 및 506)은 해시 h(528i530i)를 이제 독립적으로 구성한다. 계정들(502 및 506) 모두는 이 해시를 그것들의 레코드들에 추가한다. 계정(502)는 단계 528에서 이것의 트랜잭션의 레코드의 해시 h(528)을 생성하고, 계정(506)은 단계 530에서 이것의 트랜잭션의 레코드의 해시 h(530)을 생성한다.

[0376] 도 5에 관한 예시들에서- 시스템은 중간 또는 트랜잭션 해시들을 생성하기 위해 영지식 증명들을 사용함 -, 해시 h(530)는 계정(502)의 해시들 모두를 h(528i)로, 계정(504)의 해시들 모두를 h(526i)로, 계정(508)의 해시들 모두를 계정(506)이 h(524)를 생성했을 때 생성되는 계정(508)의 중간 또는 트랜잭션 해시까지로, 및 계정(506)의 해시들 모두를 h(530)로 유효하게 하는 정보를 포함한다. 그러나, 이것이 이것의 트랜잭션 네트워크에서 모든 해시들을 유효하게 한다 할지라도, 계정(506)은 다른 계정들, 시스템들, 또는 서버들에 입력된 트랜잭션들에 대한 트랜잭션 레코드들만을 보유한다. 이것은 심지어 이것의 해시가 계정(502) 또는 계정(504)이 그것들의 트랜잭션들(또는 해당 트랜잭션들)에 대한 해시들을 검증하는데 사용할 수 있는 정보를 포함한다 할지라도, 계정들(502 및 504) 간의 트랜잭션들에 대한 트랜잭션적인 레코드들의 컨텐츠에 관해서 아무 것도 모른다.

[0377] 중요한 것은 당사자들 모두가 동일한 중간 해시를 독립적으로 생성하는데 사용하는 알고리즘이 당사자들이 트랜잭션에 영향을 주기 위해서 교환하는 단계들을 사용한다는 것이다. 따라서, 레코드를 생성하는 트랜잭션은 해시 체인 프로세스의 컴포넌트가 되고, 해시 체인 엔트리(hash chain entry)를 생성하는 프로세스는 트랜잭션에 영향을 주는 프로세스와 동일하다. 이것을 바라보는 다른 방식은 트랜잭션이 트랜잭션의 일부로 해시를 생성하고, 해당 해시 및 이것에 수반되는 정보가 트랜잭션의 감사가 되는 것이다. 그것들은 하나가 되고 동일해진다. 블록체인을 사용하면, 트랜잭션의 개시자(initiator, 또는 이니시에이터)는 트랜잭션을 완료하고, 트랜잭션에서 통합되는 대신에, 다른 단계를 프로세스에 추가하는 추후 감사를 위해 이것의 레코드를 블록체인으로 전송한다.

[0378] 트랜잭션 그 자체가 해시 체인이 제공하는 감사 추적의 동시 컴포넌트가 되므로, 감사 추적에 의해 세부 사항들이 캡처 또는 유효하게 되지 않은 트랜잭션을 갖는 다는 것은 불가능하다. 트랜잭션 완료 후 대개 완료된 트랜잭션 레코드가 감사 시스템으로 전달된다는 점에서 대부분의 감사 추적들은 '이벤트 후(after the event'이다. 이러한 경우, 감사에 의해 수신된 레코드가 트랜잭션에 의해 생성된 레코드와 동일하지 않을 가능성이 있다. 따라서, 컴퓨터 레코드들은 대개 소문자(hearsay)로 간주된다. 올바른 PAKE 또는 유사한 프로토콜을 사용하여 영지식 증명을 통합하는 것은 감사 추적이 트랜잭션에 의해 생성되고, 트랜잭션 및 그것의 레코드들이 감사 추적의 일부가 된다는 것을 의미한다. 이것은 실시간 트랜잭션들에 대해 중대한 의미를 갖는데, 그것들이 이제 감사되고, 실시간으로 보고되기 때문이다.

[0379] 영지식 증명들을 사용하여 해시를 구성하는 프로세스는 해시 체인에서의 해시들을 생성하는 모든 시나리오들에 적용될 수 있다. 이것은 도 8에 의해 도시된 시스템 해시들, 라이선스 서버 해시들, 및 심지어 오프-라인 해시들에 사용될 수 있다. 중요한 것은 해시가 2 개 또는 그 이상의 엔티티들 간의 트랜잭션을 포함한다는 것으로, 그것들의(또는 해당) 엔티티들이 당사자들, 장치들, 또는 시스템들인지 여부에 관계가 없다. 프로세스는 표준 해시들(standard hashes) 중에서 하나를 사용하는 것을 배제하지 않는다. 따라서, 하나의 시스템은 장치들이 온-라인 또는 오프-라인인지 여부에 관계없이, 계정들 간의 트랜잭션들에 대한 영지식 증명들을 사용하여 생성

된 해시들을 사용할 수 있지만, 시스템 해시들 및 라이선스 해시들을 위해 표준 해시들을 사용할 수 있다. 제2 시스템(또는 두 번째 시스템)은 모든 해시들을 위한 영지식 증명들을 사용할 수 있지만, 제3 시스템(또는 세 번째 시스템)은 오직 표준 해시들만을 사용할 수 있다.

[0380] *다중 트랜잭션 스테이지들을 포함한 다중 패스 PAKE들(Multiple pass PAKEs with multiple transaction stages)*

[0381] 상기 예들이 트랜잭션의 양측이 공용 키를 생성하게 하는 2 개 또는 3 개의 패스들이 필요한 PAKE들을 사용하는 2 개 또는 3 개의 스테이지들을 포함하는 트랜잭션들을 사용하는 방법에 관한 것이라 할지라도, 시스템은 해당 예들에 한정되지 않는다. 실제로는 다른 복수의 패스들이 필요한 PAKE들을 사용하는 복수의 스테이지들을 포함하는 트랜잭션들을 지원하는 시스템에서 동일한 방법이 효과가 있을 거라는 점(또는 작동할 것이라는 점)이다. 그러나, 시스템은 트랜잭션의 모든 스테이지들을 커버하는데 필요한 많은 PAKE들 실행을 간단하게 사용한다. 이것은 최종 공통 키를 생성하는 요구된 PAKE 패스들을 생성하기 위해 최종 스테이지를 몇 번이고 반복하므로 트랜잭션 해시를 생성한다.

[0382] *영지식 증명들을 갖는 시스템 해시 체인(System hash chain with zero knowledge proofs)*

[0383] 도 6으로 돌아가서, 영지식 증명들 및 클래식 해시들을 사용하여 생성된 해시들 모두를 사용할 수 있는 해시 체인이 도시된다. 도면은 시스템 해시들 h(606), h(608), h(612), 등과 함께, 동일한 시스템(606) 상의 두 개의 계정들(602 및604)을 도시한다. 시스템은 레코드가 상주하는 위치에 관계없이 레코드를 발생시키는 모든 액션에 대한 레코드의 새로운 해시를 생성한다. 계정들 간의 트랜잭션들은 상술한 바와 같이, 계정들 각각에 대한 중간 또는 트랜잭션 해시들을 생성하기 위해서 영지식 증명들을 사용할 수 있다. 시스템 해시는 해당 레코드를 생성할 때 각 레코드의 시스템 해시를 포함한다.

[0384] 단계들 614 및 616에서 계정들(602 및 604) 간의 트랜잭션이 3 개의 패스들 후에 당사자들이 공통 해시를 구성하게 하는 PAKE 알고리즘을 사용하여 3 개의 개별 스테이지들을 포함한다고 가정한다.

[0385] 트랜잭션의 제1 단계(또는 첫 번째 단계)에서, 계정(602)은 시스템 계정(606)과 해시 h(610)- 이것의 이전 레코드의 해시인 -를 단계 608에서 생성된 시스템 해시 h(608)로 교환한다. 이것은 이 시스템 해시 및 이것의 해시 h(610)를 단계 610에서 생성된 제1 스테이지의 트랜잭션적인 정보에 추가하고, 제1 영지식 증명을 구성하고, 이것을 계정(604)으로 전송한다. 영지식 증명은 제1 스테이지의 트랜잭션적인 정보, 해시 h(610), 및 해시 h(608)을 구성하는 정보를 수반한다.

[0386] 트랜잭션의 제2 단계(또는 두 번째 단계)에서, 계정(604)은 시스템 계정과 해시 h(604)를 단계 608에서 생성된 시스템 해시 h(608)로 교환한다. 이것은 이 시스템 해시 및 이것의 해시 h(604)- 제1 스테이지의 트랜잭션적인 정보에 대한 이것의 이전 레코드의 해시임 -를 제1 스테이지의 트랜잭션적인 정보에 추가하고, 제2 영지식 증명을 구성하고, 이것을 602로 전달한다. 영지식 증명은 제2 스테이지의 트랜잭션적인 정보, 해시 h(604), 및 해시 h(608)을 구성하는 정보를 수반한다.

[0387] 트랜잭션의 제3 단계(또는 세 번째 단계)에서, 시스템 계정(606)은 h(610) 및 h(604)를 이것의 레코드에 추가하고, 중간 시스템 해시 h(612i)를 생성한다.

[0388] 제4 단계(또는 네 번째 단계)에서, 계정(602)은 제3 스테이지의 트랜잭션적인 정보를 구성하는 정보를 사용하여 제3 영지식 증명을 구성하고, 이것을 계정(604)으로 전송한다. 제3 영지식 증명은 제3 스테이지의 트랜잭션적인 정보를 구성하는 정보를 수반한다.

[0389] 제5 단계(또는 다섯 번째 단계)에서, 계정들(602 및 604)는 해시 h(614i616i)를 독립적으로 구성한다. 계정들(602 및604) 모두는 이 해시를 그것들의 레코드들에 추가한다. 해시 h(614i616i)는 트랜잭션의 해시다.

[0390] 제6 단계(또는 여섯 번째 단계)에서, 계정(602)는 시스템 계정(606)과 h(614i616i)를 h(612i)로 교환하고, h(612i)를 이것의 레코드들에 추가하고, 단계 613에서 이것의 트랜잭션의 레코드의 해시 h(614)를 생성한다. 계정(604)는 시스템 계정(606)과 h(614i616i)를 h(612i)로 교환하고, h(612i)를 이것의 레코드들에 추가하고, 단계 616에서 이것의 트랜잭션의 레코드의 해시 h(616)을 생성하고, 시스템 계정(606)은 h(614i616i)의 2 개의 카피들을 이것의 레코드에 추가하고, 단계 612에서 새로운 시스템 해시 h(612)를 생성한다.

[0391] 단계 614에서 계정(602)의 트랜잭션의 레코드는 해시 h(610), 해시 h(604), 시스템 해시 h(608), 트랜잭션 해시 h(614i616i), 중간 시스템 해시 h(612i), 3 개의 스테이지들의 트랜잭션적인 정보, 이것의 트랜잭션의 레코드,

계정 ID, 및 해시 h(614)를 포함한다.

[0392] 단계 616에서 계정(604)의 트랜잭션의 레코드는 해시h(610), 해시 h(604), 시스템 해시 h(608), 트랜잭션 해시 h(614i616i), 중간 시스템 해시 h(612i), 3 개의 스테이지들의 트랜잭션적인 정보, 이것의 트랜잭션의 레코드, 계정 ID, 및 해시 h(616)을 포함한다.

[0393] (계정(602)의 트랜잭션의 레코드들은 다른 상태들로 트랜잭션을 시작하고 종료할 때마다, 계정(604)의 그것과 다를 것이고, 각각은 다른 계정 세부 사항들 및 ID들을 가진 다른 계정이다.)

[0394] 시스템 해시 h(612)는 각 개별 트랜잭션 뿐만 아니라, 전체 트랜잭션의 양측의 해시들을 포함하므로, 해시 체인을 상당하게 강화한다.

[0395] Tereon이 다른 시스템 상의 계정들 간의 트랜잭션을 관리하는 경우, 프로세스는 약간 다르며, 여기에서, 각 시스템은 이것이 관리하는 계정과 이것의 시스템 해시 및 중간 시스템 해시를 교환할 것이다. 그렇지 않으면, 도 6과 관련하여 상술한 방법은 계정들(602 및 604) 및 시스템(606)를 갖는 대신에, 도면이 계정(602)과 관련된 시스템(606), 및 계정(604)과 관련된 제2 시스템(또는 두 번째 시스템; 605)을 도시할 것이라는 점을 제외하고는 동일하다. 단계들 614 및 616에서 발생된 트랜잭션과 함께, 발생할 시스템 해시는 단계 612에서 시스템 트랜잭션 및 계정(604)에 대응하는 제2 시스템(또는 두 번째 시스템; 605) 상에서 대응하는 트랜잭션을 나타낼 것이다. 실제로, 동시에 거래할 수 있는 여러 계정들을 포함하는 시스템에서, 시스템은 레코드를 생성하는 각 상호 작용에 대한 해시들을 생성할 것이다.

[0396] 도 6이 순차적 해시들(sequential hashes) 및 중간 해시들을 도시하지만, 현실은 다를 것이다. 도 6a는 3 개의 계정들(602a, 604a, 및 606a) 도시하며, 모두 시스템 계정(608a)와 함께 외부 서버들 상의 계정들과 상호 작용하고 있다. 트랜잭션들의 스테이지들은 트랜잭션들이 시스템 상에서 동시에 발생할 때 발생할 수 있는 것을 설명하기 위해(또는 예시하기 위해) 인터리빙한다. 편의를 위해, 이것들은 모두 동일한 서버 상에 도시된다.

[0397] 상기 예들에서, 단계 612a에서의 계정(602a)는 h(612a)를 획득하기 위해서 시스템(608a)와 이것의 해시 h(602a)를 교환할 것이다. 시스템(608a)은 상기 예들이 중간 해시 h(616ai)로서 나타내는 것을 이제 생성할 것이다. 이 첨자(subscript) "i"는 각 트랜잭션이 3 개의 시스템 해시들, 트랜잭션 이전의 원본 해시, 트랜잭션의 특정 스테이지 동안 시스템 해시(중간 해시), 및 트랜잭션의 끝에서 시스템 해시를 포함한다는 것을 명확히 하기 위해 사용된다. 첨자 "i"는 중산 해시를 나타낸다. 상기 추론(reasoning, 또는 추리)에 따라 최종 시스템 해시는 h(616a)일 수 있다. 다중 동시에 발생하는 또는 인터리브된 트랜잭션들이 있는 경우, 라벨링(labelling)으로는 더 이상 진행 상황을 알 수 없다. 대신에, 각 시스템 해시는 트랜잭션 중에 또는 후에 이것이 생성되는지 여부에 관계없이, 이전 해시에 대한 증분(increment)이긴 하지만, 시스템 해시다. 계정(602a)가 시작하고, 그런 다음 계정(604a)가 시작하고, 계정(606a)가 시작하고, 계정(602a)가 종료하고, 계정(604a)가 종료하기 전에 계정(606a)가 종료하기 위해서 3 개의 트랜잭션들이 발생하는 경우, 다른 트랜잭션들 또는 액션들이 서버 상의 이것들(또는 계정들) 또는 임의의 다른 계정들에서 발생하지 않았었다면, 해시들의 순서는 다음과 같을 수 있고, 결과적으로 다이어그램은 이전 도면들과 미묘하게 다르다.

[0398] 계정(602a)는 h(612a)를 획득하기 위해서 시스템과 이것의 해시 h(610a)를 교환할 것이다. 시스템은 해당 해시 h(610a)를 이제 사용하고, 다음 시스템 해시 h(616a)를 생성한다(이것은 H(627ai)로 원래 라벨되었을 것인데, 계정(602a)에 대한 트랜잭션이 완료되면 해시 h(628i)가 그 트랜잭션에 대한 최종 시스템 해시이기 때문임).

[0399] 계정(604a)는 h(616a)를 획득하기 위해 시스템과 이것의 해시 h(614a)를 교환할 것이다. 시스템은 이제 다음 시스템 해시 h(620a)를 생성하기 위해 해당 해시 h(614a)를 사용한다.

[0400] 계정(606a)는 h(620a)를 획득하기 위해서 시스템과 이것의 해시 h(718a)를 교환할 것이다. 시스템은 다음 시스템 해시 h(624a)를 생성하기 위해서 해당 해시 h(618a)를 이제 사용한다.

[0401] 계정(602a)가 이것의 중간 또는 트랜잭션 해시를 생성하면, 이것은 해당 해시 h(622a)를 시스템 해시 h(624a)로 교환할 것이다. 시스템은 다음 시스템 해시 h(628a)를 생성하기 위해서 해당 해시 h(622a)를 이제 사용한다.

[0402] 계정(606a)가 이것의 중간 또는 트랜잭션 해시를 생성하면, 이것은 해당 해시 h(626a)를 시스템 해시 h(628a)로 교환할 것이다. 시스템은 다음 시스템 해시 h(632a)를 생성하기 위해 해당 해시 h(626a)를 이제 사용한다.

[0403] 계정(604a)가 이것의 중간 또는 트랜잭션 해시를 생성하면, 이것은 해당 해시 h(630a)를 시스템 해시 h(632a)로 교환할 것이다. 시스템은 다음 시스템 해시 h(미도시; 636a)를 생성하기 위해서 해당 해시 h(630a)를 이제 사

용한다.

[0404] 해시 체인은 시스템이 트랜잭션을 처리하고, 해당 트랜잭션을 감사하고, 동시에 해당 트랜잭션에 의해 전송되거나 생성된 데이터를 인증하게 한다. 이러한 단계들은 이제 동시에 발생한다. 장치가 트랜잭션을 감사 시스템으로 정직하게 보고한다고 가정할 필요가 없다. 트랜잭션은 감사를 생성하고, 감사는 트랜잭션을 생성한다.

[0405] 이것은 프로그램된 장치에 의해 수행되는 트랜잭션의 특성을 모두 변경한다. IoT 장치를 포함하는 임의의 프로그램된 장치는 트랜잭션, 이것의 감사 및 인증이 동시에 발생되므로, 이것과 임의의 다른 장치 간의 트랜잭션들 및 데이터를 이제 유효하게 하고, 의존(또는 신뢰)할 수 있다.

[0406] 장치가 정확한 트랜잭션의 레코드를 감사 시스템으로 전송할 것이라고 가정할 필요가 없는데, 해당 트랜잭션 및 감사가 동일한 프로세스의 일부로 생성되기 때문이고, 이 동시에 발생하는 특성은 감사 추적의 증거 값(evidential value)의 품질을 변경한다. 각 장치는 다른 것에 의해 전송되는 정보에 의존(또는 정보를 신뢰)할 수 있는데, 다른 것의 정직에 관해서는 가정하지 않아도 된다. 전송 및 수신되는 데이터는 거래되는 데이터, 및 인증 및 감사되는 데이터이다.

[0407] 룩-업 서비스와 결합될 때, 이전에 상호 작용하지 않은 장치들은 이제 서로 인증하고, 각각 수행하는 서비스들 또는 기능들을 결정하고, 그런 다음 상호 간에 통신하고, 임의의 인간이 개입할 필요없이 프로그램된대로 작업들을 수행하기 위해 해당 통신에 의존(또는 해당 통신을 신뢰)할 수 있다.

[0408] 해시 체인은 IoT 장치들을 포함하는 프로그램된 장치들이 온-라인 및 오프-라인 모두에서 동작하게 한다. 장치들이 오프-라인일 때, 타임 스탬프들, 해당 장치의 클럭 스크류(또는 시계 왜곡)에 대한 정보, 장치의 고유한 트랜잭션 ID(), 및 트랜잭션 정보에서의 기타 동기화 정보를 포함하는 경우, 그것들은 그것들의 서버들이 장치들 또는 제3 파티 서버들(또는 세 번째 파티 서버들, 세 번째 당사자 서버들)로부터 오프-라인 트랜잭션들의 레코드들을 최종적으로 수신할 때 그것들의 서버들이 각 트랜잭션에 대한 인과관계를 보존하는 정확한 타임 라인들을 재구성하게 한다. 해시 체인은 이것의 온-라인 및 오프-라인 모드들 모두에서 서버들이 트랜잭션적인 레코드들의 컨텐츠에 의존하게 한다.

[0409] 장치-간(inter-device) 통신을 보호하는 통신 보안 모델과 결합될 때, 장치들 및 서버들은 중간자 공격에 영향을 받지않는 방식으로 통신할 수 있다. Tereon은 IoT 및 다른 프로그램된 장치들이 안전하게 통신하고, 해당 장치들 간의 전송된 데이터에 의존(또는 전송된 데이터를 신회)하게 한다.

[0410] 하나의 이런 예는 산업용 센서들 및 컨트롤들의 세트로 동작하는 IoT 및 다른 프로그램된 장치들의 네트워크일 수 있다. 보안 모델은 룩-업 디렉토리 서비스를 사용하여 이러한 장치들이 그것들 간에 안전하게 통신하게 하고, 원본 컬렉션에 추가될 때 해당 장치들이 새로운 장치들과 상호 작용하게 한다. Tereon은 장치들- 그것들이 새로운 장치들을 인식하고, 새로운 장치들을 신뢰하게 함 -을 재구성할 필요가 없다. 해시-체인은 장치들이 그것들 간의 동신의 컨텐츠 및 타이밍을 신뢰하게 하고, 전송된 데이터의 진실성(veracity)에 대한 임의의 사람의 평가가 필요없이 운영자가 생성 및 전송된 데이터에 의존(또는 전송된 데이터를 신뢰)할 수 있게 한다. 제3 파티(또는 세 번째 파티, 세 번째 당사자)는 감사 및 인증 체인들이 이것의 전송과 동시에 발생하는 데이터와 인터페이스할 수 없다.

[0411] 보안 모델 및 룩-업 서비스와 결합될 때, 룩-업 서비스는 장치들이 인간의 개입이 필요없이 그것들이 신뢰하고 인증할 수 있는 애드혹 상호 연결들(ad hoc interconnections)을 생성하게 한다. 장치가 인증되고, 이것의 세부 사항들이 룩-업 서비스에 추가되면, 필요에 따라 다른 장치들은 해당 장치에 연결(또는 접속)할 수 있다. 해당 장치가 임의의 방식에서 손상되는 경우, 이것에 대한 모든 액세스는 동일한 룩-업 서비스를 통해 비활성화될 수 있다.

[0412] 시스템은 이것의 해시 체인 및 이것의 룩-업 서비스로부터 발생하는 추가 해택을 제공한다. 모든 장치들이 개별적으로 인증되고, 감사되므로, 시스템은 필요에 따라 특정 장치들이 그것들의 장치의 소프트웨어에 대한 업데이트를 다운로드하도록 지시할 수 있고, 장치들은 안전하고, 신뢰되는 소스들로부터만 수행할 수 있다. 룩-업 서비스는 특정 장치가 제공하고 사용하는 예를 들어, 서비스들, 인터페이스들, 및 데이터 포멧들을 상세하게 설명한다. 따라서, 장치가 특정 장치(survive)에 액세스하기 위해서 다른 장치에 연결하고자 하고, 요구되는 인터페이스 또는 포멧을 지원하는데 필요한 소프트웨어가 없는 경우, 이것 또는 이것이 연결 중인 장치 중 어느 하나, 또는 필요한 경우 장치들 모두는 2 개의 장치들이 서로 통신하게 하는 필요한 소프트웨어 또는 구성을 다운로드하기 위해 시스템 서버와 통신할 수 있다. 장치간(inter-device) 통신이 완료된 후에 장치들이 소프트웨어를 유지할지 여부는 장치 또는 장치들이 수행하는 서비스들, 및 해당 장치들의 용량에 의해 결정될 것이다.

해시-체인은 그것들이 해당 소프트웨어를 제거하더라도(그것들이 대시 통신할 때, 그것들은 이것을 재설치할 수 있음), 2 개의 장치들이 필요한 경우 그것들이 나중에 다른 장치 또는 서버로 업로드할 수 있는 장치간 통신의 완전한 감사 및 레코드를 유지할 것을 의미한다. 이 설비는 완전히 자동화된 IoT 장치에서부터 프로그램된 임의의 다른 장치- 결제 장치와 같은 -에 이르기까지 모든 타입의 장치로 확장된다.

[0413] *해시 체인의 분산 레코드(Distributed records of the hash chain)*

[0414] 전체 해시 체인의 분산 복제를 제공하기 위해, Tereon 시스템은 해당 서버의 현재 연결과 마지막 연결 사이에 발생한 모든 트랜잭션에 대해 해시 체인을 라이센스 서버, 조회 서버, 또는 다른 서버 세트와 같은 중앙 서버 집합에 업로드한다. 그런 다음 동일한 Tereon 시스템이 다른 Tereon 시스템에 해당하는 해시 체인을 다운로드할 수 있다. 이는 모든 Tereon 시스템에 대한 모든 트랜잭션에 대해 해시 체인의 분산 원장을 제공하지만, 각 트랜잭션에 대해 각 해시 체인을 다시 계산하지 않아도 되는 오버 헤드가 없다. 그러나 Tereon 시스템에 추가적인 스토리지 부담을 부과한다. 중앙 서버는 라이센스 및 검색 서버와 같은 글로벌 서버일 수도 있고 산업, 지역 또는 기타 제약 조건에 한정될 수도 있다. 해시 체인의 카피의 도달 범위를 제한함으로써, 이 변형의 계산 및 저장 부담을 줄일 수 있다.

[0415] 중앙 서버의 범위를 제한하는 대신에, 다른 시스템에서 업로드한 해시 체인을 다운로드 할 수 있는 시스템이 제한될 수 있다. 은행의 해시 체인은 다른 은행에 의해서 다운로드될 수 있지만, 해당 은행이 업로드 은행과 동일한 지역에 있는지 여부 또는 다른 은행과 거래했는지 여부에 따라 제약을 받을 수 있다. 마찬가지로, 병원 시스템은 동일한 지역의 병원에 의해서 업로드된 해시 체인만 다운로드할 수 있다. 유연성에는 제약이 없다.

[0416] Tereon에서 사용되는 해시 체인에는 매우 유용한 속성을 갖는다. 그것은 로컬 원장을 제공하지만 분산 인증을 제공한다. 그것은 트랜잭션 정보를 트랜잭션과 관련된 사용자 및 서비스에 비공개로 유지하지만, 해시에서 제공한 인증을 모든 서버, 서비스 및 장치에 분산한다. 영지식 증명으로 생성된 해시는 이를 나타낸다. 특정 트랜잭션과 관련된 시스템만 트랜잭션 정보를 보유한다. 그러나, 시스템과 상호 작용하는 모든 시스템과 장치는 해당 시스템의 초기 해시에 대한 정보를 포함하는 해시를 생성한다.

[0417] 분산 인증은 변조된 레코드(tampered record)를 숨기려고 하는 잠재적 사기범(potential fraudster)에게 계산상 불가능한 장벽을 제공하기 때문에 중요하다.

[0418] 블록 체인을 사용하면, 사기꾼은 변조된 레코드를 숨기고 블록 체인을 변경하여 잘못된 레코드를 유효한 레코드로 기록하기 위해 서버의 25-33 % 만 제어하면 된다. 일단 완료되면, 프로세스는 사실상 되돌릴 수 없다.

[0419] Tereon 해시 체인을 사용하면, 사기꾼은 모든 Tereon 서버, 모든 Tereon 서비스 및 모든 Tereon 장치를 제어하고 해당 서버 및 장치 모두에서 체인의 모든 해시를 다시 계산해야 한다. 이것은 계산상으로 실행 불가능하다.

[0420] 해시 체인은 블록 체인의 제안자가 후자에 대해 예측하는 것과 최소한 동일한 수준의 경제적 절감 및 경제적 효율성을 제공한다. 차이점은 Tereon 해시 체인이 실제로 그렇게 할 수 있다는 것이다; 블록 체인은 그 설계와 그 설계에 내제된 한계로 인해 그렇게 할 수 없다.

[0421] 이 시스템의 장점은 사기꾼이 모든 해시 및 해당 레코드와 연결된 연결된 해시를 다시 계산하지 않고도 데이터베이스에서 레코드를 삭제하거나 수정할 수 없음을 의미한다. Tereon이 시스템 해시가 없고 라이센스 서버에 연결되지 않은 단일 서버에서 작동하는 경우 이론적으로 가능할 수 있지만, 링크된 체인 중 하나가 다른 서버 또는 장치의 당사자와의 트랜잭션을 포함하면 사기꾼은 다른 서버 또는 장치의 모든 해시를 다시 계산할 필요가 있다. 이렇게 하는 어려움은 원본 레코드의 날짜와 시간 이후에 해시 체인과 상호 작용하는 각 추가 서버 또는 장치에 따라 기하 급수적으로 증가한다.

[0422] 해시 체인은 조직이 모든 장치에서 수집, 생성 또는 관리되는 데이터의 진실성을 보장하고, 레코드의 원본 컨텐츠와 무결성을 보장하고, 이전 레코드(earlier record)를 기반으로 한 트랜잭션의 컨텐츠와 무결성을 보장할 수 있도록 한다. 이는 모든 장치 또는 트랜잭션, 결제 장치에서부터 의료 장치, 교통 센서, 기상 센서, 수류 감지기 등에 적용될 수 있다.

[0423] 각 지역 원장(local ledger)이 각 개별 조직의 책임이기 때문에 이것은 확실한 관리 혜택을 갖지만, 그것들은 공유 강도(shared strength)에게 명확하게 명시된 책임과 의무를 제공하는 방식으로 다른 조직의 원장으로부터 배우고 의지할 수 있다. 해시 체인은 정보 및 트랜잭션 관리를 시행하고 지원하는 기술 도구를 만든다.

[0424] 또한, 해시 체인이 결제 시스템의 구성 요소로 사용될 때, Tereon은 결제 금액을 처리하고, 아키텍처가 오늘 결제 방식과 일치하므로, 그것은 Bitcoin과 같은 암호화 통화와 동등하거나 우월한 이점을 제공한다. 그것은 설

립된 결제 서비스 제공자와 중앙 은행 'Bitcoin beater'를 제공한다.

[0425] 해시 체인은 매우 안정적인 매우 빠른 인증을 가능하게 하기 때문에 Tereon 시스템에서 특히 흥미로운 부분이다.

[0426] Tereon의 고유한 기능 중 하나는 포괄적인 실시간 로그 및 감사 추적(audit trail)을 생성할 수 있다는 것입니다. Tereon 트랜잭션 레코드는 규정 요구 사항 및 비즈니스 요구 사항이 충족되는 동안 트랜잭션에 필요한 모든 데이터 및 메타 데이터와 함께 트랜잭션에서 요구하는 모든 키스트로크(keystroke, PIN 및 패스워드와 같은 실제 인증 크리덴셜 제외)를 포함한다. 이러한 레코드들을 손댄 흔적이 분명하게 보이게 하는 것이 중요하며, 그것들이 여러 서비스 제공자에 저장될 때 트랜잭션의 순서와 문제의 트랜잭션에 손댄 흔적이 분명하게 보이게 하는 것이 중요하다.

[0427] 블록체인은 이것을 할 수 없다. 그것은 레코드가 생성되었지만 권한이 부여되기 전에 트랜잭션 레코드만 승인할 수 있다. 블록체인은 여러 레코드를 연결하고 블록을 생성한 다음 이를 블록체인에 추가한다. 그것은 블록체인이 이전의 모든 트랜잭션과 관련된 정보가 포함된 블록들을 포함한다는 사실에 의존한다. 블록체인이 추가 블록을 추가함에 따라 이러한 블록의 존재 여부에 따라 블록체인 내의 레코드와 모든 이전 레코드의 유효성을 검사한다. 이로 인해 파일 크기가 커질 때 크기 조정 문제가 발생하고, 불일치가 발생하면 전체 지점(branch)에서 인증이 손실된다.

[0428] 블록체인 또는 그 파생물(derivative)을 사용하는 대신, Tereon의 해시 체인은 후속 트랜잭션(subsequent transactions)의 인증을 손상시키지 않으면서 의심스러운 레코드를 격리시키는 해싱 전략을 사용한다. 그것은 정적 레코드(static records)이든 실시간 트랜잭션(real-time transactions)이든 관계없이, 모든 레코드 유형에 맞게된 설계를 가짐으로써 스케일링 문제를 피할 수 있다.

[0429] 중간 해시(intermediate hashes)를 포함한 해시는 관리자가 해시 체인을 신속하게 탐색하여 해시 및 해당 레코드를 확인하고 확인하는 데 필요한 정보를 제공할 수 있다. 레코드 자체도 마찬가지이다.

[0430] 트랜잭션이나 작업(action)이 발생하면, 이전 해시(previous hashes)가 조정되며, 사용자와 시스템이 새 트랜잭션의 출력을 신뢰할 수 있음을 의미한다. 따라서 Tereon은 트랜잭션을 수행하기 전에 각 계정의 누적 합계(running totals)를 신뢰할 수 있다. 해시 체인의 유효성은 누적 합계가 올바른지 확인한다.

[0431] 해시 체인을 블록체인 및 그 파생물로부터 분리시키는 개정된 레코드, 삭제된 레코드 또는 변조된 레코드의 효과를 격리하는 것이 이 기능이다. 정의에 따라, 블록체인에서 성공적으로 숨겨진 수정되거나 변조된 레코드는 해당 블록체인의 전체 재계산에 영향을 미친다. 모든 블록체인을 수정해야 하므로, 전체 블록체인 커뮤니티의 민주적 결정 이외에 위조된 레코드 또는 거짓 레코드를 검색하고 수정할 방법이 없다. 보안 연구자가 블록체인 설계의 주요 결함으로 확인한 것은 이 기능이었다. 그 디자인은 바꿀 수 없다.

[0432] 해시 체인의 경우, 변조된 레코드는 공격자가 모든 후속 해시를 다시 계산할 수 없는 한 해시 체인의 나머지 부분에 영향을 줄 수 없다. 임의의 변경 이전의 해시가 유효하기 때문에, 해시 및 해시와 관련된 값을 기반으로 하는 모든 트랜잭션은 유효하다.

[0433] 오프라인 트랜잭션에 대한 덴드리틱 해시 체인(dendritic hash chain)은 오프라인 장치가 서버에 다시 연결할 수 있기 전에 해당 장치가 손실되거나 손상되더라도, 오프라인 장치에 의해서 수행된 오프라인 트랜잭션들을 등록할 수 있음을 의미한다.

[0434] 해시 체인은 블록체인 및 그 파생물만으로는 달성할 수 없는 오프라인 트랜잭션의 유효성을 완벽하게 지원한다. 블록체인의 복사본을 운영하는 노드는 블록을 확인하기 위해 온라인 상태여야 한다. 비트 코인 지갑은 오프라인에서 트랜잭션을 생성할 수 있지만, 온라인 상태가 되고 해당 트랜잭션의 레코드를 노드에 푸시할 때까지 해당 트랜잭션의 유효성을 검사할 수 없다. 노드들 중 하나가 블록체인에서 다음 블록을 생성하는 경쟁에서 이기고 해당 블록에 레코드를 추가할 때까지 트랜잭션의 유효성이 검사되지 않는다.

[0435] 디렉토리 서비스(Directory Service)

[0436] 운송 시스템, EMV(Europay, MasterCard, Visa)와 같은 결제 네트워크, 및 기타 레거시 시스템과 같은 기존 시스템은 허브 앤 스포크 아키텍처(hub and spoke architecture)를 사용한다. 이에, 모든 트랜잭션은 장애 또는 취약성의 잠재적인 단일 지점을 나타내며 확장에 비용이 많이 드는 중앙 유틸리티를 거친다.

[0437] 해시 체인 검증이 피어-투-피어 네트워크의 모든 요소에서 발생하므로, Tereon 시스템은 보안을 위한 해시 체인

이 중요한 이유인 피어-투-피어- 한 서버가 다른 서버와 직접 통신함 -이다.

[0438] 논의된 바와 같이, Tereon 시스템은 시스템의 크리덴셜 및 정보의 디렉토리인 디렉토리 서비스(216)를 갖는다. 디렉토리 서비스(216)는 특정 사용자에 관한 다수의 상이한 유형의 크리덴셜을 저장하기 때문에 사용자 또는 장치(218)가 어느 서버에 등록되어 있는지 또는 어느 서버가 특정 서비스 또는 기능을 제공하는지를 식별하고, 사용자(218)의 다수 인증 방법이 발생하게 한다. 예를 들어, 사용자(218)는 자신의 모바일 폰 번호, 이메일 주소, 지리적 위치, PAN(프라이머리 계정 번호) 등을 사용하여 인증될 수 있으며 매번 인증할 필요가 없도록 모든 것을 캐시한다.

[0439] 디렉토리 서비스(216)는 기본 서비스, 서버 및 실제 사용자 계정으로부터 사용자의 인증 ID를 분리하는 추상화 레이어를 제공한다. 이는 사용자(218) 또는 머천트 서비스에 액세스하기 위해 사용할 수 있는 크리덴셜들과 Tereon이 서비스 자체를 수행하는 데 필요한 정보 사이에 추상화를 제공한다. 예를 들어, 결제 서비스에서, 디렉토리 서비스(216)는 모바일 폰 번호와 같은 인증 ID 및 아마도 통화 코드를 서버 주소와 링크시킬 것이다. 사용자(218)가 은행 계정을 가지고 있는지 여부 또는 사용자(218)가 거래(뱅킹)하는 은행을 결정하는 방법은 전혀 없다.

[0440] 디렉토리 서비스(216)는 서비스 제공자가 서로를 볼 수 없게 하여 사용자 데이터의 보안이 제공되도록 다양한 서비스들 사이의 중개자 역할을 한다. 각 서비스는 해당 서비스에 특정한 필드(변수) 및 값 집합을 정의한다. 그러나 각 서비스는 서비스를 식별하는 특정 필드와 값을 포함한다.

[0441] 트랜잭션이 알지 못하는 당사자와의 트랜잭션이 완료될 때, 사용자(218)와 연관된 Tereon 서버는 디렉토리 서비스(216)에 URN(uniform resource name)을 전송하며, 디렉토리 서비스(216)는 사용자(218)에 의해서 요청된 서비스에 대한 결제 서비스 제공자의 Tereon 서버에 대한 IP 주소를 반송한다. 이는 트랜잭션이 피어-투-피어(peer-to-peer)를 기준으로 사용자(218)와 서비스 제공자 사이에서 직접 완료되는 것을 허용한다. 또한 Tereon 서버는 이후의 모든 트랜잭션에서 디렉터리 서비스(216)을 사용할 필요가 없도록 캐시에 IP 주소를 유지한다.

[0442] 이러한 추상화는 사용자 및 서비스 세부 사항에 보안 및 개인 정보를 제공하고, 일반 사용자 크리덴셜(public user credentials)에 영향을 주지 않고 기본 서비스를 추가 및 수정할 수 있는 유연성을 제공하며, 필요할 경우 각각 다른 것들과 격리되어 있을 수 있는 여러 서비스를 분할하고 지원할 수 있는 기능을 제공한다. 데이터 서비스의 어떤 필드도 트랜잭션을 시작하는 데 필요한 데이터를 포함하지 않으며 사용자의 인증 ID 이외의 사용자 데이터는 디렉토리 서비스(216)에 저장되지 않는다.

[0443] 그러나 Tereon 디렉토리 서비스(216)는 이보다 훨씬 더 중요하다. 그것은 다중 크리덴셜들을 지원한다. 따라서, 사용자(218)는 임의의 수의 크리덴셜을 결제 ID로서 사용할 수 있다. 예들은 모바일 폰 번호, PAN, 전자 메일 주소 등을 포함한다. 크리덴셜이 고유해야 Tereon이 지원한다.

[0444] 디렉토리 서비스(216)는 다수의 서비스를 지원할 수 있다. 이것은 다면적 크리덴셜- 또는 '사이킥 페이퍼(psychic paper)'-의 개념이 나오는 곳이다. 서비스 제공자가 디렉토리 서비스(216)상의 크리덴셜을 체크하면, 그 크리덴셜이 자신의 서비스에 대해 등록되어 있는지를 볼 수 있고, 그 크리덴셜을 서비스하는 Tereon 서버만 볼 수 있다. 서비스 제공자는 사용자(218)가 자격을 얻거나 등록할 수 있는 다른 서비스의 세부 사항을 볼 수 없다.

[0445] 예를 들어, 모바일 폰 또는 카드는 도서관의 도서관 카드 크리덴셜, 버스 또는 기차의 교통 티켓, 방 또는 시설에 액세스하기위한 보안 키, 회사 식당의 사내 결제 장치, 극장 티켓, 및 슈퍼마켓의 표준 결제 장치가 된다. 그것은 운전 면허증, 건강 관리 카드 또는 신분증 카드가 되어 서비스에 자격이 있음을 증명할 수 있다. 서비스가 필요한 경우 머천트의 장치 상에서 사진 ID를 가져올 수 있다. 장치가 될 수 있는 크리덴셜 유형에 대한 제한은 거의 없다.

[0446] 카드의 원래 외관을 위장하는 것은 어렵지만(카드가 OLED 커버 또는 컬러 전자 종이 커버를 포함하면 이를 수행할 수 있다. 예를 들어, 서비스가 카드에 특정 크리덴셜이나 서비스에 필요한 정보를 표시하도록 지시할 수 있다.), 폰 애플리케이션의 외관은 크리덴셜 및 서비스의 성격을 반영하기 위해 Tereon에 의해서 변경된다.

[0447] 역 룩-업 기능은 각 서버에 대해 구현될 수 있습니다. 이 기능을 사용하면 서버는 서버와 통신하는 서버가 라이센스를 받고 인증되었는지 여부를 확인할 수 있다. Tereon 장치들(카드들, 단말기들, 모바일들 또는 서버들) 사이의 모든 통신이 서명되어야 하므로, 이 기능은 필요하지 않다. 그러나 운영자가 역 룩-업을 통해 추가 보안을 필요로 하거나 원하는 상황이 있을 수 있다. 여기서 디렉토리 서비스(216)는 서비스, Tereon 서버 도메인 주소 Tereon 서버 번호, Tereon 서버 운영자, TTL(Time To Live), 단말기 인증 ID 등과 같은 다수의 필드를 포

함할 것이다. 여기서 서비스 태그는 트랜잭션 서비스가 아닌 서버 역 룩-업을 나타낸다.

[0448] 도 9는 2개의 서버, 즉 서버(202a) 및 서버(202b)를 갖는 예를 도시한다. 사용자(218)는 서버(202b)에 등록되고, 서버(202a)에 접속된 단말기를 통해 서비스에 액세스한다.

[0449] 단계 902에서, 사용자(218)는 자동으로 단말기에 자신을 식별하는 자신의 장치를 갖고 단말기에 자신을 식별한다. 또한, 단말기는 그가 스마트 장치를 사용하는 경우 사용자의 장치로 ID를 전달한다(사용자(218)가 카드를 사용하는 경우, 단말기는 장치가 마이크로-프로세서 카드인 경우에만 ID를 사용자의 장치로 전달한다. 이 경우, 카드는 단말기의 ID를 서버(202b)로 전달하기 위해 단말기를 통한 암호화된 터널을 통해 서버(202b), 그가 등록된 서버와 통신한다.).

[0450] 단계 (904)에서, 서버(202a)는 사용자의 장치에 의해 제공되는 ID를 취하고, 그것이 유지하는 목록에 대하여 ID를 체크한다. 그것은 ID를 보유하지 않으므로 이전에 사용자(218)를 다루지 않았다. 서버(202a)는 이제 디렉토리 서비스(216)에 접속한다. 디렉토리 서비스(216)는 서버(202a)의 통신 상의 서명을 검사하고 그것이 유효한 것으로 본다. 디렉토리 서비스(216)는 요청된 서비스(서버(202a)의 서명이 서버가 그 서비스에 대한 요청을 행할 권한이 있음을 확인함)에 대한 서비스 태그에 대하여 ID를 검색하고, 라이브 정보에 대한 캐시 타임과 함께 서버(202c)를 식별하는 정보로 응답한다.

[0451] 단계 906에서, 서버(202a)는 사용자의 장치가 서버(202b)에 등록되어 있는지를 확인하기 위해 서버(202b)에 접속한다. 서버(202a)는 단말기의 ID를 서버(202b)로 전달한다.

[0452] 단계 908에서, 이미 그렇게하지 않았다면, 서버(202b)는 단말기가 등록된 서버를 검색하도록 디렉토리 서비스(216)에 유사한 요청을 할 수 있다. 또한, 그것은 서버(202a)에서 단말기가 요청된 서비스에 대해 등록되었는지를 확인할 수 있다. 디렉토리 서비스(216)는 라이브 정보에 대한 캐시 타임과 함께 서버(202a)를 식별하는 정보로 응답한다.

[0453] 단계 910에서, 서버(202a)와 서버(202b)는 이제 필요한 트랜잭션을 수행하기 위해 서로 직접 통신한다. 이것은 결제에서 문 열림에 이르기까지 다양하다.

[0454] Tereon 서버 자체는 트랜잭션을 시작하는 데 필요한 정보를 포함하고, 라이센스가 부여되고 인증된 다른 서버들 또는 장치들과 만 통신한다.

[0455] 일단 서버들이 디렉토리 서비스(216) 및 서로 통신하면, 그것들은 데이터가 자신의 미니 디렉토리 서비스에서 만료될 때까지 데이터를 캐시한다.

[0456] 이 경우, Tereon 서버(202a)와 Tereon 서버(202b) 사이의 접속을 확립하기위한 통신은 명백하고 간단하다. 이것은 도 10에 도시된다.

[0457] 단계 1002에서, 사용자(218)는 자동으로 단말기에 자신을 식별하는 자신의 장치를 갖고 서버(202a)에 접속된 단말기에 자신을 식별한다. 또한, 단말기는 그가 스마트 장치를 사용하는 경우 사용자의 장치로 ID를 전달한다.

[0458] 단계 (1004)에서, 서버(202a)는 사용자의 장치에 의해 제공되는 ID를 취하고, 그것이 유지하는 목록에 대하여 ID를 체크한다. 그것이 보유하고 있는 데이터는 유효하므로, 서버(202a)는 장치가 요청된 서비스에 대해 그것에 여전히 등록되어 있는지를 확인하기 서버(202b)에 접속한다. 또한, 서버(202a)는 단말기의 ID를 서버(202b)로 전달한다. 서버(202b)는 장치가 등록되어 있음을 확인한다.

[0459] 서버(202a)의 캐시는 단말기의 ID에 대한 유효한 데이터를 포함하고 있으므로, 그것은 단말기가 그것에 여전히 등록되어 있는지를 확인하기 위해 서버(202b)에 접속한다. 서버(202b)는 이를 확인한다.

[0460] 단계 1006에서, 서버(202a) 및 서버(202b)는 이제 필요한 트랜잭션을 수행하기 위해 서로 직접 통신한다.

[0461] 캐시된 데이터가 서버에서 만료되면, 해당 서버는 이전처럼 디렉토리 서비스(216)에 접속한다. 사용자(218)가 다른 서버로 이동한 경우, 통신은 약간 다르다. 도 11은 이 경우를 도시한다. 차이점은, 현재 낡음(out-of-date) 캐시된 정보에 기초한 서버 (202b)와의 첫번째 통신은 서버(202a)가 디렉토리 서비스(216)에서 새로운 데이터를 검색하도록 강제할 것이라는 점이다.

[0462] 단계 1102에서, 사용자(218)는 자동으로 단말기에 자신을 식별하는 자신의 장치를 갖고 서버(202a)에 접속된 단말기에 자신을 식별한다. 또한, 단말기는 그가 스마트 장치를 사용하는 경우 사용자의 장치로 ID를 전달한다. 서버(202a)는 사용자의 장치에 의해 제공되는 ID를 취하고, 그것이 유지하는 목록에 대하여 ID를 체크한다. 그

것은 해당 ID를 보유하고, 캐시된 데이터가 ID가 서버(202b)에 등록되어 있음을 나타내는 것을 보여준다.

[0463] 단계 1104에서, 서버(202a)는 사용자의 장치가 서버 (202b)에 등록되어 있는지를 확인하기 위해 서버 (202b)에 접속한다. 서버(202a)는 단말기의 ID를 서버(202b)로 전달한다. 서버(202b)는 ID가 더 이상 등록되어 있지 않았다고 응답한다.

[0464] 단계 1106에서, 서버(202a)는 이제 디렉토리 서비스(216)에 접속한다. 디렉토리 서비스(216)는 서버(202a)의 통신 상의 서명을 검사하고 그것이 유효한 것으로 본다. 디렉토리 서비스(216)는 요청된 서비스에 대한 서비스 태그에 대하여 ID를 검색하고, 라이브 정보에 대한 캐시 타임과 함께 서버(202c)를 식별하는 정보로 응답한다.

[0465] 단계 1108에서, 서버(202a)는 사용자의 장치가 동일한 서비스에 대해 서버 (202c)에 등록되어 있는지를 확인하기 위해 서버(202c)에 접속한다. 또한, 서버(202a)는 단말기의 ID를 서버(202c)에 전달하고, 사용자의 장치로부터 ID에 대한 새로운 세부사항(또는 세부정보)으로 캐시를 갱신한다.

[0466] 단계 1110에서, 이미 그렇게하지 않았다면, 서버(202c)가 단말기가 등록된 서버를 검색하도록 디렉토리 서비스(216)에 유사한 요청을 할 수 있다. 또한, 그것은 서버(202a)에서 단말기가 요청된 서비스에 대해 등록되었는지를 확인할 수 있다. 디렉토리 서비스(216)는 라이브 정보에 대한 캐시 타임과 함께 서버(202a)를 식별하는 정보로 응답한다.

[0467] 단계 1112에서, 서버(202a)와 서버(202c)는 이제 필요한 트랜잭션을 수행하기 위해 서로 직접 통신한다.

[0468] 디렉토리 서비스(216)는 구 및 신규(old and new)인 사용자(218)가 등록한 사용자 ID의 완전한 흔적을 그것들이 사용자(218)에게 할당된 날짜와 함께 항상 유지할 것이다.

[0469] 서버(202c)는 ID가 그것에 등록된 날짜부터 등록된 ID와 관련된 정보만을 유지한다. 서버 (202b)는 그 ID를 서비스한 기간과 관련된 데이터를 보유할 것이다.

[0470] 디렉토리 서비스(216)에 의해 제공되는 추상화 레이어는 서비스를 세그먼트 화함에 따라 더 나아 간다. 따라서, 위의 예에서, 서버(202a)는 요청된 서비스에 대해 사용자의 장치를 등록한 서버를 식별하는 정보만을 요청할 수 있다.

[0471] 서버(202a)는 장치와의 각각의 통신에 서명해야 하고, 그 서명은 통신이 관련된 서비스를 식별할 것이다. 서버가 둘 이상의 서비스를 제공할 수 있는 경우, 그것은 해당 서비스 각각에 대해 개인 키를 가지며, 해당 키를 사용하여 관련 통신에 서명한다.

[0472] Tereon 서버 자체- 위의 경우에는 서버(202a 및 202b) -는 제공되는 태그 또는 정보로부터 사용자의 계정 데이터를 식별하는 룩업 정보를 포함한다. 따라서, 서버(202b)만이 사용자 장치의 ID를 사용자의 계정에 매핑하는 데이터를 포함한다; 디렉토리 서비스(216)의 정보는 단순히 서버(202b)에 대한 포인터이다. 사용자의 장치는 다양한 서비스를 위해 다른 서버에 쉽게 등록될 수 있다. Tereon 서버가 정확한 서버를 찾을 수 있게 하는 것은 사용자의 장치 ID와 서비스를 정의하는 크리덴셜의 조합이다.

[0473] 일단 서버(202a)가 서버(202b)와 통신하고, 서비스 태그, 사용자 ID, 및 임의의 다른 관련 트랜잭션 데이터(예를 들어, 연령, 통화, 금액 등)를 전달하면, 서버 (202b)는 관련 사용자의 데이터를 검색하고, 트랜잭션의 측(side)을 수행한다. 서버(202a)는 결코 사용자의 데이터를 보지 못한다. 볼 수 있는 것은 사용자의 인증 ID와 서버(202b)에 의해 전달된 트랜잭션 데이터이다.

[0474] 마찬가지로, 서버(202b)는 단말기가 접속된 계정을 식별하는 정보를 결코 보지 못한다. 그것은 단순히 서버(202a)에 의해 전달된 단말 ID 및 트랜잭션 데이터를 본다.

[0475] *사이킥 페이퍼 - 다면적 크리덴셜(Psychic paper - the multifaceted credential)*

[0476] 디렉토리 서비스의 보다 흥미로운 효과 중 하나는 크리덴셜이 필요할 때 특정 서비스에 맞게 조정된 애드 혹 다면적 크리덴셜(ad hoc multifaceted credentials)을 만드는 기능이다. 디렉토리 서비스가 이러한 크리덴셜을 제공할 수 있도록 디렉토리 서비스를 만들어진 시점에서 서비스는 구상되었을 필요 없다. 이것은 '사이킥 페이퍼(psychic paper)'로 알려져 있다.

[0477] 애드 혹 다면적 크리덴셜은 사용자의 장치가 특정 서비스에 필요할 수 있는 크리덴셜이 되며 더 이상 자격이 없음을 의미한다. 그것은 서비스 인증, 권한 부여, 또는 서비스로부터 기타 혜택을 받기 위해 필요한 정보를 정확하게 제공하고, 그것은 서비스 제공 업체가 보는 모든 것이다.

[0478] 예로서, 사용자(218)는 자신의 은행으로부터의 결제 서비스 및 그의 지역 도서관에서의 도서관 차용 서비스(library borrowing service)와 같은 다수의 상이한 서비스에 대해 등록했다. 그는 Tereon에 등록할 때 생일을 제공해야했기 때문에 자동으로 연령 확인 서비스(age verification service)를 액세스할 수 있다.

[0479] 도 12는 사용자(218)가 요청한 서비스에 따라 디렉토리 서비스(216)가 요청하는 서버(서버 (202a))를 2개의 다른 서버(서버들(202b 및 202c))로 향하게 할 수 있는 방법을 도시한다. 필요한 경우 별도의 서비스를 위한 두 개 이상의 개별 디렉토리 서비스는 사용될 수 있다. 중요한 것은 트랜잭션 데이터가 추상화의 일부이며 기본 계정 데이터와 분리되어 있다는 것이다.

[0480] 사용자(218)는 예를 들어, 바(bar, 서비스 2)에서 알콜 음료를 구매하기 위해 나이를 확인할 필요가 있다. 이 경우, 서버들(202a 및 202b)보다는 서버들(202a 및 202c) 사이에 있지만, 단계들 1202 내지 1210은 도 9의 단계들 902 내지 910로서 수행된다. 따라서, 단계 1210에서, 서버(202a) 및 서버(202c)는 서로 직접 통신한다. 이 경우, 서버(202a)는 사용자(218)가 21세 이상이라는 것을 검증하기를 원한다. 서버(202c)는 그가 21 세 이상임을 단순히 확인한다.

[0481] 운영자가 법적 또는 규제 요구사항으로 인해 추가 확인을 요구하면, 서버(202c)는 단말기에 표시하기 위해 사용자(218)의 여권-유형 이미지를 보낼 수 있고, 이를 통해 운영자는 그 또는 그녀가 실제로 사용자(218)와 대화하고 있음을 볼 수 있다. 사용자(218)가 자신을 서버(202a)에 이미 식별하였으므로 그렇게 할 필요는 거의 없지만, 서버는 정확한 사용자임을 추가 확인하기 위해 사용자(218)가 답변하도록 질문을 보낼 수도 있다. 운영자는 사용자의 실제 나이 또는 필요하지 않은 개인 정보를 결코 볼 수 없다. 운영자가 알 필요가 있는 것은 사용자(218)가 알콜 음료를 살만큼 충분히 나이기 있다는 것이다. 사용자(218)가 자신의 음료를 결제하기 위해 그의 장치를 사용하면, 서버(202a)에 접속된 단말기는 서버(202c)에 다시 접속할 것이지만, 이번에는 결제 서비스(서비스 1)를 위한 것이다.

[0482] 사용자(218)는 이제 자신의 지역 도서관으로 가서 책을 빌리고(서비스 3) 싶어한다. 단계 1212에서, 사용자(218)는 자동으로 단말기에 자신을 식별하는 자신의 장치를 갖고 도서관 내의 단말기에 자신을 식별한다. 도서관 내의 단말기는 서버(202b)에 접속된다. 또한, 단말기는 그가 스마트 장치를 사용하는 경우 사용자의 장치로 ID를 전달한다.

[0483] 단계 1214에서, 서버(202b)는 사용자의 장치에 의해 제공되는 ID를 취하고, 그것이 유지하는 목록에 대하여 ID를 체크한다. 그것은 해당 ID를 보유하고 있지만 캐시가 오래되었다. 서버(202b)는 이제 디렉토리 서비스(216)와 접속한다. 디렉토리 서비스(216)는 서버(202b)의 통신 상의 서명을 검사하고 그것이 유효한 것으로 본다. 디렉토리 서비스(216)는 요청된 서비스에 대한 서비스 태그에 대하여 ID를 검색하고, 라이브 정보에 대한 캐시 타임과 함께 서버(202c)를 식별하는 정보로 응답한다.

[0484] 단계 1216에서, 서버(202b)는 사용자의 장치가 서버(202c)에 등록되어 있는지를 확인하기 위해 서버(202c)에 접속한다. 또한, 서버(202b)는 단말기의 ID를 서버(202c)에 전달하고, 사용자의 장치로부터의 ID에 대한 새로운 세부사항(또는 세부정보)으로 캐시를 갱신한다.

[0485] 단계 1218에서, 이미 그렇게하지 않았다면, 서버(202c)는 단말기가 등록된 서버를 검색하도록 디렉토리 서비스(216)에 유사한 요청을 할 수 있다. 또한, 그것은 서버(202b)에서 단말기가 요청된 서비스에 대해 등록되었는지를 확인할 수 있다. 디렉토리 서비스(216)는 서버(202b)를 식별하는 크리덴셜로 응답한다.

[0486] 단계 1220에서, 서버(202b)와 서버(202c)는 이제 필요한 트랜잭션을 수행하기 위해 서로 직접 통신한다. 서버(202b)는 사용자(218)가 책을 빌릴 수 있는지(서비스 3)를 알고 싶어하고, 서버(202c)는 사용자가(218)가 책을 빌릴 수 있는 도서관 서비스에 등록되어 있음을 확인한다(Tereon 운영자가 도서관에 제공하는 서비스이다). 사용자(218)가 책을 빌리는데 요금을 지불하기 위해 자신의 장치를 사용할 필요가 있다면, 단말기는 서버(202c)에 다시 접속할 것이지만, 이번에는 결제 서비스(서비스 1)를 위한 것이다.

[0487] 서버(202c)는 임의의 서비스를 도서관에 제공할 필요가 없다. 사용자(218)는 서버(202d)(미도시)와 같은 다른 서버에 쉽게 등록할 수 있으며, 이 경우 서버(202d)는 사용자(218)가 책을 빌릴 수 있음을 서버(202b)에 확인한다. 중요한 것은 첫 번째 경우에 서버(202a)에서 사용자(218)가 21세 이상임을 확인하는 것뿐이다. 그것은 그가 책을 빌릴 수 있다는 것을 알지 못하며 사용자(218)가 Tereon에 의해 결제할 수 있다는 것을 알지 못한다. 마찬가지로, 서버(202b)는 사용자(218)가 책을 빌릴 수 있음을 알지만, 그가 특정 연령을 넘었거나 Tereon에 의해 결제할 수 있다는 것을 알 수 없다.

[0488] 요청 서버는 특정 트랜잭션에 대한 크리덴셜 집합을 어셈블해야하는 경우 별도의 서버들에 여러 요청들을 할 수도 있다. 예를 들어, 사용자(218)가 연령 제한이 있는 영화를 빌리고 싶어한다고 가정한다. 이 경우 요청 서버는 사용자의 나이를 확인하기 위한 하나의 요청과 도서관으로부터 영화를 빌려 오기 위해 등록되었음을 확인하는 두 가지 별도의 요청을 한다. Tereon은 도서관에서 요구하는 트레덴셜 집합을 구성하기 위해 검증된 개별 크리덴셜을 수집한다.

[0489] 디렉토리 서비스(216)의 구조는 개별 크리덴셜을 전달하는 서버가 분리되도록한다. 따라서 요청 서버는 특정 서비스를 사용자(218)에게 전달할 수 있는지 여부를 확인하는 데 필요한 크리덴셜 세트를 구성하는 데 필요한 개별 크리덴셜을 얻기 위해 임의의 수의 서버에 질의할 수 있다.

[0490] 도 13은 서버(202a)가 사용자(218)에게 서비스를 제공하기 위한 다면적인 크리덴셜(multifaceted credential)을 구성하기 위해 3개의 서버(202c, 202d, 및 202e)로부터 크리덴셜을 획득할 필요가 있는 경우를 나타낸다. 예를 들어, 서버(202d) 상에서 서비스 2는 필름을 렌트하는 서비스이고, 서버(202c)로부터 제1 크리덴셜(첫번째 크리덴셜)으로서의 연령 검증(age verification), 서버(202d)로부터 멤버쉽 크리덴셜(membership credential), 및 서버(202e)로부터 충분한 자금 크리덴셜(sufficient funds credential)을 필요로 한다.

[0491] 관계는 반드시 일대일이 아니며, 3개의 서버 각각이 하나의 크리덴셜만 보유한다. 3개의 서버들 중에서 임의의 서버는 각각 하나 이상의 크리덴셜을 서버(202a)에 전달할 수 있다. 이것들은 서버(202a)에 하나의 크리덴셜만을 전달할 수있다. 크리덴셜의 수는 무관한다. 중요한 것은 서버(202a)에서 사용자(218)가 서비스에 액세스할 수 있게 하는 데 필요한 크리덴셜을 획득하기 위해 하나 이상의 외부 서버와 접촉하는 것이다.

[0492] 사용자(218)가 단말기에 액세스하는 서버(202a)는 일부 서비스를 사용자(218)에게 전달하기 위해 필요한 크리덴셜을 이미 보유하고 있을 수 있다. 그러나, 데이터 보호의 목적 상, 사용자(218)는 서버(202a)에 특정 세부사항(또는 세부정보)(예를 들면, 그의 나이 등)을 제공하기를 원하지 않는다. 모든 서버(202a)에서 사용 (218)가 특정 연령을 넘었거나 특정 물건을 주문하도록 허용되는지 검증할 필요가 있는 경우, 그 질문을 확인하거나 거부할 서버들에 간단히 접속할 수 있다. 이것은 전자 상거래 웹 사이트에 매우 유용하다. 그것들은 정확한 사실을 모른 채 특정 사실이나 매개 변수를 확인할 수 있다. 본질적으로, 디렉토리 서비스(216)는 영지식 증명 제공자(zero-knowledge proof provider) 또는 기밀 공증인(confidential notary)으로서 역할을 할 수 있다. Tereon은 그 사실이 무엇인지 공개하지 않고 사실 또는 매개 변수를 서버 202a에 증명하거나 반증할 수 있습니다.

[0493] 따라서, 특정 서비스에 대한 크리덴셜은 202a, 202c, 202d, 202e 및 다른 서버로부터의 크리덴셜을 포함할 수 있다. 크리덴셜은 한 서버에 있거나 여러 서버에 분산될 수 있습니다.

[0494] 이는 공개할 필요없는 정보를 공개할 필요없이 개인 및 조직이 서비스 받을 자격이 있음을 증명할 수 있기 때문에 매우 강력하다. 다시, 전자 상거래 웹 사이트의 예를 취하면, 사용자(218)는 자신의 이름과 주소를 웹 사이트에 등록할 수 있다. 그러나, 그의 은행은 결제 크리덴셜을 보유하고, 정부 서버는 그가 제한된 아이템을 구입할 수 있는 권한이 있다는 사실을 등록하고, 지역 철도 회사는 그의 여행 허가를 보유하고, 그의 보건 당국의 서버는 그의 나이를 확인할 수 있다.

[0495] 서비스에 대한 크리덴셜의 애드 혹 집합을 조합하는 방법은 사용자와 해당 장치에만 적용되지 않는다. 다른 시간에 다른 서비스에 연결해야 하는 IoT 장치와 같은 자율 센서, 장치 및 서비스에도 마찬가지로 적용할 수 있습니다. 이러한 크리덴셜의 집합이 필요할 때 그것들은 해당 서비스에 필요한 크리덴셜을 간단하게 조합할 수 있다.

[0496] *계정 전환(Account switching)*

[0497] 종종 새로운 시스템의 채택을 지연시키는 주요 문제는 손실 또는 서비스 중단없이 레거시 시스템에서 새로운 시스템으로 데이터를 전송하는 것이 어렵다는 인식이다. 동일한 문제는 시스템 업그레이드에 영향을 미치며 운영자는 업그레이드 또는 업데이트시 데이터 손실의 위험성에 대한 인식으로 인해 업그레이드 및 업데이트보다는 초기 하드웨어 및 소프트웨어 구성을 유지하려고 하는 경우가 많다.

[0498] 디렉토리 서비스(216)는 데이터, 계정 및 구성 정보(configuration information)를 하나의 서버 또는 데이터 저장소에서 다른 서버 또는 데이터 저장소로 원활하게 이동시키는 메커니즘을 제공함으로써 이러한 문제를 해결한다. 기관들 간의 실시간 계좌 이체를 지원하는 블록들 중 하나는 미정인 결제(in-the-air payments)를 파악하고 처리하는 방법에 대한 질문이다. 이 산업은 현재 총 18개월이 걸리는 계정 이전 시스템을 가지고 있다(초기

전환의 경우 7일 후, 결제 또는 이체를 받기까지 18개월). 이는 한 데이터 저장소에서 다른 데이터 저장소로 데이터 집합을 전환하는 데에도 적용할 수 있다.

[0499] 디렉토리 서비스(216)는 기본 서비스, 서버 및 실제 사용자 계정으로부터 사용자의 인증 ID를 분리하는 추상화 레이어를 제공한다. 따라서, 사용자(218)는 자신의 장치가 등록된 서비스 및 기본 서버를 변경하는 동안 자신의 인증 ID를 유지할 수 있다.

[0500] 계정 전환 프로세스는 예를 통해 가장 잘 설명된다. 이 예에서, 사용자(218)는 은행 A와 거래한다(bank). 도 14는 은행 A 및 그것의 Tereon 서버(202a)와의 사용자 관계를 도시한다. 사용자(218)가 아직 고객이 아니더라도, 은행 B는 서버(202b)상에서 Tereon을 지원한다. 사용자(218)는 그의 계정을 은행 A에서 은행 B로 이동시키기로 결정한다.

[0501] 도 15는 사용자(218)가 자신의 계정을 뱅크 A로부터 뱅크 B로 이체하기 위해 착수하는 프로세스를 도시한다. 이 예에서, 사용자(218)는 뱅크 A로부터 초과 인출되지 않고 대출을 갖지 않는다.

[0502] 단계 1502에서, 사용자(218)는 은행 B에 계정을 열고, 그의 카드 및 모바일 폰을 그 은행 및 Tereon 서버(202b)에 등록한다.

[0503] 단계 1504에서, 은행 B의 Tereon 서버(202b)는 Tereon 디렉토리 서비스(216)상에서 사용자의 모바일 번호 및 카드의 PAN을 검색하고, 이것들 모두가 은행 A에 등록되어 있는지를 검출한다.

[0504] 단계 1506에서, 은행 B의 Tereon 서버(202b)는 사용자(218)가 등록을 은행 B로 이동시키길 원하는지 확인하기 위해 사용자(218)와 접촉하고, 사용자(218)는 특별히 이 목적을 위해 사용자(218)에게 보내진 추가 인증 코드를 입력함으로써 이것을 확인한다.

[0505] 단계 1508에서, 은행 B의 Tereon 서버(202b)는 이제 은행 A의 서버(202a)에 접속하고, 사용자(218)가 자신의 계좌 및 ID에 대해 은행 B로의 이동을 요청하고 이것을 확인했다는 것을 은행 A의 서버(202a)에 알린다.

[0506] 단계 1510에서, 은행 A의 Tereon 서버(202a)는 이제 사용자(218)에게 계좌를 이동하기를 원하는지를 확인하는 요청을 전송하고, 사용자(218)는 자신의 이동을 확인한다.

[0507] 단계 1512에서, 은행 A의 Tereon 서버(202a)는 이제 이를 은행 B의 Tereon 서버(202b)와 확인하고, 사용자의 계좌 등록, 잔액, 구성(configurations), 결제 지시 등을 사용자 B의 서버(202b)에 알린다. 은행 B의 서버(202b)는 이러한 계정들을 은행 A와 동일한 방식으로 설정하거나 제공 권한이 부여된 서비스를 제공하기 위해 할 수 있는 한 가까이 배치한다.

[0508] 예를 들어, 사용자(218)는 GBP, USD 및 EUR를 보유할 수 있게 하는 은행 A에 3개의 분리된 통화 계정들(currency accounts)을 갖는다. 유감스럽게도, 은행 B는 GBP와 USD 계정만 제공하지만, 그는 어떤 계좌라도 EUR를 결제하고 받을 수 있다. 은행 B의 서버(202b)는 사용자가 계정을 개설하면 이를 사용자(218)에 알리고, 사용자는 EUR를 GBP로 변환하기로 결정한다. 그런 다음, 은행 B는 은행 A에게 GBP로 EUR을 보내도록 지시한다.

[0509] 단계 1514에서, 은행 B의 Tereon 서버(202b)는 이제 사용자의 ID가 서버(202b)에 등록되었다는 것을 디렉토리 서비스(216)에 통지한다.

[0510] 단계 1516에서, 은행 B의 Tereon 서버(202b)는 디렉토리 서비스(216)에 사용자의 ID를 등록했다는 것을 은행 A의 서버(202a)에게 알려주고, 은행 A에게 잔액을 이체하도록 지시한다.

[0511] 단계 1518에서, 은행 A는 더 이상 사용자의 ID를 관리하지 않는다는 것을 디렉토리 서비스(216)에서 확인한다. 디렉토리 서비스(216)는 새로운 ID 등록에 대한 시작 날짜 및 시간(start date and time)을 은행 B에 설정하고, 은행 A에 대한 구 등록(old registration)에 대해 종료 날짜 및 시간(end date and time)을 필드에 설정한다. 이제 은행 A는 디렉토리 서비스를 설정하여 더 이상 사용자 계정을 보유하지 않은 사용자(218)에게 결제하려고 시도하는 임의의 서버에 통지하고, 해당 서버에 사용자의 세부사항(또는 세부정보)을 디렉토리 서비스(216) 내에서 검색하게 명령한다. 종료 날짜 필드(end date field)에 날짜와 시간을 입력하여 이를 수행합니다. 이제, 은행 B는 처음에는 은행 A로 연결되었던 사용자(218)에게 결제된 모든 결제를 수신할 것이다.

[0512] 디렉토리 서비스(216)는 이제 사용자(218)가 새로운 계정으로 전환한 후에 사용자의 구 계정(old account)에 이루어진 결제인 미정인 결제(in-the-air payments)를 캐치할 수 있다. 유사한 방식으로, Tereon은 구 계정(old account)에서 지급될 예정인 연기된 결제들(deferred payments)도 받을 수 있다. 잔액이 양도된 후에는 새로운 계좌에서 출금될 것이고, 작업(task)은 며칠, 몇 주 또는 몇 개월이 아닌 몇 분이 걸린다.

[0513] 단계 1520에서, 은행 A는 잔액을 은행 B로 이체한다. 은행 B는 은행 A에게 자금을 수신했음을 통지한다.

[0514] 단계 1522에서, 은행 A는 사용자의 계정를 폐쇄하고, 그렇게 했고 새로운 은행으로 잔액을 이체했다는 것을 사용자(218)에게 알린다.

[0515] 단계 1524에서, 은행 B는 은행 A로부터 자신의 잔액을 수신했다는 것을 사용자(218)에게 알린다.

[0516] 사용자(218)가 은행 A의 하나 이상의 그의 계좌에서 초과 인출하고, 은행 B가 그의 사업을 수락하기로 동의하면, 단계 516 및 520에서 은행 B는 잔액을 은행 A로 이체하고 은행 B의 사용자에 대응하는 계정들은 초과 인출될 것이다. 사용자(218)는 은행 B에 자신의 계정을 이체하기 전에 초과 인출(overdraft)을 클리어하기 위해 뱅크 A의 계정들 간에 자금을 이체하기로 결정할 수도 있다.

[0517] 결제의 경우, Tereon 넘버링 시스템(Tereon numbering system)은 사용자, 조직, 계정, 서비스 유형 및 트랜잭션을 구분한다. 그것들은 모두 별도의 넘버링 시스템(번호 체계)를 갖는다. 이러한 특성들은 디렉토리 서버에게 사용자(218)가 그의 계정을 새로운 서비스 제공자에게 실시간으로 이동시키는 프로세스를 관리하도록 한다. 디렉터리 서비스(216)의 구조는 트랜잭션을 실시간으로 처리할 있는 기능과 함께 사용자들에게 며칠이 아닌 단 몇 분만에 계정을 변경할 수 있도록 한다.

[0518] 상술한 바와 같이, 디렉토리 서비스(216)는 모든 트랜잭션의 실시간 처리와 함께 미정인 결제(in-the-air payment)와 같은 미정인 트랜잭션(in-the-air transaction)의 문제를 제거한다. Tereon을 사용하면, 트랜잭션은 단순히 미정인 상태(in-the-air state)에 진입할 수 없다. 그것들은 완료하거나 취소된다.

[0519] Tereon은 은행 계정 이동성(bank account portability)과 같은 계정 이동성(account portability)이라는 개념을 지원하는데, 시장에서의 경쟁을 증가시킬 수 있는 기능이지만, 은행 및 규제 기관은 구현이 불가능하다고 믿는다. Tereon은 계정 세부사항(또는 세부정보)을 직접 사용하지 않고 각 납부자(payer)와 수취인(payee)을 식별하기 위해 별도의 크리덴셜을 사용하기 때문에, 사용자(218)과 사용자의 은행 계정 세부사항(또는 세부정보) 간에 추상화를 삽입한다. 디렉터리 서비스(216)가 제공하는 추상화는 계정 스위칭 및 이동성을 용이하게 한다.

[0520] *크리덴셜 변경(Changing credentials)*

[0521] 디렉토리 서비스(216)는 운영자 및 사용자가 기존 ID 크리덴셜을 새로운 크리덴셜로 대체하고, ID의 이전 사용자에 대한 트랜잭션과의 혼동 없이 과거 크리덴셜 재사용할 수 있게 한다. 디렉토리 서비스(216)에 의해 제공되는 추상화 레이어는 Tereon이 이것을 가능하게 하도록 한다.

[0522] 사용자(218)가 자신의 계정을 다른 서버에 이체하면, 사용자(218)는 PAN과 같은 특정 크리덴셜을 보유할 수 있거나 또는 서버가 사용자(218)에게 새로운 크리덴셜을 발행할 수 있다. 후자의 경우, 원래 서버는 거의 즉시 크리덴셜을 다시 사용할 수 있다. 각 크리덴셜은 시간 및 날짜 스탬프를 갖고 그것이 사용자(218)에게 발행될 때를 반영하기 때문에, 특정 크리덴셜의 새로운 사용자(218)는 그 크리덴셜을 거의 즉시 사용할 수 있다.

[0523] 각 크리덴셜은 특정 서버에서 특정 사용자에게 발급된 시간 및 날짜 스탬프를 갖는다. 각 트랜잭션도 시간 및 날짜 스탬프가 유지하고 각 Tereon 서버도 각 트랜잭션에 사용된 크리덴셜을 보유하므로, Tereon은 트랜잭션을 올바른 목적지로 라우팅하기 위해 이러한 컴퍼넌트들을 사용한다. 예를 들어, 사용자(218)는 크리덴셜 A(예를 들어, 모바일 폰 번호)를 갖는 머천트로부터 무언가를 구매할 수 있고, 다른 크리덴셜 B(예를 들어, 새로운 모바일 폰 번호)를 사용할 필요가 있을 때 며칠 후 다른 은행으로 이동할 수 있다. 아이템이 결함이 있을 때, 나중에 사용자(218)는 아이템을 머천트에게 되돌려 보낸다. 머천트는 트랜잭션을 찾고 환불을 누르면 된다. 원래 트랜잭션이 크리덴셜 A를 사용했지만, 크리덴셜 A에 대한 서버는 크리덴셜의 변경을 나타내는 시간 및 날짜 스탬프를 보고한다. 머천트의 서버는 크리덴셜 A를 찾고, 트랜잭션 당시 크리덴셜 A를 사용했던 사용자(218)가 현재 크리덴셜 B를 사용하고 있음을 발견한다. 이제 서버는 크리덴셜 B에 대한 서버- 크리덴셜 B에 대한 사용자(218)가 트랜잭션 당시 크리덴셜 A를 사용했음을 확인함 -에 접속하고, 서버는 환불 프로세스를 시작한다.

[0524] Tereon의 보안 모델이 모든 통신에 서명해야 하므로, 사용자 A는 사용자 B가 사기성이 없음을 확신할 수 있습니다. 서버(202b)는 라이센스 서버로부터의 유효한 라이센스를 갖는 경우에만 그 통신에 서명할 수 있고, 서버(202b)가 장치의 라이센스를 발행하고 확인할 것이므로 사용자 B의 장치는 서버(202b)가 유효한 경우 그 통신을 서명할 수 있다. 사용자 B가 트랜잭션을 승인하거나 장치의 애플리케이션에 액세스하는 데 필요한 정확한 크리덴셜을 알고 있지 않으면, 사용자 B는 트랜잭션을 완료할 수 없습니다.

[0525] 또 다른 예에서, 사용자는 자신의 전화 번호부(phone directory)에 연락처의 모바일 폰 번호를 입력하고, 그 연락처에 서프라이즈 P2P 이체를 원할 수 있다. Tereon은 해당 번호에 대한 레코드를 검색하고, 위와 같이 연락

처가 모바일 번호로 변경되었음을 발견한다(연락처가 Tereon 사용자인 경우). 새 서버 번호를 사용하는 사용자가 이전 서버(previous server)에 등록된 구 번호(old number)를 사용하곤 했다는 것을 정확한 서버에서 확인한다. 또한, Tereon은 특정 승인된 연락처가 구 크리덴셜(old credential)을 통해 트랜잭션을 시도할 때 디렉터리 서버에서 해당 사용자의 모바일 번호 또는 다른 Tereon 크리덴셜을 업데이트하도록 연락처가 자신의 계정을 설정하는 기능을 지원한다. 이 예에서, 숙모의 조카딸은 가족 구성원을 모두 업데이트하도록 계정을 설정했으므로, 다음에 이모가 연락처 목록에 액세스하면 그녀는 그녀의 조카딸의 새로운 모바일 번호를 볼 수 있다.

[0526] 도 16은 서버(202a), 서버(202b), 및 디렉토리 서비스(216)에 대한 예를 도시한다. 여기서, 구 사용자는 자신의 계정을 서버(202a)에서 서버 (202b)로 이전했다. 202a는 은행 A의 서버이고, 202b는 은행 B의 서버이다.

[0527] 구 사용자는 초기에 자산의 ID로 모바일 번호 1을 사용한다. 그의 계정을 이전한 후, 그는 계속해서 모바일 번호 1을 계속 사용한다. 사용자(218), 디렉토리 서비스(216), 및 서버 (202a 및 202b) 사이의 통신은 도 15에 도시되고 전술한 바와 같이 진행된다. 디렉토리 서비스의 엔트리들은 사용자(218)가 날짜-시간 1(date-time 1)에서 날짜-시간 3(date-time 3)까지 서버(202a)를 사용했고, 날짜-시간 2(date-time 2)부터 그가 서버(202b)를 사용했다는 것을 보여준다. 약간의 오버랩(중복)은 모든 미정인 결제(in-air payments)가 잡히는 것과, 사용자가 자신의 ID가 등록된 서버가 없는 시간 간격이 없다는 것을 보증하는 것이다(계정이 마이그레이션되는 서버가 해당 마이그레이션에 대한 모든 날짜-시간 및 ID 엔트리들을 제어하도록 보장함으로써 날짜-시간 엔트리들이 중복되는 것을 피할 수 있으며, 이것이 시스템 마이그레이션이 작동하는 방법이다.).

[0528] 어느 시점에서, 사용자(218)는 모바일 번호를 변경하기로 결정했다. 새로운 모바일 번호 2를 자신의 ID로 서버(202b)에 등록하고, 모바일 번호 1을 등록 해제한다. 서버(202b)는 디렉토리 서비스(216)에 변경을 통지한다. 이는 사용자가 날짜-시간 4에서 모바일 번호 2를 자신의 ID로 사용하기 시작했고, 모바일 번호 1이 날짜-시간 5에서 서버(202b)에 대해 ID로의 사용이 중지되었음을 보여준다.

[0529] 나중에 새로운 사용자는 서버(202a)에 계정을 생성하고, 날짜-시간 6에 자신의 ID로 모바일 번호 1을 등록한다. 새 사용자에게 구 사용자의 이전 모바일이 제공되었을 수도 있고, 해당 번호가 모바일 운영자에 의해 재사용을 위해 해제되었을 수도 있다. 서버(202a)는 (ID가 이용 가능하다는 것을 확인한 후에) ID를 등록했다는 것을 디렉토리 서비스(216)에 통지하고, 디렉토리 서비스는 이제 날짜-시간 6에서 모바일 번호 1이 서버(202a)에 등록되어 있음을 나타낸다.

[0530] 도 16에 도시 된 바와 같이, 구 사용자(old user)가 은행 A(202a)에 의해 발행된 카드를 사용하는 경우, 일단 사용자(218)가 자신의 계좌를 은행 B(202b)로 이전하면, 은행은 PAN과 같은 크리덴셜을 사용하여 새로운 카드를 사용자(218)에 발급한다. 사용자(218)는 카드를 수신하면 카드를 활성화시키고, 은행 B의 서버(202b)는 사용자의 원래 크리덴셜이 더 이상 사용되지 않는다는 것을 은행 A의 서버(202a)에 알린다. 은행 B는 새로운 크리덴셜을 Tereon 디렉토리 서비스(216)에 등록한다. 사용자(218)는 원래의 크리덴셜을 유지할 것을 요구할 수 있으며, 이 경우 은행 A가 요청에 동의하면 그렇게 하기 위해 그는 은행 A에 의해서 약간의 요금을 부과받을 수 있다. 따라서 Tereon은 카드 번호 또는 PAN 이동성을 지원한다.

[0531] 사용자는 장래에 은행 A가 처음 발행 한 카드 사용을 중단하고 해당 크리덴셜을 해제할 수 있다. 은행 B가 그것을 해제한 후 또는 사용자가 그의 계정을 은행 B로 이체한 후 6개월 동안, 은행 A는 해당 PAN 크리덴셜을 재사용 할 수 없다; 정확한 시간은 은행의 규제 당국이 허용하는 것에 달려 있다. 이 시간이 지나면 디렉토리 서비스(216)가 모바일 번호, PAN 또는 다른 크리덴셜을 포함하지 않기 때문에, 그것은 크리덴셜을 사용할 수 있다; 또한 그것은 크리덴셜이 등록된 날짜, 사용자 기준으로 만료되거나 사용자별로 해제된 날짜의 목록을 포함한다.

[0532] 계정 전환 방법은 시스템이 미정인 결제(in-the-air payments)를 캡처할 수 있도록 한다. 또한, 이전 거래에서 사용된 크리덴셜을 기반으로 이전 거래에서 이어지는 트랜잭션을 지시할 수 있는 매우 유연하고 강력한 방법을 제공한다. 이전 거래에 대한 환불은 실제 사례 중 하나이다. 원래 ID가 나중에 다시 사용된 경우에도 디렉토리 서비스(216)이 서버에 올바른 ID를 지불하도록 지시하므로 이전 ID에 대해 환불하는 머천트는 올바른 계정을 환불할 수 있다. EMV 및 현재 모바일 룩-업 기술은 숫자가 결코 재사용되지 않는다고 가정한다. 불행히도, 때로는 있다.

[0533] 도 16은 이를 도시한다. 날짜-시간 1과 날짜-시간 2 사이의 어떤 시간에 구 사용자가 모바일 번호 1을 ID로 갖는 장치를 사용하여 머천트로부터 아이템을 구매한다고 가정한다. 나중에, 아이템이 잘못되었음을 증명하고, 사용자는 환불을 원한다.

[0534] 사용자(218)가 환불을 위해 날짜-시간 1과 날짜-시간 2 사이에 머천트에게 가는 경우, Tereon 시스템은 머천트 시스템이 시스템(202a) 상의 사용자 계정으로 환불 결제하도록 지시할 것이다(사용자가 아직 계정을 폐쇄하지 않았기 때문에).

[0535] 사용자(218)가 환불을 위해 날짜-시간 2와 날짜-시간 4 사이에 머천트에게 가는 경우, 아이템에 대한 결제가 원래 서버(202a)로부터 왔음에도 불구하고 Tereon 시스템은 머천트 시스템이 시스템(202b) 상의 사용자 계정으로 환불 결제하도록 지시할 것이다.

[0536] 계정 전환 방법은 사용자의 새로운 ID로 고려할 것이다. 사용자(218)가 환불을 위해 날짜-시간 4 이후에 머천트에게 가고, 그의 모바일 번호 2를 그의 ID로 사용하는 경우, 비록 아이템에 대한 결제가 원래 서버(202a)로부터 왔고 사용자가 원래 자신의 결제 ID로 모바일 번호 1을 사용했음에도 불구하고 Tereon 시스템은 머천트 시스템이 시스템(202b) 상의 사용자 계정으로 환불 결제하도록 지시할 것이다.

[0537] PAN, 전자메일 주소, 및 기타 재사용 가능한 크리덴셜들의 레코드들도 동일하게 유지된다(생체 크리덴셜들(Biometric credentials)은 명백한 이유로 재사용할 수 없다).

[0538] 시스템은 크리덴셜을 임의의 수준으로 세분화할 수 있다. 이 결제 방법의 한 가지 예는 통화(currency) 또는 통화 코드(currency code)를 포함한다. 여기서, 사용자는 동일한 통화 또는 별도의 서버에서 서로 다른 통화에 대해 서로 다른 ID를 사용할 수 있다.

[0539] 도 17은 서버(202b), 서버(202c), 및 디렉토리 서비스(216)에 대한 예를 도시한다. 도면에서, 사용자(218)는 도 15에 도시된 바와 같이 관리되는 서버 간 통신을 통해 도 16에 도시된 것과 유사한 방식으로 서버(202b)로부터 서버(202c)로 자신의 계정을 이미 이동시켰다(migrate).

[0540] 사용자(218)는 초기에 자산의 ID로 모바일 번호 1을 사용한다. 그의 계정을 이전한 후, 그는 통화 1(currency 1) 및 통화 2(currency 2) 내 트랜잭션에 대해 한 동안 모바일 번호 1을 계속 사용한다. 디렉토리 서비스(216)의 엔트리들은 사용자(218)가 날짜-시간 1(date-time 1)에서 날짜-시간 3(date-time 3)까지 서버(202b)를 사용했고, 날짜-시간 2(date-time 2)부터 그가 서버(202c)를 사용했다는 것을 보여준다. 약간의 오버랩(중복)은 모든 미정인 결제(in-air payments)가 잡히는 것과, 사용자가 자신의 ID가 등록된 서버가 없는 시간 간격이 없다는 것을 보증하는 것이다.

[0541] 어느 시점에서, 사용자(218)는 통화 2(currency 2) 내 트랜잭션에 대해 새로운 모바일을 사용하기로 결정했다. 그는 통화 2(currency 2) 내 트랜잭션에 대해 자신의 새로운 모바일 번호 2를 자신의 ID로 서버(202b)에 등록했다. 서버(202b)는 디렉토리 서비스(216)에 변경을 통지했다. 이는 사용자가 날짜-시간 4에서 통화 2(currency 2) 내 모든 트랜잭션에 대해 모바일 번호 2를 자신의 ID로 사용하기 시작했고, 모바일 번호 1이 날짜-시간 5까지의 모든 거래에 대해 ID로의 사용이 중지되었음을 보여준다.

[0542] 도 17a는 서버(202b), 서버(202c), 및 디렉토리 서비스(216)에 대한 다른 예를 도시한다. 도면에서, 사용자(218)는 도 15에 도시된 바와 같이 관리되는 서버 간 통신을 통해 도 16에 도시된 것과 유사한 방식으로 서버(202b)로부터 서버(202c)로 자신의 통화 1 계정(currency 1 account)을 이미 이동시켰다(migrate).

[0543] 계정 이동 후에, 사용자는 모바일 번호 1을 사용하는 시간 동안, 통화 1(currency 1)과 통화 2(currency 2) 사이의 트랜잭션 시간 동안 계속 했다. 디렉토리 서비스(216) 내 엔트리들은 사용자(218)가 날짜-시간 1(date-time 1)에서 날짜-시간 3(date-time 3)까지 서버(202b)를 사용했고, 날짜-시간 2(date-time 2)부터 그가 통화 1(currency 1) 내 트랜잭션에 대해 모바일 번호 1을 자신의 ID로 서버(202c)에 사용했음을 보여준다. 또한, 디렉토리 서비스 엔트리들은 사용자가 통화 2(currency 2) 내 트랜잭션에 대해 모바일 번호 1을 자신의 ID로 서버(202b)에 계속 사용했음을 보여준다.

[0544] 어느 시점에서, 사용자(218)는 통화 2(currency 2) 내 트랜잭션에 대해 새로운 모바일을 사용하기로 결정했다. 그는 통화 2(currency 2) 내 트랜잭션에 대해 자신의 새로운 모바일 번호 2를 자신의 ID로 서버(202b)에 등록했다. 서버(202b)는 디렉토리 서비스(216)에 변경을 통지했다. 이는 사용자가 날짜-시간 4에서 통화 2(currency 2) 내 모든 트랜잭션에 대해 모바일 번호 2를 자신의 ID로 사용하기 시작했고, 모바일 번호 1이 날짜-시간 5까지의 모든 거래에 대해 ID로의 사용이 중지되었음을 보여준다.

[0545] 날짜-시간 4 이전에, 사용자(218)는 그의 모바일 번호 1을 그의 모든 트랜잭션들에 대해 ID로 사용했다. 디렉토리 서비스(216)은 단순하게 트랜잭션들이 통화 2(currency 2) 내 있으면 트랜잭션들을 서버(202b)로 향하게 하고, 트랜잭션들이 통화 1(currency 1) 내에 있으면 트랜잭션들을 서버(202c)로 향하게 했다. 트랜잭션이 전

달되는 서버를 제어하는 크리덴셜들의 완전한 집합이므로, 사용자가 두 서버에 같은 ID를 등록했다는 사실은 관련이 없다. 날짜-시간 2 이후에 처음으로 통화 1(currency 1)에서 사용자와 거래하는 머천트의 시스템은 사용자가 이전에 해당 통화 내 트랜잭션들에 대해 서버(202b)를 사용했다는 것을 결코 알지 못한다. 마찬가지로, 머천트의 시스템이 통화 2로 사용자와 트랜잭션을 시작하지 않는 한, 머천트의 시스템은 사용자가 통화 2(currency 2) 내 트랜잭션들에 대해 같은 ID를 서버(202b)에 사용했는지를 알 지 못한다.

[0546] Tereon은 단순히 사용자(218)를 하나의 네트워크에서 다른 네트워크로 전환하는 것 이상의 역할을 한다. 이미 언급했듯이, 사용자 전환의 일반적인 방법은 미정인 결제(in-the-air payments)를 처리하지 못한다. 창안자들에 의해 주장된 바와 같이, 현재 이용 가능한 가장 진보된 계정 전환 시스템은 사용자가 스스로를 기다리기 전에 결제금을 받기 위해 18개월의 수동 프로세스를 필요로 한다. 18개월 동안, 은행과 사용자는 기존 계정의 모든 기존 결제 인스트럭션들을 새 계정으로 이전하도록 노력해야 한다. Tereon은 이 요구사항을 완전히 없애 버린다.

[0547] 현재, 은행은 결제 크리덴셜들을 재사용할 수 없다. Tereon의 계정 전환 메커니즘은 이러한 제한을 제거하고, 규제 기관이 허용하기를 원하는 경우 특정 기간이 지난 후에 은행이 PAN 및 계정 번호를 재발행할 수 있다.

[0548] 이 방법을 계정 전환 기능이라고 부르지만, 사실상 기본 계정 전환 이상의 많은 애플리케이션들이 있다. 예를 들어, 그것은 은행 코어 시스템이 실패한 경우 백업 서비스 제공자에 장애 극복(failover)를 제공하여, 정보 손실없이 하나의 데이터 형식에서 다른 데이터 형식으로 변환함으로써 한 시스템에서 다른 시스템으로 데이터를 마이그레이션할 수 있는 방법을 제공한다.

[0549] 또 다른 예는 모바일 시스템에서 번호 이동성(number portability)을 간소화하는 것이다. 현재, 사용자가 한 공급자에서 다른 공급자로 자신의 모바일 번호를 전환한 경우, 제1 공급자(첫번째 공급자)는 새 공급자에 대한 모든 호출(call)을 다시 라우팅해야 한다. 사용자가 제3 공급자(세번째 공급자)로 전환한 경우, 제1 공급자는 제2 공급자(두번째 공급자)에게 호출을 라우팅해야 하고, 제2 공급자는 제3 공급자에게 호출을 라우팅해야 한다. 이것은 극도로 비효율적이며 비용이 많이 들지만, 운영자는 번호 이동성을 지원해야 한다. Tereon은 호출을 여러 번 라우팅해야 할 필요가 없다.

[0550] 운영자들이 번호 이동성을 지원하기 위해 Tereon을 사용한다면, 그들은 다중 홉을 지원할 필요가 없다. 사용자가 자신의 번호를 제1 운영자(첫번째 운영자)에서 제2 운영자(두번째 운영자)로 이전하기(port)로 결정하면, 제2 운영자는 단순히 디렉토리 서버에 현재 해당 모바일 번호를 지원하기로 했음을 알리기만 하면 된다. 제1 운영자는 해당 번호에 대한 호출을 디렉토리 서버로 돌리면 제2 운영자에게 호출이 라우팅된다. 사용자가 자신의 번호를 다시 이전할(port) 때마다, 새로운 운영자는 디렉토리 서버에 변경 사항을 알리고, 디렉토리 서버는 해당 번호를 서비스하는 운영자에게 호출을 라우팅한다(사용자들이 전 세계적으로 고유한 IBAN과 같은 은행 계정을 보유한 경우, Tereon은 모바일 번호 이동성을 지원하는 것과 동일한 방식으로 은행 계정 이동성을 지원한다).

[0551] 유사한 예는 물리적 기계, 논리 기계, 가상 기계, 컨테이너, 또는 실행 가능한 코드를 포함하는 다른 일반적으로 사용되는 메커니즘 등의 단순 마이그레이션은 충분하지 않은 Tereon 시스템을 업그레이드하기 위해, 운영자가 한 서버에서 다른 서버로 IoT 서비스와 장치를 마이그레이션하는 예이다.

[0552] 또 다른 예는 시스템이 마이그레이션 툴로 작동하는 것이다. 예를 들어, 이는 운영자가 Tereon 시스템의 한 버전에서 업그레이드된 버전으로 장치가 등록된 계정과 함께 서비스를 마이그레이션하려는 경우이다. 운영자는 장치 등록, 계정, 및 시스템 컨피규레이션(system configurations)을 새로운 서버로 전송하도록 이전 서버(old server)를 설정하기 만하면, 시스템은 전송을 수행한다. 각 계정은 데이터 및 감사 로그(audit logs)와 함께 전송되며, 전송 진행에 따라 서버는 디렉토리 서비스 216을 업데이트한다. 이제, 결제 장치, 교통 센서, IoT 장치 등인 서버와 통신하기를 원하는 현장의 장치인 경우, 디렉토리 서비스(216)는 계좌가 이전되기 전후에 서버에 접속했는지 여부에 따라 이전 또는 새로운 서버로 간단히 리다이렉트할 것이다.

[0553] 위의 예는 Tereon이 크리덴셜 이동성을 용이하게 하고, 애드 혹 다면적인 크리덴셜들(ad hoc multifaceted credentials)을 지원하는 방법을 보여준다. 이는 광범위한 애플리케이션을 보유하고 있으며, Tereon을 네트워크에서 크리덴셜들을 관리해야 하는 거의 모든 네트워크 분야에 적용한다.

[0554] 확장 가능한 프레임워크(Extensible Framework)

[0555] 기존 트랜잭션 처리 시스템들의 워크플로우들은 본질적으로 너무 정적이다. 일단 구현되면, 그것들은 변경하기

가 매우 어렵고, 시스템들이 지원하는 서비스들 또는 동작들(operations)은 유연성이 없다.

[0556] 지금까지, 결제 제공자가 서비스를 시작한 경우, 해당 서비스에 대한 결제 패턴은 정적이었다. 제공자는 대체 또는 수정된 서비스를 시작하고, 해당 서비스를 지원하기 위해 새로운 카드 또는 애플리케이션을 발급함으로써 해당 서비스를 수정만 할 수 있다. 이것이 EMV의 심각한 취약점에 대한 보편적인 지식임에도 불구하고, 존재하는 EMV 카드를 리콜하고 EMV 결제 인프라를 재프로그래밍하고 실행하고 새로운 카드들을 발급한다는 의미에서, 시스템을 수정하는 것이 불가능한 이유 중 하나이다. 이는 협력할 수천 명의 발급자(issuer)와 인수자(acquirer)가 필요하다.

[0557] Tereon은 SDASF를 사용하여 백-엔드(back-end)에 모든 기능을 넣고, 백-엔드(back-end)는 프로세스를 통해 실시간으로 머천트 장치를 안내할 수 있다. 이는 서비스 제공자가 개별 사용자만큼 세부적인 새 서비스를 만들도록 할 수 있다.

[0558] 확장 가능한 프레임워크는 Tereon 시스템 내에 위치하는 프레임워크이고, Tereon 시스템을 재구성할 필요 없이 새로운 서비스를 추가할 수 있게 해준다. 확장 가능한 프레임워크는 Tereon 시스템에 여러 가지 이점을 제공하기 위해 디렉토리 서비스 216과 함께 작동한다.

[0559] *유연한 메시지 구조(Flexible message structure)*

[0560] 확장 가능한 프레임워크는 유연한 메시지 구조- 가변 길이 필드가 있는 모든 데이터 또는 레코드 유형이 제공될 수 있어서, Tereon 시스템이 레거시 또는 호환되지 않는 시스템에서 작동하도록 필드의 길이를 수정할 수 있음-에 의해 부분적으로 제공된다.

[0561] 확장 가능한 프레임워크는 프로세스들의 표준 순서를 변경함으로써 통신 인프라스트럭처에 추가적인 보안 레이어의 추가를 허용한다. 많은 산업 분야에서, 결제는 단지 예일뿐이며, 통신은 고정된 메시지 구조를 사용한다. 통신이 암화화된 경우에도, 이는 범죄자가 악용할 수 있는 약점을 발생시킨다. 구조화된 메시지들은 깊이 있는 공격에 취약하다. 조직 및 다른 사람들은 HMAC(hash message authentication code)를 사용하여 메시지 무결성을 보호할 수 있지만, HMAC는 메시지가 유치해야 하는 절대적인 비밀을 보관하지 않는다.

[0562] 확장 가능한 프레임워크는 모든 트랜잭션 처리 시스템에 대해 정적 시스템들의 문제를 설계한다. 그것은 기존 시스템 및 서비스와 함께 작동할 수 있는 유연성을 제공하며, 제공자가 기존 서비스를 업데이트하고, 인프라스트럭처를 다시 가동하거나 카드와 같은 새로운 종단 장치(end-point device)을 발행할 필요 없이 새로운 서비스를 구축하도록 한다. 이에 대해서는 아래에서 설명한다.

[0563] *난독화(Obfuscation)*

[0564] 구조화된 메시지 형식을 가진 시스템이 직면하는 이론적 위험 중 하나는 메시지 형식의 반복적 사용이 해커가 무차별 공격에 사용할 수 있는 충분한 자료를 제공한다는 것이다. 이는 임의의 형태의 랜덤 시딩(random seeding)으로 암호화 알고리즘을 올바르게 구현하지 않은 시스템에 해당한다.

[0565] 확장 가능한 프레임워크는 운영자와 사용자가 장치와 서버 간에 구조화된 메지지를 전송할 필요성을 없애준다. 대신에, 메시지는 난독화될 수 있다.

[0566] Tereon 내 각 트랜잭션 통신은 해당 필드의 레이블과 함께 두 개 이상의 필드를 포함한다. 모든 통신에 대해 필드의 고정된 순서에 따르는 대신에, 순서는 랜덤 방식(random manner)으로 변경될 수 있다. 각 필드는 항상 식별 태그를 수반하므로, 통신의 각 끝에 있는 장치들이 먼저 해독하고 필드들을 처리하기 전에 필드들을 정리(order, 또는 정렬)해야 한다.

[0567] 예를들어, JSON(JavaScript Object Notation) 문서(documentation)에 의해 제공되는 사례에서의 발췌를 이용하면(물론 다른 형식도 시스템 내에 있을 수 있고, 사용됨), 다음의 세가지 표현이 동일하다:

[0568] • {"version": 1, "firstName": "John", "lastName": "Smith", "isAlive": true, "age": 25}

[0569] • {"version": 1, "firstName": "John", "isAlive": true, "lastName": "Smith", "age": 25}

[0570] • {"age": 25, "firstName": "John", "isAlive": true, "lastName": "Smith", "version": 1}

[0571] 공격자는 자신이 가지고 있는 사이퍼 텍스트(cypher texts)가 알려진 순서와 같은 정보를 포함하고 있는지 알 수 없다. 난독화(obfuscation)의 정확한 모드는 사용된 형식 및 사용된 직렬화 프로토콜에 따라 다르지만, 원

칙은 동일하게 유지된다.

[0572] 난독화 모드는 추가적인 이점을 갖는다. 미리 정의된 통신들의 컨텐츠들은 통신 프로토콜을 위반하지 않고 확장될 수 있다. 장치가 처리할 수 없는 필드들을 수신하면, 장치는 해당 필드들과 해당 값을 단순히 버릴 것이다. 따라서, 하나 이상의 랜덤 필드 및 값의 쌍(pair)은 시스템이 버리는 것에 포함될 수 있고, 추가적인 불확도(additional uncertainty)가 통신에 추가된다.

[0573] 따라서 다음 세가지 통신들은 동일하다:

[0574] • {"version": 1, "firstName": "John", "nonce": 5780534, "lastName": "Smith", "isAlive": true, "age": 25}

[0575] • {"whoknows": "698gtHGF", "version": 1, "firstName": "John", "isAlive": true, "lastName": "Smith", "age": 25}

[0576] • {"age": 25, "firstName": "John", "isAlive": true, "lastName": "Smith", "whatis this": "Jor90%hr," "version": 1}

[0577] 위의 통신들 각각에서, 장치들은 알 수 없는 필드 및 값의 쌍(pair)을 버린다.

[0578] 피드 이름들은 각 통신에 대해 임의의 방식으로 사례들을 혼합함으로써 혼란스럽게 만들 수 있다. 장치들은 이 필드들을 정규 형식(canonical form)으로 처리한다.

[0579] 따라서 다음 세가지 통신들은 동일하다:

[0580] • {"veRsioN": 1, "firstName": "John", "nOnce": 5780534, "laStnAMe": "Smith", "isAlive": true, "Age": 25}

[0581] • {"whoknows": "698gtHGF", "vErsion": 1, "fiRStname": "John", "iSaLive": true, "lastName": "Smith", "age": 25}

[0582] • {"aGE": 25, "firstname": "John", "isAlive": true, "lasTName": "Smith", "whatis this": "Jor90%hr," "versIOn": 1}

[0583] • {"veRsioN": 1, "firstName": "John", "nOnce": 5780534, "laStnAMe": "Smith", "isAlive": true, "Age": 25}

[0584] 버전 1만을 이해하는 장치는 추가 필드들을 포함할 수 있는 버전 2 메시지가 전송되면 메시지를 거부하거나, 하위 호환성(backwards compatibility)이 보장되면 이해하는 필드들을 처리하고 나머지는 버릴 수 있다. 이는 어느 버전이 일부 필드와 하위 호환되는지를 나타내는 필드를 제공함으로써 더욱 향상될 수 있다.

[0585] 이는 공격에 대한 취약성을 상세히 제거한다. 메시지의 구조는 유지될 수도 있지만, 가변 길이 필드들(variable length fields)이 있다. 다시 이것은 유사한 결과를 달성한다. 또한, HMAC를 사용함으로써, 메시지의 무결성(integrity)과 기밀성(secrecy)이 모두 보호된다. 최종 조직의 코어 시스템들이 구조화된 형식의 메시지들을 필요로 한 경우, Tereon은 일단 메시지가 서버에 도달하면 메시지를 재구성하고, 조직의 코어 시스템들에 의해 요구되는 형식으로 메시지를 재포맷한다(reformat). 확장 가능한 프레임워크는 레거시 시스템의 보안 문제를 극복되게 할 수 있지만, 여전히 이러한 시스템들로 작동한다.

[0586] 확장 가능한 프레임워크는 위에서 언급한 보안 및 유연성 수준과 완전히 동일한 데이터 또는 레코드 유형을 지원한다.

[0587] *추상화된 워크플로우 컴퍼넌트들(Abstracted workflow components)*

[0588] 기존 솔루션들에서, 결제 프로세스는 소프트웨어로 정의되고, 구현되고, 테스트되고, 그러고 나서 릴리스된다(released). 그 결제 트랜잭션 구조는 고정되며, 장치들, 단말기들, 및 서버들을 리콜 및 교체하거나 재프로그램하기 위한 상당한 노력 없이 변경될 수 없다.

[0589] Tereon은 이것을 하지 않는다. 대신에, 이것은 각각이 연결된 컴퍼넌트와 상호작용하는 개별 컴퍼넌트들로부터 결제 프로세스를 구성한다. 이러한 컴퍼넌트들은 본질적으로 프로세스의 워크플로우를 배치한다(lay out). 각

컴퍼넌트는 업데이트될 수 있고, 결제 프로세스 자체에 영향을 주이 않고 추가된 기능을 가질 수 있다. 이는 장치로부터 프로세스 컴퍼넌트들을 추상화하므로, 한 번 정의된 트랜잭션이 카드들 및 카드 단말기들, 모바일 폰들, 또는 웹 포털들과 같은 복수의 장치들에 적용될 수 있다.

[0590] 각 컴퍼넌트는 수신한 인스트럭션의 결과에 따라 인스트럭션들 및 정보를 다음 컴퍼넌트로 전달한다. 인스트럭션들은 트랜잭셔널(transactional)이거나 다음 컴퍼넌트가 작동하는 방법과 같은 제어들을 포함할 수 있다(예를 들어, 선택 사항인 경우 PIN 요청, 선택의 집합을 제공, 특정 메시지 표시, 및 예상되거나 허용되는 응답들).

[0591] 이는 기존 엔드 포인트들을 재프로그래밍하거나 교체할 필요 없이 기존 결제 서비스들을 변경하고 새로운 서비스를 구축하는 기능을 제공한다. 현재, 결제 서비스 제공자가 결제 시스템을 구현하면, 결제 서비스 제공자는 엔드 포인트를 교체하지 않고 시스템을 쉽게 변경할 수 없다. 기존 시스템은 기본적으로 정적이다. 이는 그것들을 동적 시스템으로 교체한다.

[0592] 확장 가능한 프레임워크는 운영자가 이러한 컴퍼넌트들을 사용하여 특정 트랜잭션에 대한 워크플로우를 계획할 수 있도록 한다. 그것은 의사 결정 트리 등을 포함하는 워크플로우를 구현한다. 운영자는 기존 컴퍼넌트들을 재배열하거나, 새로운 기능들을 제공하는 새로운 컴퍼넌트들을 추가하거나, 컴퍼넌트들을 제거함으로써 기존 워크플로우를 수정한다. 기존 시스템에서 이를 수행하기 위해, 서버들 및 단말기들은 재프로그램될 필요가 있으며, 카드드들 자체는 교체될 필요가 있다.

[0593] 이것의 예들은 도 18 내지 도 20에 도시되어 있다. 컴퍼너트들 자체는 각 컴퍼넌트가 하는 것을 쉽게 시각화할 수 있도록 터미널 스크린에 의해 블록으로 표시된다. 그러나, 컴퍼넌트들은 모바일 트랜잭션들(mobile transactions), 웹 포털 트랜잭션들(web portal transactions), 및 카드 터미널 트랜잭션(card terminal transactions)에 동일하게 적용된다. 기존 워크플로우를 변경하기 위해, 컴퍼너트들의 순서와 연결은 간단히 변경된다. 새로운 워크플로우를 만들기 위해, 필요한 컴퍼넌트들은 원하는 순서로 함께 간단히 연결된다.

[0594] 일반 결제 프로세스는 비접촉식(contact-less), 연락처(contact), 및 모바일 결제에 대해 별도의 결제 프로세스를 생성한다. 컴퍼넌트(1804)는 도 18에 도시된 바와 같은 '정시에 완전한 트랜잭션(complete transaction in time)' 컴퍼넌트(1802) 직후에 체인의 좌측에 일반적으로 나타난다.

[0595] 그러나, 도 19에 도시된 바와 같이, 이 컴퍼넌트를 우측을 따라 더 이동시킴으로써, 2개의 다른 결정 컴퍼넌트들(1902 및 1904)를 체인에 삽입하면, 운영자는 하나의 단일 결제 프로세스에서 접촉, 비접촉식, 및 모바일 지불을 관리할 수 있는 단일 결제 프로세스를 생성할 수 있다.

[0596] 운영자는 더 나아갈 수 있다. 아마도 시스템이 고객을 식별한 후 프로세스에 특별한 계절 제안(seasonal offer)을 추가하려고 한다. 도 20에 도시된 바와 같이, 언제든지 컴퍼넌트(1804)를 더 오른쪽으로 이동시키고, 머천트(merchant)가 금액 및 PIN을 입력해야 하기 전에 자동적으로 고객에게 제안을 제공하는 새로운 컴퍼넌트(2002)를 원래 위치에 삽입할 수 있다. 운영자는 예를 들어, 크리스마스로 이어지는 24일 이내에 작동하도록 해당 컴퍼넌트를 구성하고, 이후 새해를 맞이하는 기간 동안 다른 컴퍼넌트를 제공할 수 있다. 운영자가 리콜, 재프로그램, 및 장치를 요구하지 않고도, 이것은 크리스마스 및 새해 시즌의 결제 프로세스를 동적으로 변경한다. 컴퍼넌트들은 고객에게 제안을 표시하도록 모바일 폰 또는 카드 단말기인 디스플레이 장치에게 지시할 뿐이다. 운영자는 PIN 요구사항을 비활성화하는 컴퍼넌트(1804)를 구성함으로써 PIN 요구사항을 쉽게 비활성화할 수 있다. 마찬가지로, 컴퍼넌트가 PIN을 요구하는 기능을 가지고 있지 않은 경우, 운영자는 기능을 포함하도록 해당 컴퍼넌트를 업데이트할 수 있다.

[0597] 운영자는 더 나아가 고객이 원할 경우 다양한 제안 중에서 선택할 수 있도록 전체 의사 결정 트리를 구축할 수 있다. 제안 시즌(offer season)이 끝나면, 운영자는 새 컴퍼넌트를 간단히 제거하고, 프로세스는 원래 구조를 다시 시작한다.

[0598] 중요한 것은 운영자가 언제든지 프로세스를 변경하기 위해 장치를 리콜해야 한다는 것이다. 백 엔드(back end)에서 프로세스를 재구성하기만 하면 선택한 시간과 날짜에 변경 사항을 구현한다.

[0599] Tereon 서버들의 내부 관리 및 운영을 제공하는 프레임워크는 정확히 같은 방식으로 구성할 수 있다. 여기서, 프레임워크 컴퍼넌트들은 사용자와 관리자가 액세스할 수 있는 방법 및 어떤 정보, 수행할 수 있는 작업을 관리하는 액세스 컨텍스트와 상호작용한다.

[0600] *동적 서비스들(Dynamic services)*

[0601] 확장 가능한 프레임워크는 조직이 새로운 서비스를 신속하게 만들고 구현하도록 하게 한다. 운영자는 필요한

블록들을 함께 연결하고, 관련 메시지들을 정의함으로써 이러한 서비스들을 간단히 정의한다. 서비스 코드를 작성하기 위해 프로그래머를 고용하는 대신에(또는 고용할 필요 없이), 프레임워크는 마케팅 및 IT 부서가 워크플로우를 정의하는 정의 파일(definition file)을 작성하거나, '워크플로우를 그리기" 위한 그래픽 시스템(graphical system)을 사용하거나, 또는 프로세스를 정의하는 다른 워크플로우에 의해서 서비스들을 구현할 수 있게 한다. 워크플로우를 확인하면, 운영자는 정의된 단계들 또는 블록들을 함께 연결함으로써 워크플로우를 간단히 구현하고, Tereon은 모든 해당 사용자들(qualifying users)이 서비스를 이용할 수 있게 한다.

[0602] 예를 들어, 운영자는 PIN을 요청할 수 있는 후속 블록(subsequent block)으로 어떤 가치의 지불을 받기위해 블록을 사용할 필요가 있다. 그러나, 운영자가 액세스 제어 시스템을 제공하려는 경우, 동일 운영자는 다른 세트의 룸에 액세스할 수 있는 PIN을 요청하기 위한 블록을 사용하는 동안 한 세트의 룸에 PIN 없이 액세스를 허용하는 블록을 생성한다.

[0603] 이는, 기존 시스템과 달리, 조직이 트랜잭션 처리 시스템을 시작한 후에도 사용자에게 발행된 장치들을 교체할 필요 없이, 시스템이 조직들이 새로운 서비스들을 설계 및 구현하거나, 기존 서비스들을 수정 또는 제거하도록 하는 것을 의미한다. 장치가 정의된 단계들을 이해하고 작동할 수 있다면, 해당 장치는 조직이 정의한 임의의 서비스를 해당 단계들을 사용하여 제공할 수 있다. 조직이 서비스를 정의하면, 시스템은 해당 서비스를 대상 사용자 또는 사용자들에게 즉시 사용할 수 있도록 한다.

[0604] *추상화된 장치들(Abstracted devices)*

[0605] 확장 가능한 프레임워크는 추상화의 원리를 받아들이고, 장치들 자체를 추상화한다. 프레임워크는 장치의 각 클래스에 대한 프로세스 컴퍼넌트들- 해당 장치의 기능들과 관련됨 -을 정의한다. 프로세스 컴퍼넌트들은 해당 기능 컴퍼넌트들과 상호작용한다. 사용 가능한 기능에 따라, 프로세스 컴퍼넌트들은 무엇을 출력하고, 무엇을 입력할 거와 같은 작업을 수행하도록 기능 컴퍼넌트들에 지시한다.

[0606] *입도(또는 세분화, Granularity)*

[0607] Tereon은 각 장치, 사용자, 및 계정을 개별적으로 식별하고, 사용자가 장치를 사용하여 서비스에 액세스하는 컨텍스트를 액세스할 수 있다. 따라서, 운영자는 개별 사용자가 서비스에 액세스하는 컨텍스트를 기반으로 동작(action)을 트리거하기 위해 컴퍼넌트들 및 해당 컴퍼넌트들 내 옵션들을 구성할 수 있다. Tereon은 운영자가 효과적으로 각 사용자, 각 사용자의 장치, 및 사용자가 해당 장치를 사용하여 서비스에 액세스하는 컨텍스트에 맞게 서비스를 조정할 수 있도록 한다.

[0608] 예를 들어, 한 사용자는 트랜잭션에서 세 가지 제안 중에서 하나를 볼 수 있고, 다른 사용자는 자동으로 받은 하나의 제안만 볼 수 있으며, 세번째 사용자는 제안을 전혀 볼 수 없을 수 있다.

[0609] 프로세스가 레코드들(예를 들어, 환자 레코드들)에 액세스하는 것과 관련이 있는 경우 사용자는 자신의 레코드에 액세스하고, 사용자가 의료 시설 또는 홈 도메인의 레코드에 액세스 하면 사용자는 액세스 권한을 관리한다. 그러나, 사용자(또는 다른 사람)가 해당 도메인 외부의 레코드들에 액세스하는 경우, 사용자는 해당 레코드들의 하위 집합만 보거나 (해당 서비스에 대한 컨텍스트 설정에 따라) 해당 레코드들에 액세스할 수 없다.

[0610] 사용자가 카드 단말기를 사용하여 서비스에 액세스하는 경우, 컴퍼넌트들은 카드 단말기에 관련 정보를 표시하도록 지시한다. 사용자가 모바일 폰 또는 다른 화면 장치를 사용하여 동일한 서비스에 액세스하는 경우, 컴퍼넌트들은 스크린에 관련 정보를 표시하도록 지시한다. 이러한 방식으로, 확장 가능한 프레임워크의 추상화 레이어는 장치에 독립적이다. 사용자 시스템 상호작용을 제어하기 위해 적합한 디스플레이 및 액세스 포인트를 사용할 수 있다.

[0611] 제공되는 서비스에도 동일하게 적용된다. 각 사용자의 계정에는 서비스들의 제공자 기본 레벨(default level)을 갖는다. 운영자가 새로운 서비스를 추가하거나 하나 이상의 사용자에 대한 기존 서비스를 수정하는 경우, 해당 사용자의 계정에는 해당 서비스가 존재한다. 서비스의 핵심은 제공자 태그(providers tag), 사용자의 계정 번호(user's account number), 사용자의 장치 등록 태그(user's device registration tag)의 조합일 것이다. 이는 해당 사용자 대한 서비스 정의 및 규칙에 덴드리틱 경로(dendritic path)를 생성한다.

[0612] 예를 들어, 발신자는 양방향 또는 자동 이체(interactive or automatic transfer)를 허용하는 규칙을 설정한 모바일 폰을 사용할 수 있다. 수신자는 자동 이체를 허용하도록 장치를 설정했을 수 있다. 이 경우, 발신자의 장치는 자동 이체를 하는 단계들을 수행한다. 서비스 태그는 이체가 양방향인지 여부에 대한 어떠한 정보를 포함하지 않는다; 이는 발신자와 수신자의 서버에 저장된 서비스에 대한 정보로 남는다.

[0613] 수신자가 양방향 또는 자동 이체를 허용하도록 장치를 설정한 경우, 발신자의 장치는 발신자에게 사용할 모드를 요청한다. 수신자가 특정 시간 사이에 자동 이체를 허용하도록 장치를 설정하고 다른 시간에 양방향 이체를 허용하도록 장치를 설정했을 수 있다. 여기서, 수신자의 Tereon 서버는 수신자의 시간에 따라 보낸 사람의 서버에 사용할 이체의 모드를 알린다.

[0614] 발신자 또는 수신자의 장치가 양방향 이체만 수락하면, 그 다음에 수신자 및 발신자가 동시에 온라인 상태인 경우, 그들은 이체를 실행하는 단계들을 수행한다. 수신자가 카드만 가지고 있는 경우, 그 다음에 수신자는 거래의 그의 측면(his side)을 수행하기 위해 머천트의 단말기로 가야한다. 수신자가 오프-라인인 경우, 그 다음에 발신자는 그의 단계들을 수행하지만, 수신자는 Tereon이 이체를 완료하기 전에 이체를 수락하고 PIN을 입력하는 등과 같은 트랜잭션 내 단계들을 수행해야 한다. 그때까지, Tereon은 비 Tereon 사용자(non-Tereon user)에게 이체를 다루는 유사한 방식으로 에스크로 시설(escrow facility)에서 이체를 보류한다.

[0615] *동적 인터페이스(Dynamic interfaces)*

[0616] 확장 가능한 프레임워크는 컨텍스트 의존적 서비스(예를 들어, 제안, 이벤트에서 자신의 자리를 찾도록 사용자를 돕는 거, 머천트 특정 프로세스 등)를 유도한다. 이는 조직이 사용자가 Tereon과 상호작용할 때 각 사용자가 가질 서비스와 경험, 컨텍스트에 따라 서비스가 가능한 정도, 어떤 버튼이 나타나는지, 어떤 옵션이 가능한지 등을 사용자 지정할 수 있도록 한다.

[0617] 각 사용자 및 각 머천트가 상호작용할 수 있는 서비스의 수는 개별 사용자가 액세스할 수 있는 서비스와 머천트가 제공할 수 있는 서비스 간의 오버랩(overlap)에 전적으로 달려 있다.

[0618] 예를 들어, 머천트가 결제, 예금(deposit), 및 인출(withdrawl)을 제공할 수 있는 곳에서, 사용자가 해당 머천트를 방문한 경우, 해당 사용자는 머천트의 결제에만 액세스할 수 있고, 그 다음에 사용자와 머천트는 결제와 관련된 기능, 즉 결제와 환불만을 볼 수 있다. 사용자가 동일 머천트를 방문한 경우, 해당 사용자는 결제, 예금, 및 인출에 액세스할 수 있고, 그 다음에 해당 사용자는 모든 기능을 볼 수 있다. 해당 머천트가 더 이상 예금 및 인출을 지원할 만큼 충분한 자금을 보유하지 않는 경우, 그 다음에 풀-서비스 사용자가 해당 머천트를 방문한 때, 사용자는 자신의 장치 또는 머천트의 단말기에서 결제 기능만 볼 수 있다. 해당 머천트는 머천트까지 예금 또는 인출을 제안하는 머천트들에 대한 검색에서 더 이상 나타나지 않을 것이다. 사용자가 일부 머천트의 특정 서비스에 액세스할 수 없지만 다른 머천트의 서비스에 액세스할 수는 있다. 프레임워크는 이러한 경우들을 다룬다.

[0619] 동적 인터페이스는 다면적인 크리덴셜의 사용을 보완하고, 장치 및 관련 어플리케이션들이 언급한 것처럼 '사이킥 종이(psychic paper)'와 유사한 것으로 할 수 있게 한다. 이 경우, 장치는 이용 가능한 서비스들만 제공하고, 사용자가 등록될 수 있는 복수의 서비스들에 관계없이 인터페이스는 그러한 서비스들에 맞추어진다. 그것은 한 서비스에 대한 결제 장치, 다른 서비스에 대한 운송 티켓, 다른 서비스에 대한 도어 키(door key) 등과 같이 보일 수 있다. 서비스 제공자는 서비스에 액세스하기 위해 별도의 장치를 발급할 필요가 없기 때문에 서비스 제공 및 해당 서비스 업그레이드의 복잡성과 비용을 절감할 수 있다.

[0620] 확장 가능한 프레임워크는 장치가 외관을 바꿀 수 있게 하며, 장치가 사용되는 컨텍스트에 의해서 요구되는 크리덴셜 및 서비스들을 제공한다. 예를 들어, 이는 사용자가 식료품점에 있는 거와 같은 독립적인 ATM에 액세스할 때 사용자의 운영자의 모양과 느낌을 취하기 위해 해당 ATM의 스크린을 조정하고, 사용자가 가입한 서비스들만 제공한다.

[0621] *다른 레이어와의 상호작용(Interaction with other layers)*

[0622] Tereon 시스템 내 다른 컴퍼넌트들과 상호작용할 수 있는 확장 가능한 프레임워크의 능력은 확장 가능한 프레임워크의 기본 특성이다. 더 넓은 보안 모델을 포함하는 컨텍추얼 보안(contextual security) 외에도, 확장 가능한 프레임워크 인스트럭션들은 해시-체인을 통해 전송되는 트랜잭션 정보 내에 임베디드될 수 있다(영지식 증명(zero knowledge proofs)을 갖는 해시 체인과 관련하여 개시된 바와 같이)).

[0623] *오프-라인 모드(Off-line mode)*

[0624] Tereon은 세가지 오프-라인 모드들을 제공한다; 사용자 오프-라인(user off-line), 머천트 오프-라인(merchant off-line), 및 둘 다 오프-라인(both off-line).

[0625] 처음 두 경우에는, Tereon은 사각형 반대 방향으로 이동하여 실시간 트랜잭션을 완료한다. 즉, 사용자는 머천트 단말기 및 머천트의 Tereon 서버를 통해 자신의 Tereon 서버와 통신한다. 머천트나 사용자 모두 서비스 저

하를 경험하지 않는다. Tereon은 관련 장치에 대한 사각형의 세면들을 통해 보안 통로(secure pathway)를 만들기 위해 PAKE 프로토콜 또는 유사한 기능을 가진 프로토콜을 사용한다.

[0626] 두 장치 모두 오프-라인인 세번째 경우에는, 즉각적인 인상은 Tereon이 사용자 또는 머천트가 트랜잭션을 지원할 충분한 자금을 가지고 있는지 여부를 실시간 확인할 수 없으므로, Tereon이 극복하기 위해 설계된 신용 위험 노출(credit risk exposure)을 만든다.

[0627] 확장 가능한 프레임워크의 특성들 및 해시 체인의 버전을 사용하여, Tereon은 시스템이 자금을 계속 확인할 수 있도록 한다. 사용자와 머천트 모두는 자신의 모든 기능을 수행할 수 있다. 사용자는 모바일 또는 마이크로프로세서 카드를 사용해야만 하지만, 사용자나 머천트는 자신이 경험하는 서비스가 저하되는 것을 보지 않을 것이다. 머천트 장치와 사용자 장치 모두는 그것들 간의 트랜잭션의 암호화된 세부정보와, 머천트가 만든 이전 오프-라인 트랜잭션들의 랜덤 샘플을 저장한다. 머천트 장치는 사용자의 카드나 전화(phone)로 전달할 각 트랜잭션의 최대 카피 수를 설정한다.

[0628] Tereon은 어느 사용자가 오프-라인 장치들과 온-라인 장치들의 조합을 사용하여 계정 내 존재하는 것 이상을 인출하지 못하게 하기 위해, 비즈니스 로직과 보안 모델 및 해시 체인의 결합을 사용한다. 계정이 신용 기능을 제공하는 경우, 계정은 오프-라인 장치들만 지원할 수 있다. 오프-라인 로직은 크레딧을 필요로 하지 않지만, 크레딧을 제공하기 위한 허가는 서비스 제공자의 규제 기관(service provider's regulators)에 의해서 요구될 수 있다.

[0629] 장치가 오프-라인으로 작동하도록 승인되지 않은 경우, 오프-라인인 때 다른 장치와 거래할(transact) 수 없다. 장치의 서명이 온-라인 트랜잭션을 지원하는 것으로 만 식별하므로, 보안 및 인증 모델(security and authentication model)은 그렇게 하는 것으로부터 방지하고, 장치는 등록된 계정의 가치에 영향을 줄 수 있는 트랜잭션을 처리할 수 없다.

[0630] 장치가 오프-라인 트랜잭션들을 지원할 수 있는 경우, 그 다음에 서비스 제공자는 이것을 오프라인-허용량(off-line allowance)인 특정 금액(장치가 온-라인인 때 항상 업데이트되는 크레딧 한도, 또는 계정 잔액 중 일부)으로 제한한다. 장치는 계좌에서 총액(total value) 또는 해당 오프-라인 허용량으로의 자금의 이체 또는 결제만 승인할 수 있다. 물론, 서비스 제공자는 장치가 이체 또는 자금을 수용할 수 있도록 권한을 부여할 수 있으며, 이러한 수용의 가치(오프-라인 수용 허용량(off-line acceptance allowance))를 제한할 수 있다. 제1 장치가 오프-라인 동안 사용자가 계정에 액세스하면, 포털을 통해 직접 또는 다른 온-라인 장치로, 사용자는 계정 잔액에서 오프-라인 허용량을 차감한 금액까지만 계정 이체 또는 결제를 승인할 수 있다.

[0631] Tereon은 관련 레코드가 포함된 장치들 중 하나가 온-라인이 되면 모든 오프-라인 트랜잭션을 조정한다. 당연히 일부 트랜잭션들의 카피를 여러 번 받을 수 있지만, 이전 조정을 확인하는데 사용한다.

[0632] 따라서, 서버가 오프라인 장치로의 이체 또는 결제와 관련된 오프-라인 트랜잭션들의 제3자 서버들(third-party servers)로부터 레코드들을 수신하면, 그 다음에 그것은 일단 해당 트랜잭션들의 카피들을 충분히 수신하면 해당 트랜잭션들을 처리하고, 그 자금을 계정 잔액에 추가한다. 마찬가지로, 서버가 오프-라인 장치로부터의 결제 또는 이체와 관련된 오프-라인 트랜잭션들의 제3자 서버들(third-party servers)로부터 레코드들을 수신하면, 그 다음에 그것은 일단 해당 트랜잭션들의 충분한 카피들을 수신하면 해당 트랜잭션들을 처리하고, 계정 잔액과 나머지 오프-라인 허용량으로부터 해당 금액을 뺀다.

[0633] 주어진 도면이 결제에 관한 것이라도, 이것들을 시각화하기 쉽기 때문에, 동일한 동작 모드들(modes of operation)은 모든 유형의 트랜잭션 시스템에 적용될 수 있다. 한 가지 예는 IoT 장치들 또는 다른 산업 컴퍼넌트들 간의 상호작용이다. 재배치, 삽입, 또는 제거될 수 있는 모듈들을 포함하는 워크플로우들을 생성함으로써, 운영자자들은 재프로그램 및 재설치 할 필요없이 새로운 방식으로 작동하도록 장치를 재구성할 수 있다.

[0634] 운영자들은 현장에서 장치의 용도를 변경하거나, 작동 방식을 변경하거나, 장치가 다른 장치를 제어하고 해당 장치가 작동하는 환경에서 감지한 변경 사항에 따라 워크 플로우들을 수정할 수 있다.

[0635] 또한, IoT 장치들은 필요할 때, 워크 플로우들을 구성하는 모듈들의 어셈블리를 수정하여 서로의 워크 플로우들을 수정할 수 있다. 룩-업 서비스(look-up service)가 장치들이 서로를 식별하고 인증할 수 있도록 하는 동안, 장치 간 통신(inter-device communications)을 관리하는 보안 모델은 중간자 공격(man-in-the-middle attacks)에 대한 통신을 차단한다.

[0636] 오프-라인 모드는 이러한 장치들이 자율적 또는 반자동적으로 작동하고 서로 상호 운용되며, 해당 장치 간의 모

든 트랜잭션을 확인 및 검증하고, 필요할 때만 운영자의 시스템과 상호작용할 수 있도록 한다.

[0637] 아래 설명된 컨텍스추얼 보안 모델(contextual security model)은 IOT 장치와 같은 모든 유형의 장치까지 확장된다. 장치가 작동하도록 허가되어 있고 해당 장치의 서비스들이 관련 룩-업 서비스에 나열되어 있는 한, 어느 장치나 다른 장치와 통신하고, 각각은 장치들 간 트랜잭션 및 데이터 통신을 신뢰하고 유효하게 하기 위해 해시 체인을 사용하고, 각각은 장치들의 워크 플로우들을 수정하거나 장치의 시스템들을 업그레이드하거나 해당 시스템들 간에 데이터를 단순히 전달 또는 대조하기 위한 인스트럭션들을 포함한다. 각 장치는 트랜잭션들에 대한 완전한 감사를 유지한다.

[0638] 보안(Security)

[0639] Tereon 시스템은 레거시 트랜잭션 처리 시스템에 사용되는 현재 보안 모델들 및 프로토콜들에 존재하는 결함 및 제한 사항을 극복하는 복수의 고유 보안 모델들을 사용한다. 예를 들어, 보안 모델들은 장치에 데이터를 저장할 필요성을 제거한다. 이는 기존 시스템들의 주요 이슈이다.

[0640] *USSD 보안(Securing USSD)*

[0641] USSD(unstructured supplementary service data)는 피처 폰들(feature phones)과의 결제를 비롯한 다양한 거래 유형에 대한 통신 채널로 일반적으로 사용된다. Tereon은 USSD를 안전하게 사용할 수 있도록 한다.

[0642] 대부분의 구현에서는 사용자가 USSD 코드를 입력하거나 번호가 매겨진 메뉴에서 동작(action)을 선택하도록 요구한다. 일련의 암호화되지 않은 메시지들은 앞뒤로 이동한다. 이는 비용, 보안 부족, 및 사용자 경험 부족의 이슈를 발생시킨다.

[0643] 보안 우려가 발생하는 곳으로 7 비트 또는 8비트 텍스트로 메시지들을 전송하는 대신에, Tereon은 USSD 및 유사한 통신 채널들을 새로운 방식으로 사용한다. Tereon은 단순히 그것을 세션 기반의 짧은 버스트 통신 채널(session-based short-burst communications channel)로 간주한다.

[0644] Tereon은 기존 시스템 이하는 USSD에 맞게 메시지를 조정하지 않는다. 대신, 트랜잭션 세션에서 각 암호화된 통신에 대해, Tereon은 사이퍼 텍스트(cypher text)를 생성하기 위해 TCP/IP(즉, GPRS, 3G, 4G, WiFi 등)를 통한 통신과 마찬가지로 통신을 암호화하고, 사이퍼 텍스트를 기본64 7-비트 문자열(base64 7-bit character string)로 인코딩한다. 그런 다음, Tereon은 사이퍼 텍스트의 길이를 확인한다. USSD 메시지들의 허용된 공간보다 길면, 사이퍼 텍스트를 두 개 이상의 부분으로 자르고, USSD를 사용하여 개별적으로 이들을 전송한다. 반대쪽 끝에서는, Tereon은 부분들을 전체 문자열(whole character string)로 재조합하고, 이를 사이퍼 텍스트로 변환하고, 해독한다.

[0645] Tereon은 당사자들을 식별하고 인증하기 위해 먼저 TLS(transport layer security)를 사용하는 이 방법을 사용한다. 이는 제1 세션 키(first session key)를 생성한다. 그런 다음, Tereon은 당사자들이 세션에서 이후 모든 통신들을 암호화하는데 사용할 제2 세션 키(second session key)를 생성하는 PAKE 프로토콜 협상을 암호화하기 위해 이 세션 키를 사용할 수 있다.

[0646] 일부 피처 폰들은 WAP(wireless application protocol)을 지원한다. 이러한 구현들이 USSD에서 WAP를 사용하는 경우, Tereon은 WAP 프로토콜 스택을 USSD에서 통신하는 방법으로 사용한다. 이것은 추가 인증 수준으로서 역할을 단순히 수행하는 WTLS(wireless transport layer security) 레이어를 제공한다(이것은 Tereon이 기본으로 사용하는 TLS 및 고급 암화화 표준 256(advanced encryption standard 256(AES256))보다 약하므로, Tereon은 모든 이벤트에 통신들을 암호화하기 위해 AES256을 사용).

[0647] 이것은 Tereon이 보안이 부족한 다른 통신 채널(예를 들어, NFC, 블루투스 등)을 확보하는 방법이기도 하다. 메시징 세션(messaging session)을 신중하게 구성하면, USSD 및 기타 '보안되지 않는(unsecured)' 채널들의 성격은 완전히 변경될 수 있다.

[0648] *능동 장치들(및 IoT)의 보안 모델(Security model for active devices(and the Internet of Things))*

[0649] 모바일 폰, 카드 단말기 등과 같은 능동 장치들의 보안 모델은 카드들에 대한 보안 모델과 유사한 방식으로 동작한다(아래 참조). 보안 알고리즘이 얼마 전에 깨졌으므로, SIM은 사용되지 않는다. 대신에, 장치에 암호화되어 저장되는 등록 키(registration key)는 네트워크가 생성하는 고유한 키와 함께 사용된다. 모바일 장치들에서, Tereon은 모바일에 의해서 보고되는 IMSI(international mobile subscriber identity)가 정품인지를 확인하는 룩-업을 수행하기 위해 해당 키를 사용할 수 있다.

[0650] 사용자가 애플리케이션을 처음 실행하면(사용자들이 원하는 경우 다수의 애플리케이션들을 갖을 수 있음), 애플리케이션은 Tereon 서버가 장치의 모바일 번호 또는 시리얼 번호(mobile number or serial number)와 함께 사용자의 계정에 대해 생성하는 일회성 인증 코드(one-time authentication code)를 요청한다(애플리케이션이 해당 번호를 먼저 확인할 수 없는 경우). 사용자는 다수의 Tereon 서버들에 자신의 애플리케이션을 등록할 수 있다. 여기서, 각 서버는 서버가 사용자에 대해 작동하는 각 계정 또는 서비스에 대해 고유한 일회성 활성화 코드(one-time activation code)를 생성한다.

[0651] 사용자가 일회성 활성화 코드를 입력하면, 애플리케이션은 제1 PAKE 세션을 생성하기 위해 해당 코드를 그것과 서버 사이의 공유 비밀(shared secret)로 사용한다(필요한 경우, 애플리케이션과 Tereon 서버가 TLS 또는 유사한 프로토콜을 사용하여 서로를 유효성 검사를 한 후). 그것들이 제1 PAKE 세션을 설립하면, Tereon 서버는 암호화되고 서명된 등록 키(encrypted and signed registration key)를 새로운 공유 비밀과 함께 애플리케이션에 전송한다. 서버와 애플리케이션은 일회성 활성화 코드, 등록키, 및 공유 비밀의 해시를 생성함으로써 새로운 공유 비밀을 생성하기 위해 일회성 활성화 코드, 등록키, 및 공유 비밀을 사용한다.

[0652] 서버와 애플리케이션이 통신할 때마다, 그것들은 그것들이 온-라인 통신 내 그것들 사이에서 통신했던 이전 메시지들의 해시로 이전 공유 비밀을 해싱하여 공유 비밀을 생성한다. 애플리케이션과 서버가 서로 통신할 때마다, 그것들은 그것들이 이전 교환의 해시와 교환한 트랜잭션의 컨텐츠들의 해시 - 트랜잭션 해시 -를 생성한다. 그것들 모두는 새로운 공유 비밀을 생성하기 위해 이 트랜잭션 해시를 사용한다.

[0653] 사용자가 장치를 분실하거나 애플리케이션을 다시 등록하거나 장치들을 변경해야 하는 경우, Tereon 서버는 새로운 일회성 인증 코드와 등록 키를 생성한다. 서버가 애플리케이션에 전달할 새로운 공유 비밀은 서버와 애플리케이션 사이에 교환된 이전 메시지들의 해시로부터 생성된다.

[0654] 이 키 전달(key forwarding)은 애플리케이션과 Tereon 서버가 각 PAKE 세션에 대해 새로운(또는 신선한) 공유 비밀을 갖도록 하게한다. 따라서, 공격자가 TLS 세션을 깨뜨릴 수 있는 경우(서버와 애플리케이션이 메시지들에 서명을 할 때 매우 어려움), 공격자는 여전히 PAKE 세션 키를 깨뜨릴 필요가 있다. 당사자가 적(feat)을 관리했다면, 당사자에게 세션에 대한 키가 주어졌을 것이다. 각 통신에 대해 새로운 키를 생성하는 프로세스는 당사자가 각 통신에 대해 적(feat)을 반복해야 한다는 것을 의미하며, 계산상 사실 불가능한 작업이라는 것이다.

[0655] 애플리케이션은 모든 세션에서 특정 서비스에 대해 인증되기 때문에, 사용자의 애플리케이션은 해당 서비스와만 상호작용할 것이다. 서버는 사용자의 애플리케이션이 등록된 다른 서비스들을 알 지 못한다. 사실상, 애플리케이션들은 사용자가 등록될 수 있는 복수의 서비스들과 무관하게 '사이키 종이(psychic pater)'와 비슷한 것, 서비스에 요구되는 크레덴셜들만을 제공하는 식별 장치가 된다. 그것은 한 서비스에 대한 결제 장치, 다른 서비스에 대한 운송 티켓, 다른 서비스에 대한 도어 키(door key) 등과 같이 보일 수 있다. 서비스 제공자는 서비스에 액세스하기 위해 별도의 장치를 발급할 필요가 없기 때문에 서비스 제공 및 해당 서비스 업그레이드의 복잡성과 비용을 절감할 수 있다.

[0656] 보안 모델은 추가 이점을 갖는다. 사용자가 자신의 장치를 분실하면, 그 다음에 사용자는 정확히 같은 번호의 새로운 장치를 획득할 수 있다. 애플리케이션이 있는 이전 장치(old device)는 작동하지 않지만, 새로운 장치가 일단 등록되면 유효한 비밀 키와 등록 코드를 갖기 때문에 작동할 수 있다. 잃어버린 것과 잃어버린 장치를 보고하는 것 사이에 시간 차가 있을 수 있지만, 아무도 필요한 패스워드와 PIN 또는 다른 인증 토큰을 갖지 않기 때문에, 아무도 트랜잭션을 수행할 수 없다.

[0657] 사용자가 애플리케이션에 액세스하기 전에, 사용자 또는 Tereon 시스템 관리자는 암호를 요구하도록 애플리케이션을 구성할 수도 있다. 이 패스워드는 Tereon 서버와 함께 점검된다. 그것이 유효하다면, Tereon 서버는 애플리케이션에 (항상 서명되고 암호화된 통신으로)작동하도록 지시할 것이다. 패스워드가 유효하지 않으면, Tereon 서버는 애플리케이션에 제한된 수의 시도(limited number of attempts) 동안 새로운 패스워드를 요청하도록 지시할 것이다. 그런 다음, Tereon 서버는 사용자의 애플리케이션을 잠그고(lock out), 사용자는 애플리케이션의 잠금을 해제하고 장치를 재등록하기 위해 관리자를 컨택할 필요가 있다.

[0658] 각 크리덴셜은 시간이 맞춰져 있다. 즉, 이것은 한 사용자가 정의된 기간 동안 특정 크리덴셜을 할당받을 수 있다는 것과, 해당 기간 동안 해당 크리덴셜로 발생하는 모든 트랜잭션은 해당 사용자와 연결된다는 것을 의미한다. 해당 사용자가 크리덴셜을 변경하면, 그런 다음에 원래의 크리덴셜은 다른 사용자에게 할당될 수 있다. 그러나, 룩-업 서버는 크리덴셜과 해당 크리덴셜에 등록된 기간의 조합에 기초하여 트랜잭션들과 크리덴셜들을

계속 링크할 것이다.

[0659] 동일 모델은 'IoT' 내 장치들 간의 통신을 보호하기 위해 채택될 수 있다. 여기서, 인증서(certificate) 또는 하드웨어에 내장된 시리얼 번호(hard-wired serial number)는 각 장치를 식별하기 위해 사용될 수 있다. 그것이 트랜잭션 날짜 또는 장치 간에 전송된 이전 메시지들로 해시될 때, 그것은 각 장치가 첫번째 접촉(first contact)에서 스왑(swap)할 제1 공유 비밀이 된다. 두 개의 번호는 장치를 식별할 수 있는 공개 시리얼 번호, PKI(public key infrastructure) 인증서 대신의 역할, 및 공유 비밀로 작용할 수 있는 암호로 보호된 시리얼 번호로 사용될 수 있다. 대신에, 단일 시리얼 번호는 ID와 제1 공유 비밀로 사용될 수 있고, 새로운 비밀 키는 보안 통신 채널을 통해 업로드될 수 있다(시스템 아키텍처의 통신 레이들에 대한 설명 참조).

[0660] Tereon의 모바일 보안 모델은 다른 이점을 가진다. 운영자는 개별 서비스에 대한 액세스 권한을 설정하고, 특정 사용이 해당 서비스를 성공적으로 수행하려고 시도하는 장치 및 네트워크에 따라 액세스 수준을 구성하기 위해 이를 사용할 수 있다. 예를 들어, 제공자는 관리자가 모바일 장치가 아니라 고정된 장치를 통해서 보안 공용 네트워크를 통해 시스템 로그를 보고, 인터넷 네트워크를 통해 시스템 관리 기능에만 액세스할 수 있도록 지정할 수 있다.

[0661] 이 기능은 결제에 일부 애플리케이션을 갖지만(그것은 정의된 네트워크들 및 장치들에 대한 시스템 관리 기능들에 대한 액세스를 보장함), 민감한 또는 권한있는 콘텐츠에 대한 제한된 액세스가 필요한 다른 서비스에 대해서는 자체적으로 제공되므로, 사용자는 특정 데이터를 볼 수 있는 사람, 이러한 제3자가 볼 수 있는 데이터, 및 그렇게 할 수 있는 위치를 정확하게 제어할 수 있다.

[0662] 보안 모델은 조직이 모든 장치에 의해서 수집, 생성, 또는 전송되는 모든 데이터의 개인 정보 및 보안을 보장하도록 할 수 있다. 이것은 모든 장치나 트랜잭션, 결제에서부터, 의료 장치를 통해, 교통 센서, 기상 센서, 수류 감지기(water flow detector) 등에 적용될 수 있다.

[0663] *카드 보안 모델(Card security model)*

[0664] 호스트 카드 에뮬레이션을 사용하는 EMV 카드들 및 모바일 폰들은 칩 또는 폰의 보안 요소(secure element)에 PIN을 저장한다. 비접촉식 카드들(Contactless cards), 및 해당 카드들을 에뮬레이트한 모바일들은 대부분의 카드 세부사항(또는 세부정보)들(카드 정보)을 명확하게 또는 쉽게 읽을 수 있는 형식으로 저장한다. 카드 단말기들은 사용자가 카드에 저장된 PIN에 대해 입력하는 PIN을 확인한다. 이것은 EMV 시스템의 많은 약점이 밝혀지도록 하고, EMV 프로세스를 복수의 충분히 입증된 공격에 노출되도록 만든다.

[0665] Tereon은 카드에 인증 키만 저장하고 Tereon 서비스에 저장된 값(값이 실제 값과 일치하지 않는다는 것을 볼 수 있는 관리자에 대해 폐쇄되어 있는 데이터베이스의 보안 영역 내 있음)과 입력된 값을 확인하다. 그것은 서비스와 특정 기능, 자원, 시설, 또는 트랜잭션 유형, 또는 해당 서비스에 의해 제공되는 다른 유형의 서비스를 인증한다. Tereon은 두 가지 보안 모델을 사용하며, 그 중 하나는 다른 모델의 하위 집합(subset)이다.

[0666] 대부분의 카드들은 PAN(the long number)을 표시한다. Tereon은 계정을 식별하기 위해 이 번호를 사용하지 않는다. 오히려, 그것은 모바일 번호와 같은 방식으로 PAN을 사용한다; 이것은 단순히 액세스 크리덴셜이다. 각 카드는 암호화된 PAN을 갖는다. 카드는 모바일에 등록 키가 해당 장치를 인증하는 것과 같은 방식으로, 카드가 등록된 각 서비스에 대해 유효한 것으로 카드를 식별하는 암호화된 등록 키를 갖는다. Tereon 서비스에 등록된 암호화된 PAN 문자열과 관련된 주소 세부사항(또는 세부정보)을 아직 갖고 있지 않는 경우, 암호화된 코드는 머천트의 Tereon 서비스가 요청할 필요가 있는 국가 룩-업 디렉토리 서비스를 가리키는 프레픽스(prefix)를 갖는다.

[0667] 사용자가 카드를 단말기에 제시하는 때, 단말기는 암호화된 PAN을 판독하고, 카드의 등록된 단말기로 카드의 유효성을 검사하기 위해 이와 암호화된 등록 키를 사용한다. 사용자의 Tereon 서비스가 카드와 머천트의 Tereon 서비스를 모두 확인하고 인증하면, 사용자 서비스는 머천트의 Tereon 서비스에 PAN을 암호화되지 않는 형식으로 전송하고, 이에 그것은 암호화된 형식으로 이것을 캐시에 등록할 수 있다. 따라서, 사용자가 나중에 전자 상거래 포털 또는 머천트의 단말기를 통해 PAN을 투명하게 입력하면, 서비스는 연락할 다른 서비스를 알게 된다.

[0668] 카드 판독기(card reader)가 어떤 이유로든 카드를 판독할 수 없으면, 사용자 또는 머천트는 PAN을 입력할 수 있고, 머천트의 Tereon 서비스는 사용자의 Tereon 서비스의 주소를 획득하기 위해 해당 PAN을 사용한다. 카드의 PAN은 사용자가 사용할 수 있는 만은 크리덴셜들 중 하나일 뿐이다.

[0669] 일단 머천트의 Tereon 서비스가 카드를 인증하면, 머천트의 단말기는 해시된 키를 사용하여 TLS를 설정하고, 그

다음에 해시된 키를 사용하여 PAKE 세션을 Tereon 서비스와 설정한다(단말기가 서비스와 통신할 때마다 이전 키를 등록 키로 해시하여 PAKE 세션에 대한 새로운 공유 비밀을 생성함). 머천트의 단말기가 PIN을 요청할 때까지, 머천트 프로세스는 진행된다(결제 서비스 제공자에 의해 결정되고 Tereon 서비스의 비즈니스 규칙 엔진에 명시된 대로, 사용자가 해당 트랜잭션에 PIN을 필료로 하는 경우). 사용자의 Tereon 서비스는 머천트의 서비스와 PAKE 세션을 생성하고, 다음에 일회성 키를 머천트의 서비스에 전송하고, TLS를 먼저 사용하여 생성된 다른 PAKE 세션을 통해 암호화된 메시지를 단말기에 전송한다.

[0670] 머천트의 단말기는 키를 수신하고, 사용자에 의해 선택된 텍스트- 단말기가 머천트의 서비스에 의해 허가된 것을 보여줌 -를 표시하기 위해 메시지를 해독한다. 사용자는 단말기의 PAKE 세션을 통해 사용자의 서비스와 통신되는 자신의 PIN을 입력한다. 이 프로세스는 사용자가 자신의 PIN을 머천트 단말기에 입력해야 하는 경우에만 발생한다. 이것은 보안 앱- 머천트의 단말기가 사용자의 Tereon 서비스로부터 액세스하며, 사용자의 서비스가 안정하고 서명된 키 교환으로 단말기에 전송하는 제2 일회성 키로 암호화됨 -에 입력되므로, 머천트의 단말기는 PIN을 명확하게 볼 수 없다. 모든 통신은 일반적으로 머천트의 서비스를 통해 이루어지며, 단말기와 사용자 Tereon 서비스 간의 직접 통신은 단말기가 해당 기능을 지원할 수 있는 곳에서 설립될 수 있다.

[0671] 카드가 마이크로-프로세서 카드(Chip & PIN, 비접촉식, 또는 두가지 모두)인 경우, 카드는 발급 당시 처음 생성된 공유 비밀을 가질 수 있다.

[0672] 마이크로-프로세서 카드는 등록된 Tereon 서비스(또는 서비스용 서비스)와 세션을 설립하기 위해 PAKE를 사용한다. 이 세션은 Tereon 서비스가 있는 카드 단말기(모바일 태블릿 또는 PoS 카드 단말기일 수 있음)에 의해 설립된 세션과 함께 진행된다. 이것은 기존 단말기 및 Chip & PIN 카드들이 보여주는 주요 취약성(key vulnerability)- 다수의 '중간자(man-in-the-middle)' 또는 '웨지(wedge)' 공격을 통해 PIN 검증 프로세스를 방해하고 파괴하는 기존 인프라스트럭처의 취약성임 -을 즉시 제거한다.

[0673] 카드는 서비스로 전송할 키- 서비스가 PIN을 암호화하기 위해 머천트 단말기로 전송함 -를 생성하기 위해 이 채널을 사용한다. 카드가 마지막 온-라인 트랜잭션의 잔액, 오프-라인 트랜잭션들에 대해 사용할 일련의 키들을 생성하기 위한 시드로 사용할 키, 및 제3자 오프-라인 트랜잭션들의 레코드들을 저장할 때, 그것은 오프-라인 트랜잭션을 촉진하기 위해 이 채널을 사용한다.

[0674] 카드가 분실되거나 도난당한 경우, Tereon의 보안 모델은 발급자가 새로운 PAN을 발급할 필요가 없음을 의미한다.

[0675] *컨텍스트 기반 보안(Context based security)*

[0676] 대부분의 보안 프로토콜들은 몇 가지 크리덴셜을 사용하고, 기본 가정들을 기반으로 한다. 오류 및 보안 상 소실을 초래할 수 있는 가정들이다. Tereon 시스템은 시스템 없이 통신 네트워크가 불안정하고 신뢰할 수 없으며 장치가 작동하는 환경이 안전하지 않을 수 있다는 가정이외의 기존 가정에 의존하지 않는다.

[0677] Tereon 시스템은 여러 스테이지들을 거치고, 크리덴셜 세트와 해당 크리덴셜이 제시되는 컨텍스트를 모두 찾는다. 이는 추가적인 보안을 제공하고, 조직이 직원들 또는 구성원들이 일부 또는 모든 상황에서 자신의 장치(BYOD라고도 함)를 사용할 수 있게 하는 수단들 중에서 하나를 보호한다.

[0678] Tereon은 사용자의 패스워드, PIN, 또는 기타 직접 인증 크리덴셜들 뿐만 아니라, 장치의 세부사항(또는 세부정보)들, 해당 장치의 애플리케이션, 해당 장치가 Tereon에 액세스하는 네트워크, 세션 시간에 해당 장치의 지리적 위치, 및 사용자가 해당 장치로 액세스하고 있는 서비스 또는 정보를 사용한다.

[0679] Tereon은 크리덴셜을 가져와서, 해당 크리덴셜로 설정된 컨텍스트에 기초하여, 크리덴셜에 적합한 액세스 수준을 부여하는 정보에 대한 액세스를 제어한다.

[0680] 예를 들어, Tereon에서 승인하지 않은 개인 장치의 심층 관리 서비스들(deep administration services)에 액세스하려는 관리자는 해당 관리자가 직장과 회사 네트워크에 있는지 여부와 관계없이 해당 서비스로부터 차단된다. 그러나, 동일 관리자는 동일 장치의 시스템 로그들의 일부를 볼 권한이 있다.

[0681] 두번째 예는 컨텍스트 보안 모델이 세컨더리 사용자가 볼 수 있는 서비스를 관리하는 경우이다. 사용자는 설정한도(신용 한도 또는 사용 가능한 최대 금액까지)없이 예금, 인출, 및 결제와 같은 여러 기능을 제공하는 전화 또는 카드를 보유하고 있다. 사용자는 여러 차례 카페를 자주 방문했으며 항상 커피와 아몬드 크로와상을 구입했다. 현재, 사용자는 자신의 카드를 아들에게 주었고 총 지출 한도(total spending limit)를 카드에 대해 40 파운드로 설정했다. 사용자는 커피를 사기 위해 동일한 카페에 카드를 가져가는 아들의 사용을 위해 제2 PIN

(두번째 PIN)을 설정했다. 그는 과거에 6개를 이미 구입했기 때문에, Tereon 시스템은 일반적으로 사용자에게 무료 아몬드 크로와상을 제공했으며, 카페는 고객들에게 제안을 전달하기 위해 Tereon을 사용한다. 그러나, 사용자의 아들이 PIN을 입력하는 때, Tereon 시스템은 결제하고 있는 사용자의 아들(자신의 아버지의 PIN을 알지 못함)을 검출하고, 그가 땅콩 알레르기가 있고 그의 아버지가 아들의 PIN을 아들의 프로파일과 연결했기 때문에 오늘의 제안을 차단한다. 머천트는 무료 크로와상 제공에 대한 통지를 보지 못했으며, Tereon은 사용자의 아들이 견과류를 먹을 수 없다는 것을 안다. 머천트가 볼 수 있는 것은 커피에 대한 결제이다.

[0682] 사용자는 그의 아들이 10파운드까지 현금을 인출할 수 있도록 허용했지만, 자금을 예치하도록 허용한 것은 아니다. 따라서, 사용자의 아들은 최대 10파운드의 인출을 제공할 수 있는 머천트에 들어갈 때, 그는 머천트의 옵션을 볼 수 있다.

[0683] 컨텍스트 기반 보안은 액세스 제어 이상의 역할을 한다. 사용자가 장치를 제시하거나 사용하는 컨텍스트에 따라, 해당 장치는 해당 컨텍스트에 필요한 크리덴셜만 제공한다; 그것은 '사이킥 페이퍼(psychic paper)'가 된다. 이러한 방식으로, 디렉토리 서비스(216)는 컨텍스트 기반 보안을 지원할 수 있는 기능을 제공한다.

[0684] 컨텍스트 기반 보안은 특정 컨텍스트에 대한 별도의 크리덴셜들 및 장치의 필요성을 없애준다. 이제 단일 장치는 도서관의 도서관 카드 크리덴셜, 버스나 기차의 교통 티켓, 방이나 시설에 액세스하기 위한 보안 키, 회사 매점의 사내 결제 장치, 극장 티겟, 슈퍼마켓 내 표준 결제 장치, 운전 면허증, NHS 카드, 서비스에 자격이 있음을 입증하는 ID 카드 -서비스가 필요하면 머천트의 장치에 사진 ID를 가져올 수 있는 - 등이 될 수 있다.

[0685] Tereon인 동적인 실시간 트랜잭션 처리 및 결제를 제공하기 때문에, 관리자 또는 사용자는 허용된 컨텍스트나 크리덴셜을 실시간으로 수정, 추가 또는 취소할 수 있다. 수정은 서비스를 제공하는 Tereon 서버, 또는 룩-업 디렉토리 서비스 216, 또는 둘 모두에서 즉시 반영된다. 현재 시스템이 장치를 비활성화할 때까지, 분실된 장치들은 더 이상 재정적 또는 ID 노출의 위험을 제기할 필요가 없다. 사용자 또는 관리자가 크리덴셜 또는 컨텍스트를 취소하거나 수정하면, 변경 사항은 즉시 활성화된다.

[0686] *원 터치 트랜잭션(One touch transaction)*

[0687] Tereon은 기존 시스템의 보안 결함을 제거하는 원-버튼 트랜잭션 권한 부여 및 액세스 방법(one-button transaction authorisation and access method)을 구현한다. 예를 들어, 현재 PIN없는 또는 NPC 결제는 결제에 대한 인증을 제공하지 않으므로 매우 위험하다. 카드 발급자가 비접촉식 EMV 시스템에서 전화 또는 카드 크리덴셜을 취소할 때까지, 사용자는 모든 결제에 대해 책임을 진다. 장치가 발급자에 의해 취소되더라도, 소비자는 여전히 결제를 활성화하지 않았음을 증명해야 한다. 결제가 인증을 위해 PIN을 필요로 하지 않는 경우 어떻게 할 수 있을 까? 이것은 누군가가 비접촉식 카드나 전화를 들고 단순히 탭하고 가서 결제를 하도록 허용하는 구멍을 남기는 것이다. 장치가 취소될 때까지, 장치는 계속 유효하다.

[0688] Tereon은 각각이 운영하기 위해 컨텍스트들에 의존하는 3 가지 모드 중 하나에서 탭 앤 고(tap-and-go)를 지원한다. 이들 중 하나는 개인을 식별하는 접근 방식을 사용하는 원-터치 트랜잭션을 제공한다. 사용자와 서비스 제공자가 제공된 인증 수준이 만족스럽다는 점에 동의하는 경우, 시스템은 원-터치 인증 방법을 제공하여 장치가 큰 버튼을 표시하거나 사용자가 터치할 수 있도록 화면에 넓은 영역을 구성한다. 다른 모드들은 사용자가 크리덴셜들을 입력하지 않는 기존의 비접촉식 트랜잭션과 장치가 서로 식별한 후에 사용자가 표준 결제 크리덴셜을 입력하는 것과 같은 완전히 비접촉식 모드이다.

[0689] 버튼 또는 영역 자체는 터치 스크린을 통해 인증을 제공한다. 모든 개인은 개인이 누를 위치와 사용하는 압력 패턴의 관점에서 모두 독특한 방식으로 화면을 누른다. 개인이 이 기능을 사용하고자 하는 경우, Tereon은 해당 개인에게 그 개인의 서명 인쇄(signature press)를 알게 될 때까지 버튼 또는 영역을 여러 번 눌러 달라고 요청할 것이다. 스크린은 논리적으로 여러 개의 개별 셀로 나뉘며, Tereon은 트레이닝 기간 동안 사용자가 터치하는 셀의 근접도와 패턴을 보고 가능한 경우 압력 패턴과 사용자가 누를 때 발생하는 모든 장치 움직임을 확인한다. 사용자를 인증하는데 사용하는 프로파일을 작성하기 위해 해당 데이터를 사용하고 모니터한다.

[0690] 도 21은 상술한 임의의 하나 이상의 방법을 수행하게 하는 인스트럭션 세트가 실행될 수 있는 컴퓨팅 장치(2100)의 일 구현의 블록도를 도시한다. 대안적인 구현들에서, 컴퓨팅 장치는 근거리 통신망(LAN), 인트라넷(intranet), 엑스트라넷(extranet), 또는 인터넷 내의 다른 기계들에 접속될(예를 들어, 네트워크화될) 수 있다. 컴퓨팅 장치는 클라이언트-서버 네트워크 환경에서 서버 또는 클라이언트 기계의 용량으로 동작하거나 피어-투-피어(또는 분산(distributed)) 네트워크 환경에서 피어 기계로 동작할 수 있다. 컴퓨팅 장치는 PC(personal computer), 태블릿 컴퓨터(tablet computer), 셋톱 박스(STB), PDA(Personal Digital

Assistant), 셀룰러 전화(cellular telephone), 웹 기기(web appliance), 서버, 네트워크 라우터, 스위치 또는 브리지, 프로세서, 또는 기계에 의해 취해질 동작을 지정하는 일련의 인스트럭션(순차적 또는 다른 방법)을 실행할 수 있는 임의의 기계일 수 있다. 또한, 하나의 컴퓨팅 장치가 도시되어 있지만, "컴퓨팅 장치"라는 용어는 설명된 방법들 중 임의의 하나를 수행하기 이한 인스트럭션 세트(또는 다수의 세트)를 개별적 또는 공동으로 실행하는 임의의 기계들의 집합(예를 들어, 컴퓨터들)을 포함하도록 사용되어야 한다.

[0691] 예시적인 컴퓨팅 장치(2100)는 버스(2130)를 통해 서로 통신하는 처리 장치(2102), 메인 메모리(2104)(예를 들어, ROM(read-only memory), 플래시 메모리, SDRAM(synchronous DRAM) 또는 RDRAM(Rambus DRAM)과 같은 DRAM), 정적 메모리(2106)(예를 들어, 플래시 메모리, SRAM(static random access memory)), 및 세컨더리 메모리(예를 들어, 데이터 저장 장치(2118))를 포함한다.

[0692] 처리 장치(2102, processing device)는 마이크로 프로세서, 중앙 처리 장치(central processing unit) 등과 같은 하나 이상의 범용 프로세서(general-purpose processor)를 나타낸다. 특히, 처리 장치(2102)는 CISC(complex instruction set computing) 마이크로프로세서, RISC(reduced instruction set computing) 마이크로프로세서, VLIW(very long instruction word) 마이크로프로세서, 다른 인스트럭션 세트를 구현하는 프로세서, 또는 인스트럭션 세트들의 조합을 구현하는 프로세서들일 수 있다. 또한, 처리 장치(2102)는 ASIC(application specific integrated circuit), FPGA(field programmable gate array), DSP(digital signal processor), 네트워크 프로세서 등과 같은 하나 이상의 특수 목적 처리 장치(special-purpose processing device)일 수 있다. 처리 장치(2102)는 본 명세서에서 설명된 동작들 및 단계들을 수행하기 위한 처리 로직(인스트럭션들(2122))을 실행하도록 구성된다.

[0693] 컴퓨팅 장치(2100, computing device)는 네트워크 인터페이스 장치(2108, network interface device)를 더 포함할 수 있다. 또한, 컴퓨팅 장치(2100)는 비디오 디스플레이 유닛(2110, video display unit)(예를 들어, LCD(liquid crystal display), CRT(cathode ray tube)), 영숫자 입력 장치(2112, alphanumeric input device)(예를 들어, 키보드 또는 터치스크린), 커서 제어 장치(2114, cursor control device)(예를 들어, 마우스 또는 터치스크린), 및 오디오 장치(2116, audio device)(예를 들어, 스피커)를 포함할 수 있다.

[0694] 데이터 저장 장치(2118)는 상술한 임의의 하나 이상의 방법 또는 기능을 구현하는 하나 이상의 인스트럭션 세트(2122)가 저장된 하나 이상의 기계-판독 가능 저장 매체(2128, 또는 더 구체적으로 하나 이상의 비 일시적 컴퓨터 판독 가능 저장 매체(non-transitory computer-readable storage media))를 포함할 수 있다. 컴퓨터 시스템(2100), 메인 메모리(2104), 및 컴퓨터 판독 저장 매체를 구성하는 처리 장치(2102)에 의해 실행되는 동안, 인스트럭션들(2122)은 메인 메모리(2104) 및/또는 처리 장치(2102) 내에 완전히 또는 적어도 부분적으로 상주할 수 있다.

[0695] 상술한 다양한 방법들은 컴퓨터 프로그램에 의해 구현될 수 있다. 컴퓨터 프로그램은 상술한 하나 이상의 다양한 방법의 기능을 수행하도록 지시하도록 구성된 컴퓨터 코드를 포함할 수 있다. 그러한 방법을 수행하기 위한 컴퓨터 프로그램 및/또는 코드는 컴퓨터와 같은 장치, 하나 이상의 컴퓨터 판독 가능 매체, 또는 보다 일반적으로는 컴퓨터 프로그램 제품 상에 제공될 수 있다. 컴퓨터 판독 가능 매체는 일시적(transitory) 또는 비일시적(non-transitory) 일 수 있다. 예를 들어, 하나 이상의 컴퓨터 판독 가능 매체는 전자, 자기, 광학, 전자기, 적외선, 또는 반도체 시스템, 또는 데이터 전송(예를 들어, 인터넷을 통해 코드를 다운로드)을 위한 전파 매체일 수 있다. 대안적으로, 하나 이상의 컴퓨터 판독 가능 매체는 반도체 또는 고체 상태 메모리, 자기 테이프, 착탈식 컴퓨터 디스켓(removable computer diskette), RAM(random access memory), ROM(read-only memory), 강성 자기 디스크(rigid magnetic disc), 및 광학 디스크- CD-ROM, CD-R/W, 또는 DVD와 같음 -와 같은 하나 이상의 물리적 컴퓨터 판독 가능 매체의 형태를 취할 수 있다.

[0696] 일 구현 예에서, 여기에 설명된 모듈들, 컴포넌트들 및 다른 특징들은 개별 컴포넌트들로서 구현되거나, 개별화 서버(individualization server)의 일부로서 ASICS, FPGA, DSP 또는 유사한 장치들과 같은 하드웨어 컴퍼넌트들의 기능에 통합될 수 있다.

[0697] "하드웨어 컴퍼넌트"는 특정 동작을 수행할 수 있는 유형의(예를 들어, 일시적이지 않은(non-transitory)) 물리적 컴퍼넌트(예를 들어, 하나 이상의 프로세서 세트)이고, 특정 물리적 방식으로 구성되거나 배열될 수 있다. 하드웨어 컴퍼넌트는 특정 동작을 수행하도록 영구적으로 구성된 전용 회로 또는 로직(dedicated circuitry or logic)을 포함할 수 있다. 하드웨어 컴퍼넌트는 FPGA(field programmable gate array) 또는 ASIC와 같은 특수 목적 프로세서(special-purpose processor)이거나 포함할 수 있다. 또한, 하드웨어 컴퍼넌트는 특정 동작을 수행하기 위해 소프트웨어에 의해 일시적으로 구성되는 프로그래밍 가능한 로직 또는 회로(programmable logic or

circuitry)를 포함할 수 있다.

[0698] 따라서, "하드웨어 컴퍼넌트"라는 문구는 물리적으로 구성되거나, 영구적으로 구성되거나(예를 들어, 하드웨어에 내장된(hardwired)), 또는 특정 방식으로 동작하거나 기술된 특정 동작을 수행하도록 일시적으로 구성(예를 들어, 프로그래밍)될 수 있는 유형의 엔티티(entity)를 포함하는 것으로 이해되어야 한다.

[0699] 기계(machine)은 예를 들어, 물리적 기계, 논리적 기계, 가상 기계, 컨테이너, 또는 실행가능한 코드를 포함하기 위해 일반적으로 사용되는 메커니즘일 수 있다. 기계는 단일 기계일 수 있으며, 또는 기계들이 동일한 유형인지 또는 복수의 유형들인지 여부에 관계없이 복수의 연결된 또는 분산된 기계들을 나타낸다.

[0700] 모듈들 및 컴퍼넌트들은 하드웨어 장치들 내의 펌웨어 또는 기능 회로(functional circuitry)로 구현될 수도 있다. 또한, 모듈들 및 컴퍼넌트들은 하드웨어 장치 및 소프트웨어 컴퍼넌트들의 임의의 조합으로 또는 소프트웨어(예를 들어, 기계 판독 가능 매체 또는 전송 매체에 저장되거나 구현된 코드)로만 구현될 수 있다.

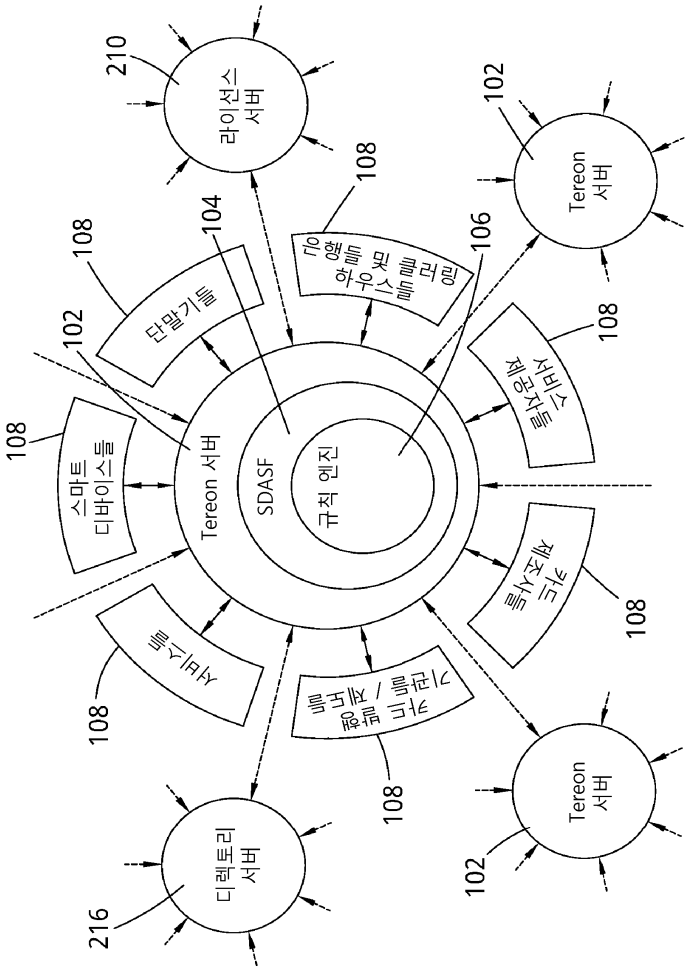
[0701] 달리 설명하지 않는 한, 다음의 설명으로부터 명백한 바와 같이, "송신(sending)", "수신(receiving)", "결정(determining)", "비교(comparing)", "가능(enabling)", "유지(maintaining)", "식별(identifying)" 등과 같은 용어는 컴퓨터 시스템 또는 유사 전자 컴퓨팅 장치- 컴퓨터 시스템의 레지스터 및 메모리 내의 물리적 (전자적) 양으로 표현된 데이터를 컴퓨터 시스템 메모리 또는 레지스터 또는 다른 정보 스토리지 내 물리량과 유사하게 표현되는 다른 데이터로 조작 및 변환함 - 전송 또는 디스플레이 장치의 동작(action) 및 프로세스를 지칭한다.

[0702] 상술한 설명은 예시적이고 제한적이 아닌 것으로 이해되어야 한다. 상술한 설명을 읽고 이해하면 많은 다른 구현 예가 당업자에게 명백할 것이다. 본 발명은 특정 예시적인 구현 예를 참조하여 설명되었지만, 설명된 실시예에 한정되지 않고 첨부된 청구 범위의 사상 및 범위 내에서 변형 및 변경하여 실시될 수 있음을 알 수 있을 것이다. 따라서, 명세서 및 도면은 제한적인 의미라기 보다는 예시적인 의미로 간주되어야 한다. 그러므로, 청구 범위가 속하는 균등물의 전체 범위와 함께, 첨부된 청구 범위를 참조하여 본 발명의 범위가 결정되어야 한다.

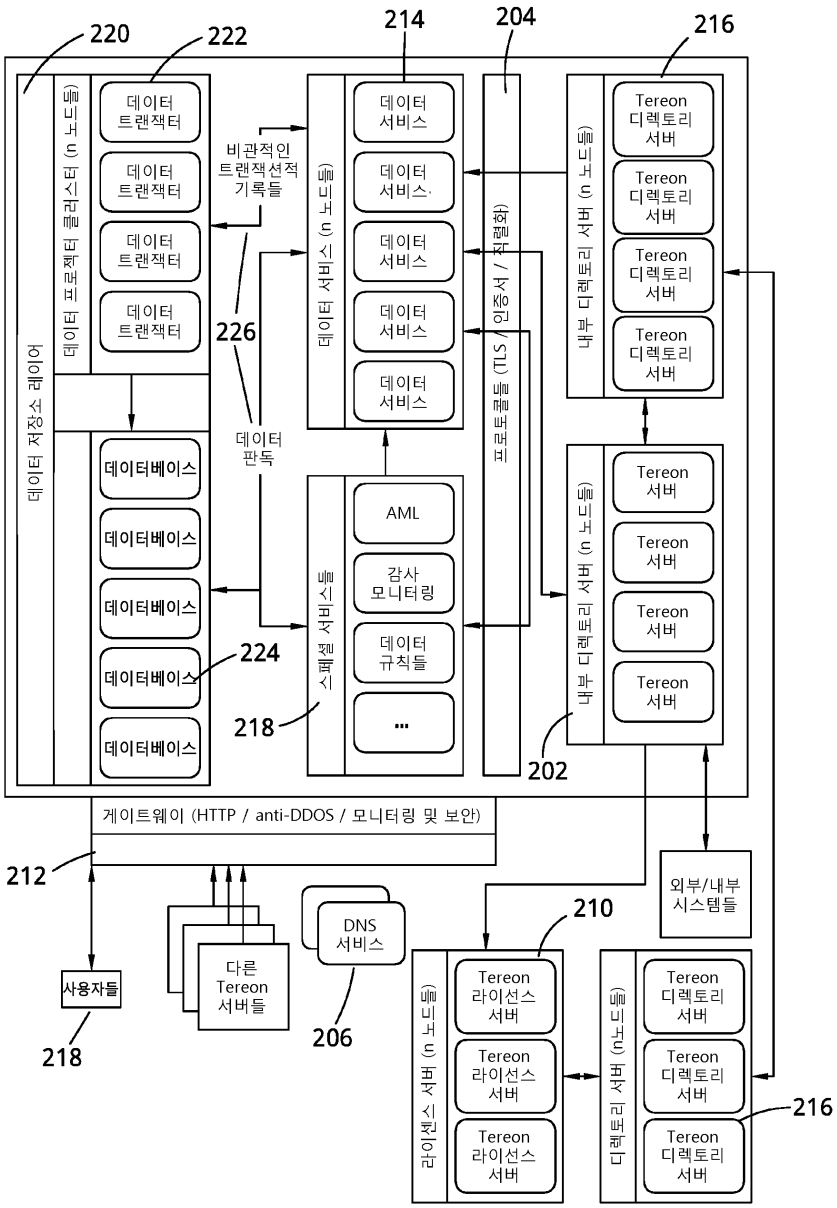
[0703] 다양한 측면의 모든 선택적인 특징은 다른 모든 측면과 관련된다. 기술된 실시예의 변형 예가 고려될 수 있으며, 예를 들어 개시된 모든 실시예의 특징이 임의의 방식으로 결합될 수 있다.

도면

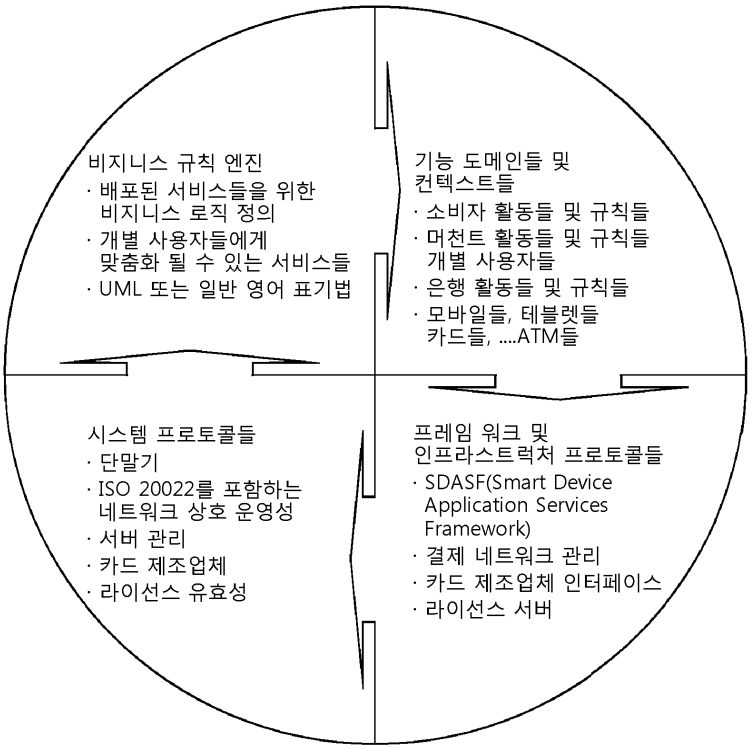
도면1


라이선스 서버
210
Tereon 서버
102
108
104
106
은행들 및 클리어링 하우스들
단말기들
108
102
서비스들
108
스마트 디바이스들
Tereon 서버
SDASF
규칙 엔진
108
카드 제조사들
108
카드 발행 기관들 / 처리들
108
서비스들
108
Tereon 서버
102
디렉토리 서버
216

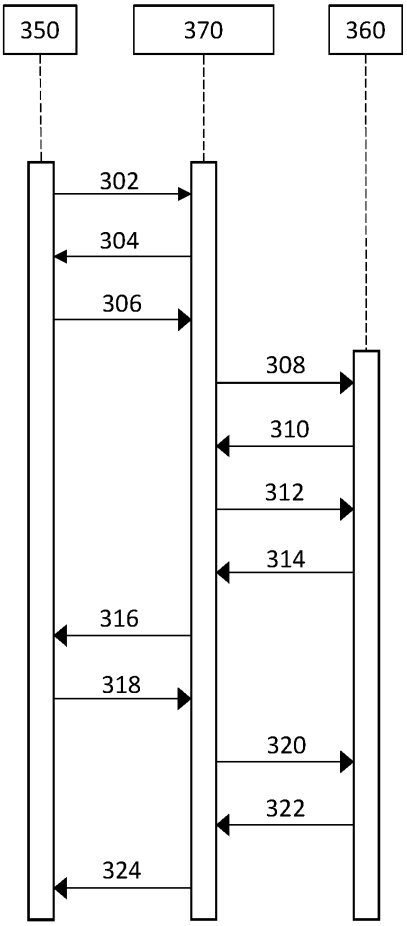
도면2


220
222
214
204
216
데이터 저장소 레이어
데이터 프로젝터 클러스터 (n 노드들)
데이터 트랜잭터
데이터 트랜잭터
데이터 트랜잭터
데이터 트랜잭터
비관적인 트랜잭션적 기록들
226
데이터 판독
데이터베이스
데이터베이스
데이터베이스
데이터베이스
데이터베이스
224
218
데이터 서비스들 (n 노드들)
데이터 서비스
데이터 서비스
데이터 서비스
데이터 서비스
데이터 서비스
스페셜 서비스들
AML
감사 모니터링
데이터 규칙들
...
프로토콜들 (TLS / 인증서 / 직렬화)
202
내부 디렉토리 서버 (n 노드들)
Tereon 디렉토리 서버
Tereon 디렉토리 서버
Tereon 디렉토리 서버
Tereon 디렉토리 서버
내부 디렉토리 서버 (n 노드들)
Tereon 서버
Tereon 서버
Tereon 서버
Tereon 서버
게이트웨이 (HTTP / anti-DDOS / 모니터링 및 보안)
212
외부/내부 시스템들
210
DNS 서비스
206
다른 Tereon 서버들
사용자들
218
라이센스 서버 (n 노드들)
Tereon 라이선스 서버
Tereon 라이선스 서버
Tereon 라이선스 서버
디렉토리 서버 (n노드들)
Tereon 디렉토리 서버
Tereon 디렉토리 서버
Tereon 디렉토리 서버
216

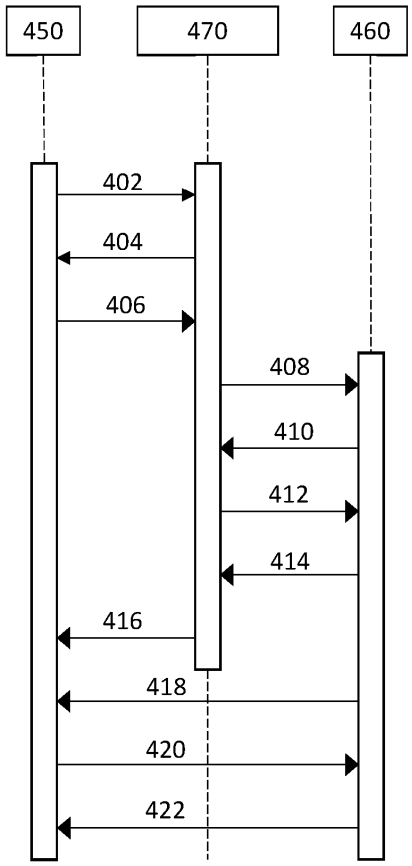
도면2a


비지니스 규칙 엔진
· 배포된 서비스들을 위한 비지니스 로직 정의
· 개별 사용자들에게 맞춤화 될 수 있는 서비스들
· UML 또는 일반 영어 표기법
기능 도메인들 및 컨텍스트들
· 소비자 활동들 및 규칙들
· 머천트 활동들 및 규칙들 개별 사용자들
· 은행 활동들 및 규칙들
· 모바일들, 테블렛들 카드들, ...ATM들
시스템 프로토콜들
· 단말기
· ISO 20022를 포함하는 네트워크 상호 운영성
· 서버 관리
· 카드 제조업체
· 라이선스 유효성
프레임 워크 및 인프라스트럭처 프로토콜들
· SDASF(Smart Device Application Services Framework)
· 결제 네트워크 관리
· 카드 제조업체 인터페이스
· 라이선스 서버

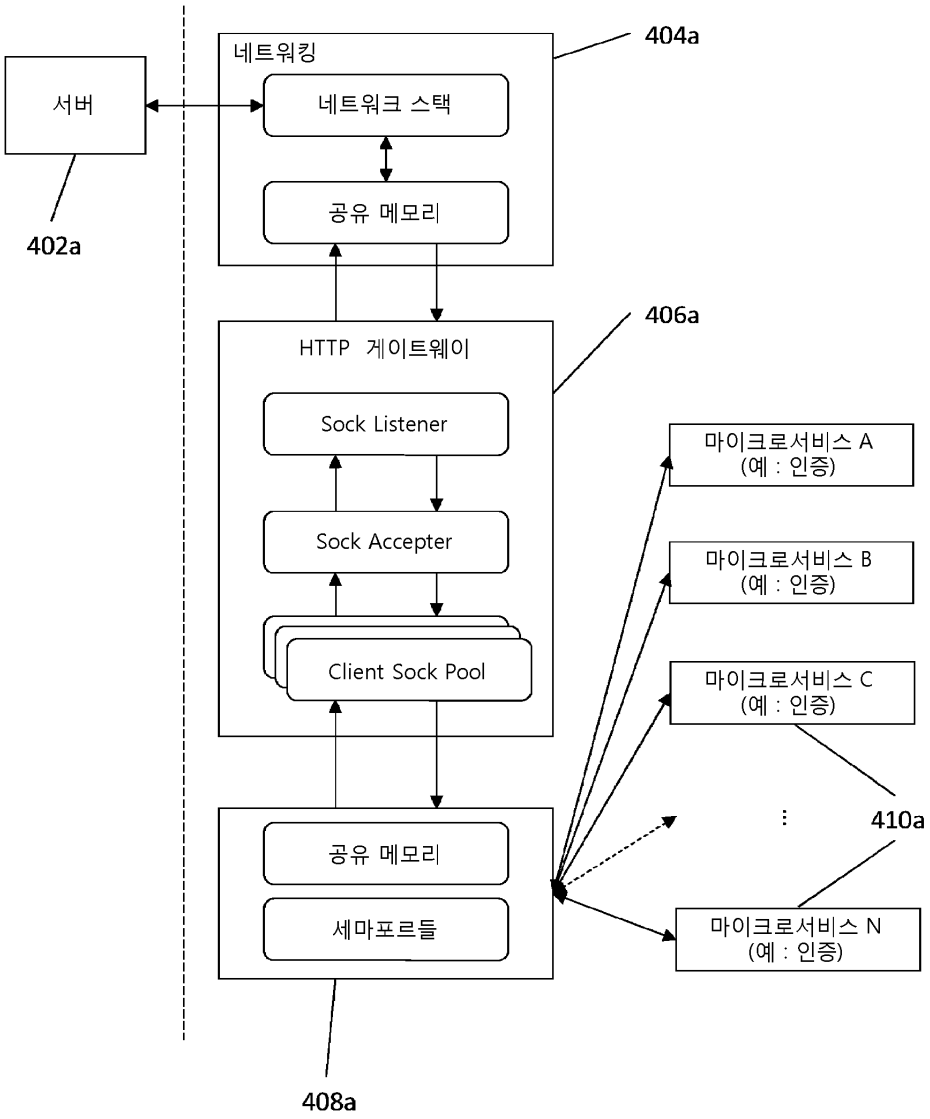
도면3



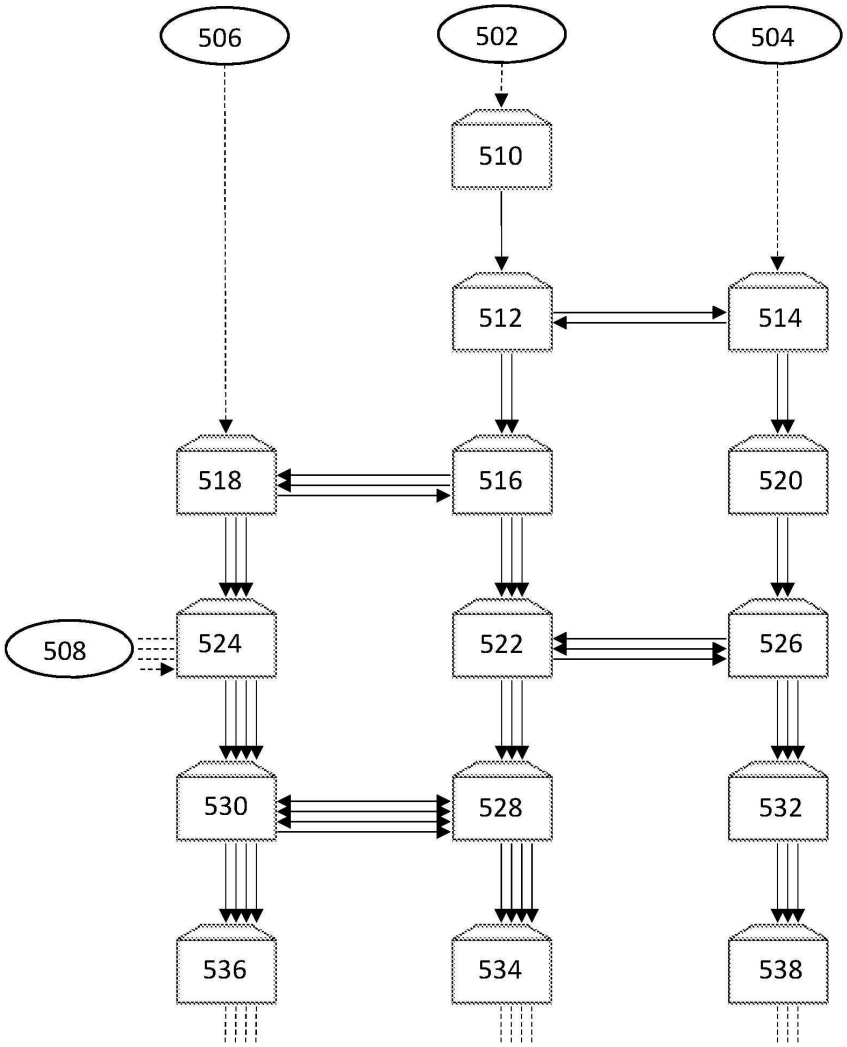
도면4



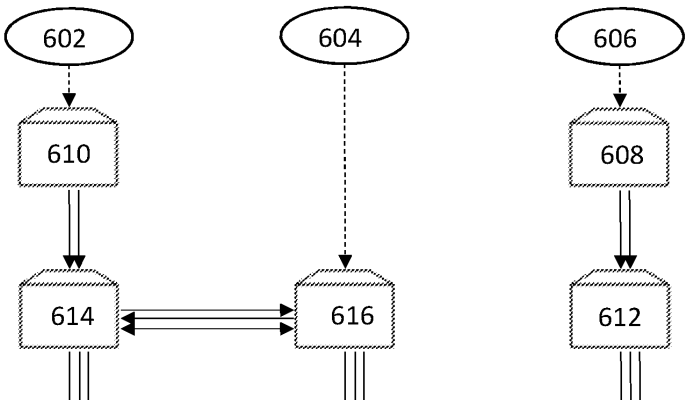
도면4a


서버
402a
네트워킹
404a
네트워크 스택
공유 메모리
HTTP 게이트웨이
406a
Sock Listener
Sock Accepter
Client Sock Pool
공유 메모리
세마포르들
408a
마이크로서비스 A
(예 : 인증)
마이크로서비스 B
(예 : 인증)
마이크로서비스 C
(예 : 인증)
410a
마이크로서비스 N
(예 : 인증)

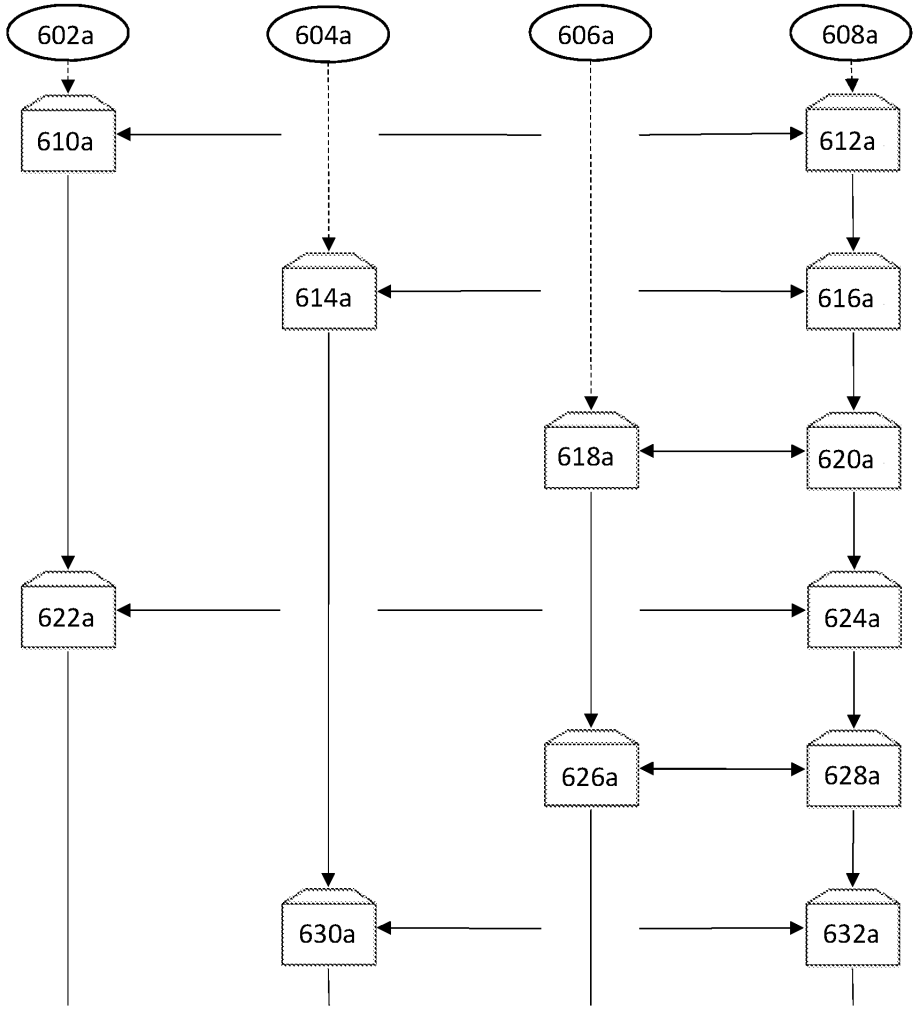
도면5


506
502
504
510
512
514
518
516
520
508
524
522
526
530
528
532
536
534
538

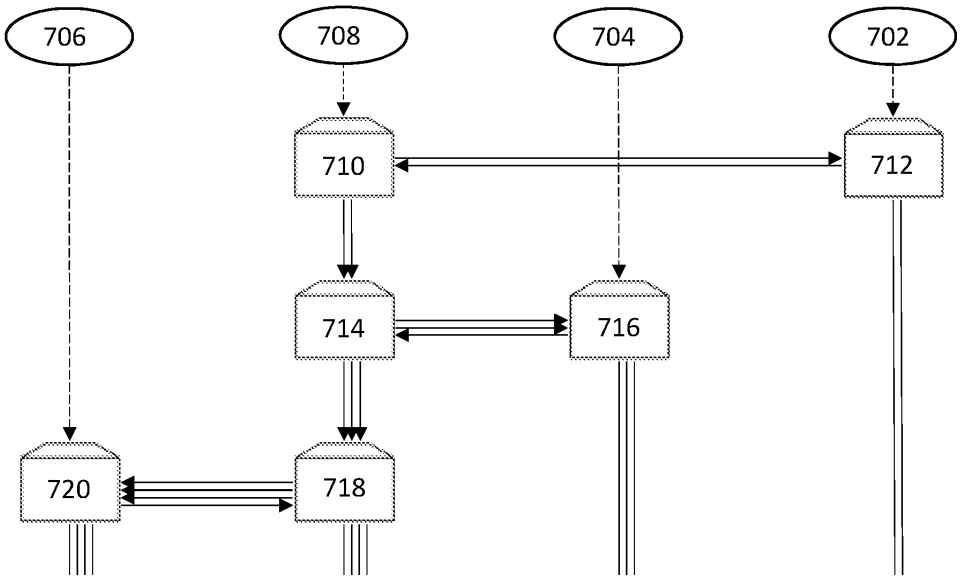
도면6


602
604
606
610
608
614
616
612

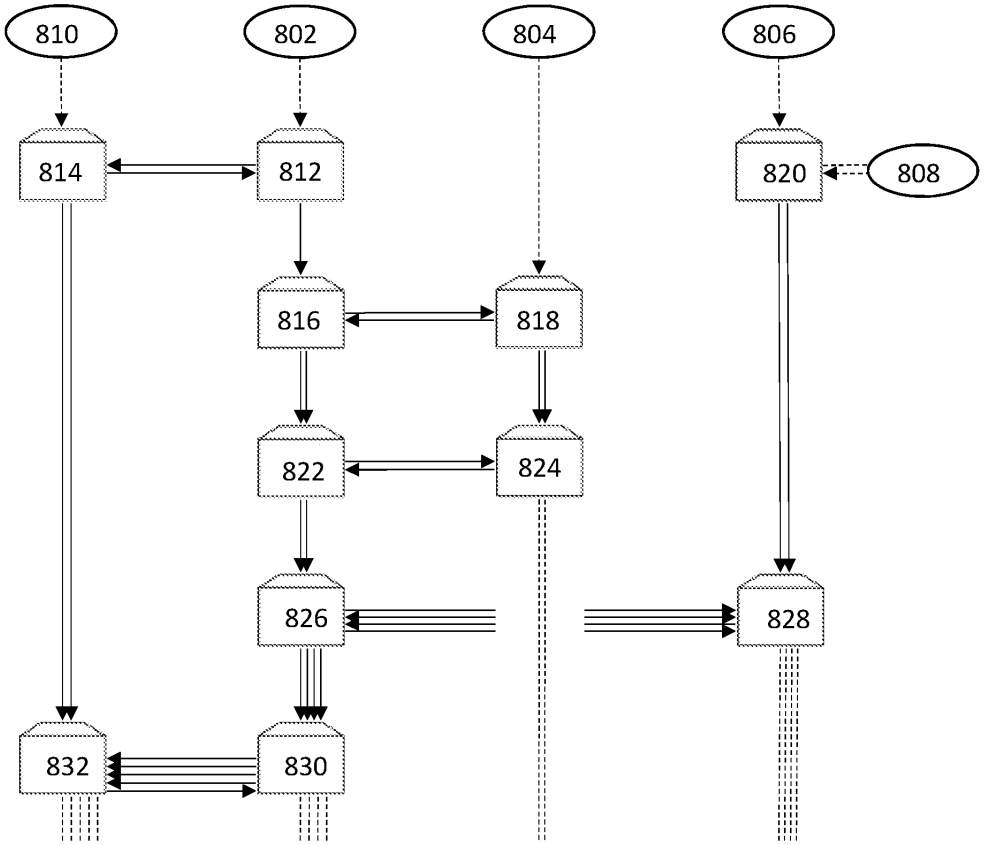
도면6a


602a
604a
606a
608a
610a
612a
614a
616a
618a
620a
622a
624a
626a
628a
630a
632a

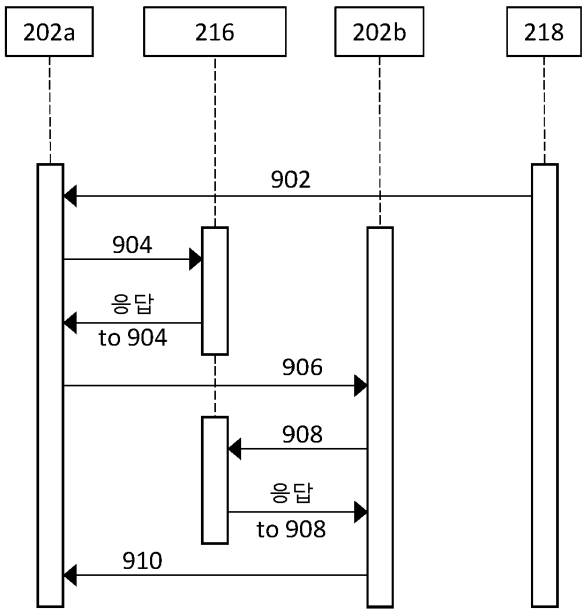
도면7


706
708
704
702
710
712
714
716
720
718

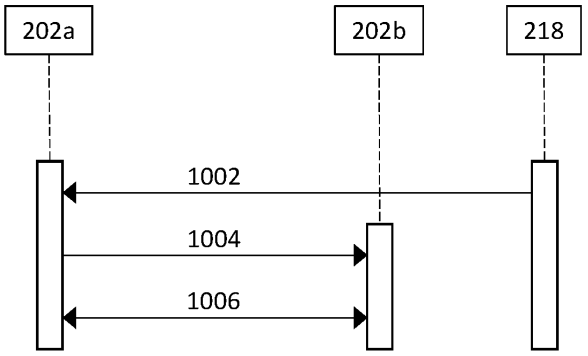
도면8


810
802
804
806
814
812
820
808
816
818
822
824
826
828
832
830

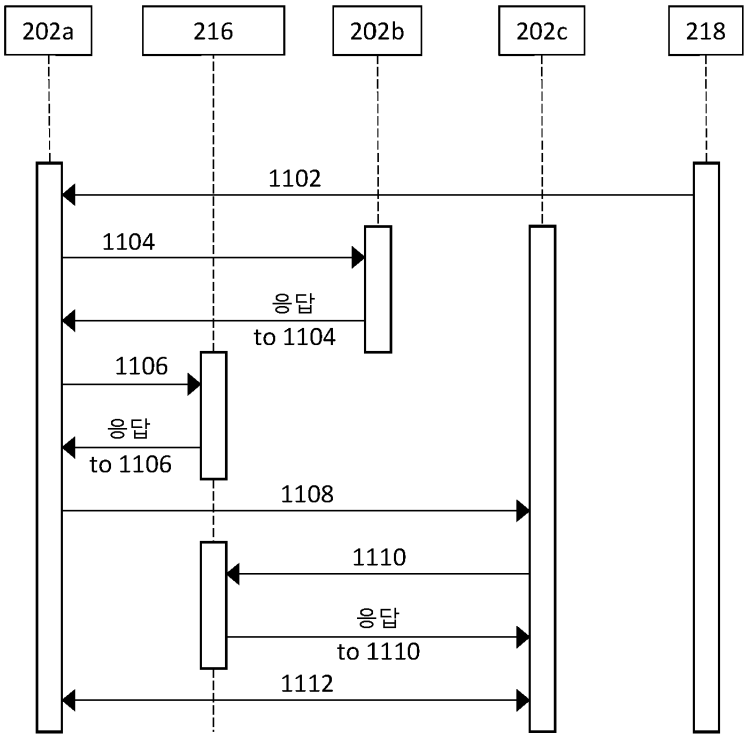
도면9


202a
216
202b
218
902
904
응답
to 904
906
908
응답
to 908
910

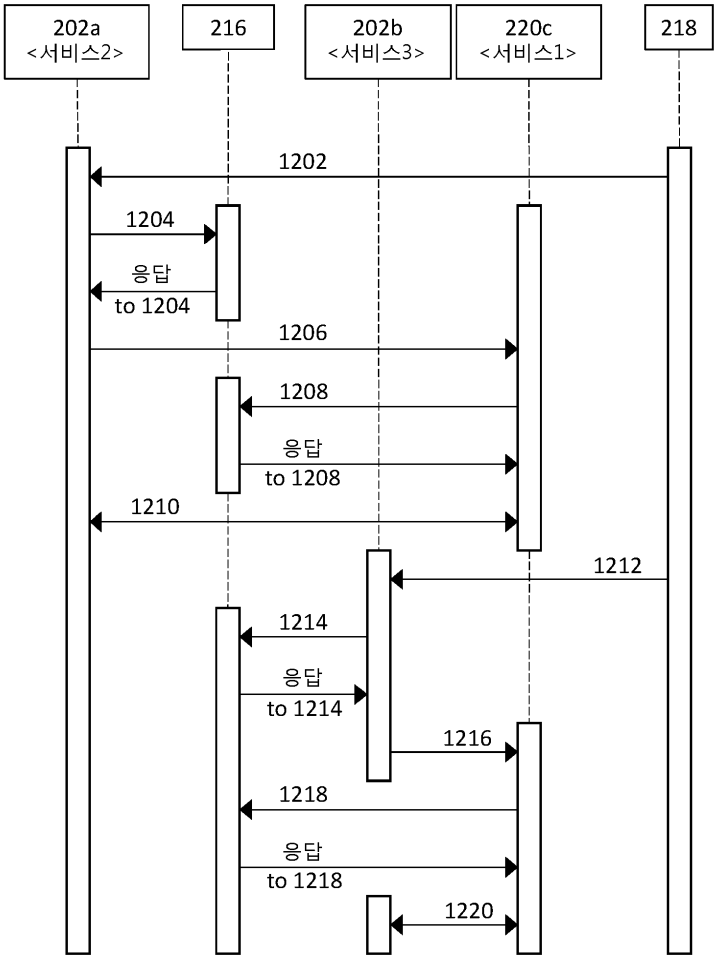
도면10


202a
202b
218
1002
1004
1006

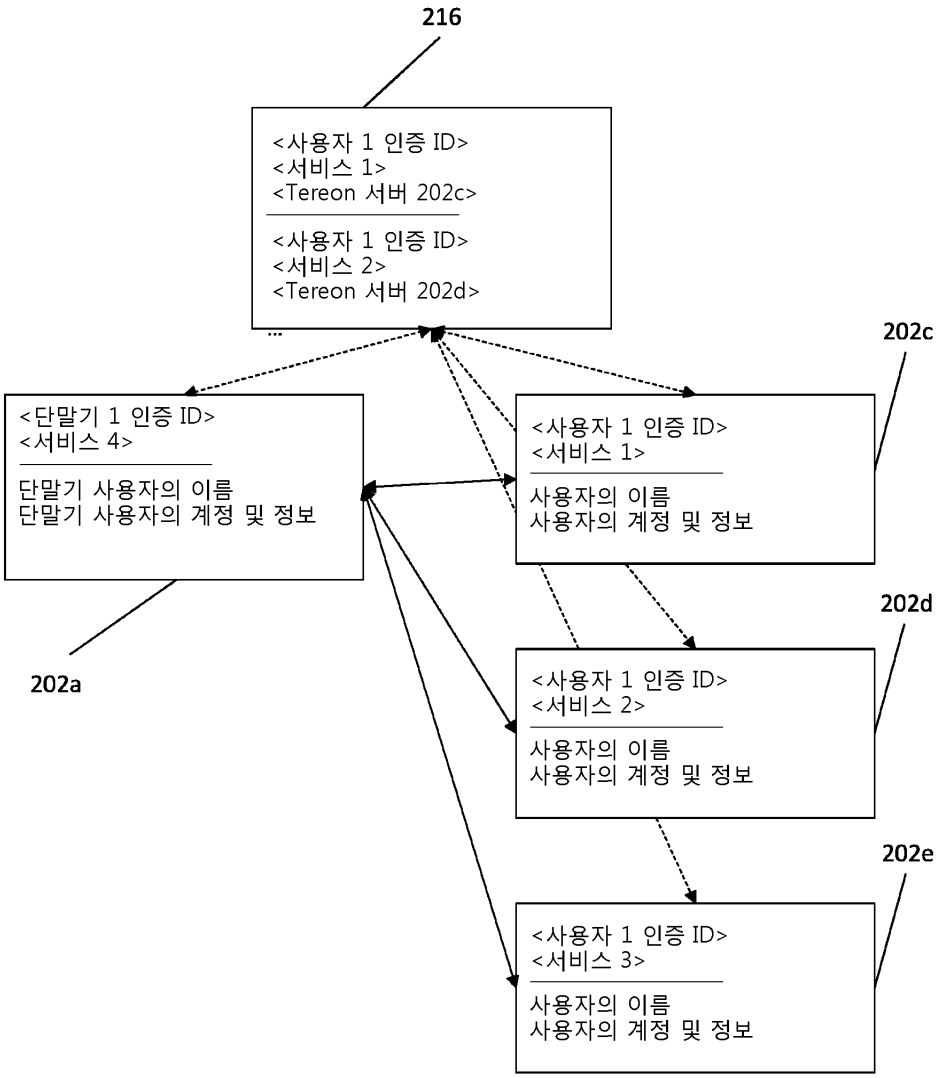
도면11


202a
216
202b
202c
218
1102
1104
응답
to 1104
1106
응답
to 1106
1108
1110
응답
to 1110
1112

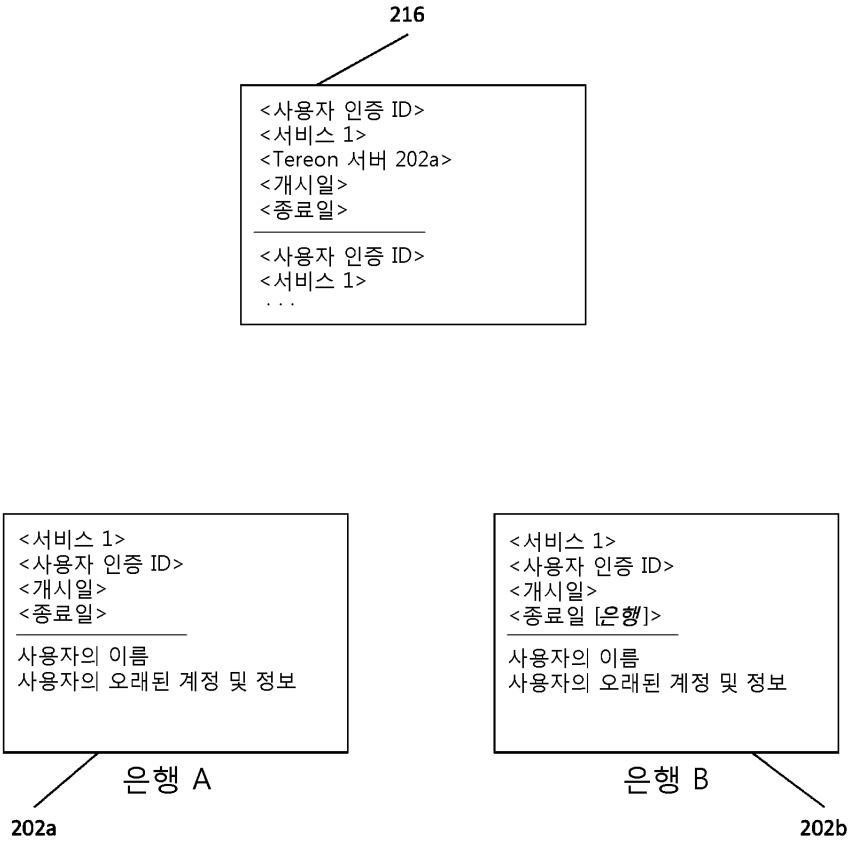
도면12


202a
<서비스2>
216
202b
<서비스3>
220c
<서비스1>
218
1202
1204
응답
to 1204
1206
1208
응답
to 1208
1210
1212
1214
응답
to 1214
1216
1218
응답
to 1218
1220

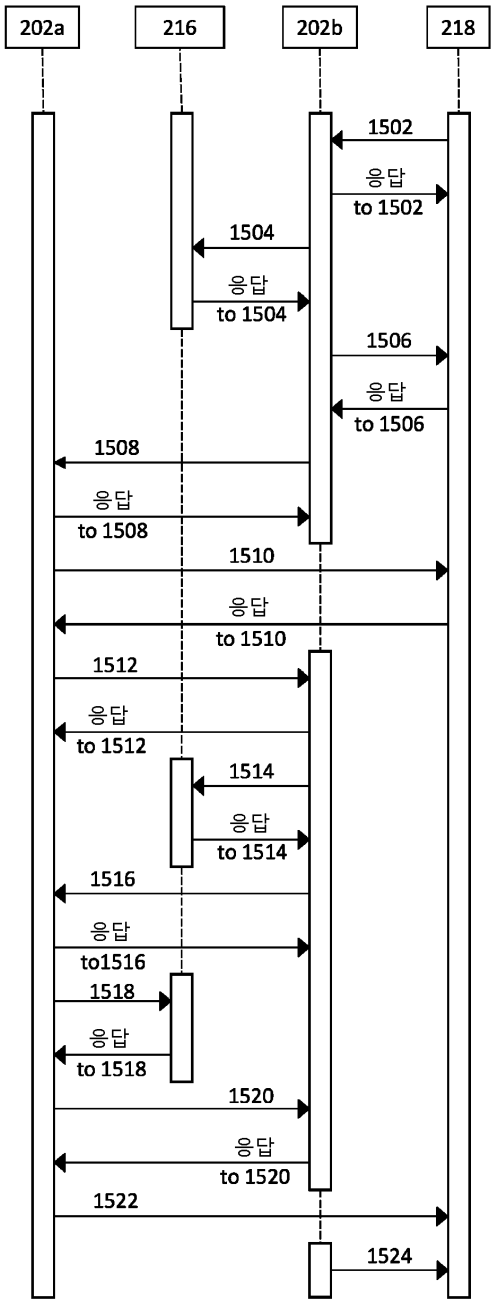
도면13


216
<사용자 1 인증 ID>
<서비스 1>
<Tereon 서버 202c>
<사용자 1 인증 ID>
<서비스 2>
<Tereon 서버 202d>
...
202c
<단말기 1 인증 ID>
<서비스 4>
단말기 사용자의 이름
단말기 사용자의 계정 및 정보
<사용자 1 인증 ID>
<서비스 1>
사용자의 이름
사용자의 계정 및 정보
202a
202d
<사용자 1 인증 ID>
<서비스 2>
사용자의 이름
사용자의 계정 및 정보
202e
<사용자 1 인증 ID>
<서비스 3>
사용자의 이름
사용자의 계정 및 정보

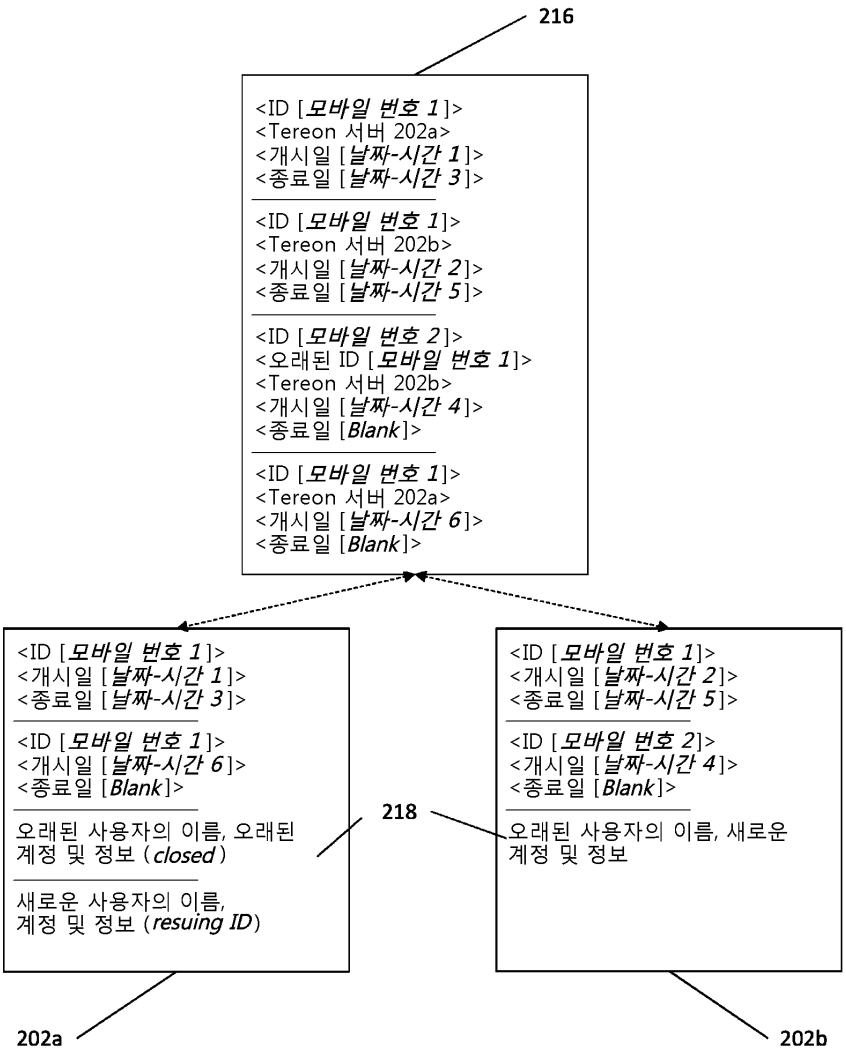
도면14


216
<사용자 인증 ID>
<서비스 1>
<Tereon 서버 202a>
<개시일>
<종료일>
<사용자 인증 ID>
<서비스 1>
. . .
<서비스 1>
<사용자 인증 ID>
<개시일>
<종료일>
사용자의 이름
사용자의 오래된 계정 및 정보
은행 A
202a
<서비스 1>
<사용자 인증 ID>
<개시일>
<종료일 [은행]>
사용자의 이름
사용자의 오래된 계정 및 정보
은행 B
202b

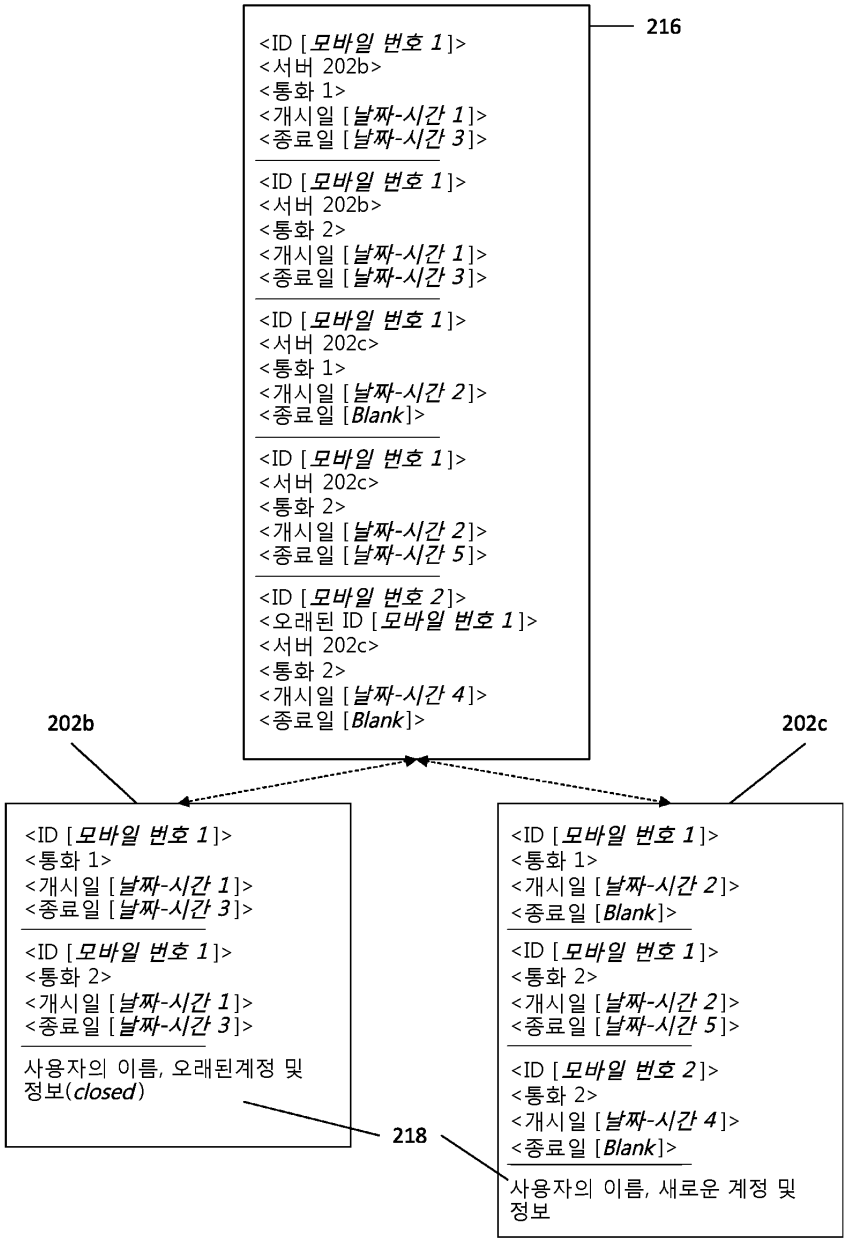
도면15


202a
216
202b
218
1502
응답
to 1502
1504
응답
to 1504
1506
응답
to 1506
1508
응답
to 1508
1510
응답
to 1510
1512
응답
to 1512
1514
응답
to 1514
1516
응답
to1516
1518
응답
to 1518
1520
응답
to 1520
1522
1524

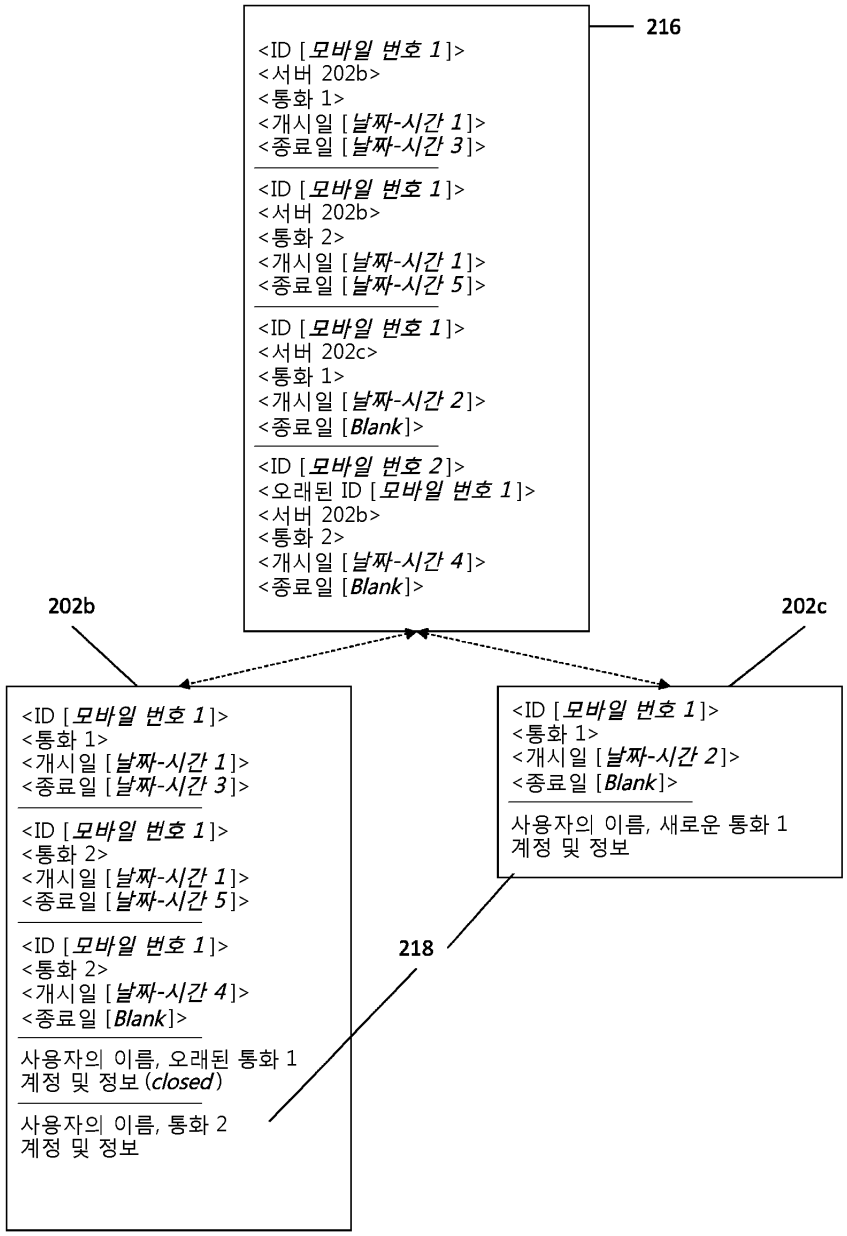
도면16


216
<ID [모바일 번호 1]>
<Tereon 서버 202a>
<개시일 [날짜-시간 1]>
<종료일 [날짜-시간 3]>
<ID [모바일 번호 1]>
<Tereon 서버 202b>
<개시일 [날짜-시간 2]>
<종료일 [날짜-시간 5]>
<ID [모바일 번호 2]>
<오래된 ID [모바일 번호 1]>
<Tereon 서버 202b>
<개시일 [날짜-시간 4]>
<종료일 [Blank]>
<ID [모바일 번호 1]>
<Tereon 서버 202a>
<개시일 [날짜-시간 6]>
<종료일 [Blank]>
<ID [모바일 번호 1]>
<개시일 [날짜-시간 1]>
<종료일 [날짜-시간 3]>
<ID [모바일 번호 1]>
<개시일 [날짜-시간 6]>
<종료일 [Blank]>
오래된 사용자의 이름, 오래된 계정 및 정보 (closed)
새로운 사용자의 이름, 계정 및 정보 (resuing ID)
218
<ID [모바일 번호 1]>
<개시일 [날짜-시간 2]>
<종료일 [날짜-시간 5]>
<ID [모바일 번호 2]>
<개시일 [날짜-시간 4]>
<종료일 [Blank]>
오래된 사용자의 이름, 새로운 계정 및 정보
202a
202b

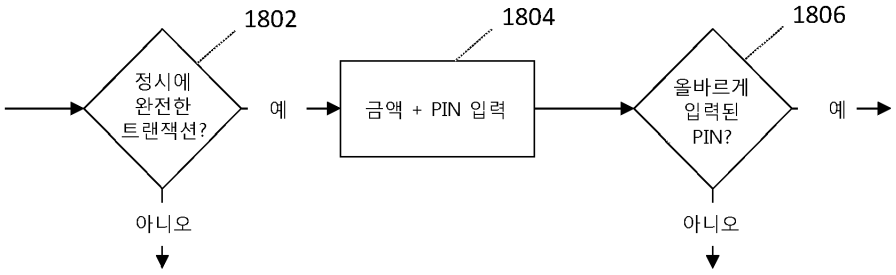
도면17


<ID [모바일 번호 1]>
<서버 202b>
<통화 1>
<개시일 [날짜-시간 1]>
<종료일 [날짜-시간 3]>
<ID [모바일 번호 1]>
<서버 202b>
<통화 2>
<개시일 [날짜-시간 1]>
<종료일 [날짜-시간 3]>
<ID [모바일 번호 1]>
<서버 202c>
<통화 1>
<개시일 [날짜-시간 2]>
<종료일 [Blank]>
<ID [모바일 번호 1]>
<서버 202c>
<통화 2>
<개시일 [날짜-시간 2]>
<종료일 [날짜-시간 5]>
<ID [모바일 번호 2]>
<오래된 ID [모바일 번호 1]>
<서버 202c>
<통화 2>
<개시일 [날짜-시간 4]>
<종료일 [Blank]>
216
202b
202c
<ID [모바일 번호 1]>
<통화 1>
<개시일 [날짜-시간 1]>
<종료일 [날짜-시간 3]>
<ID [모바일 번호 1]>
<통화 2>
<개시일 [날짜-시간 1]>
<종료일 [날짜-시간 3]>
사용자의 이름, 오래된계정 및 정보(closed)
218
<ID [모바일 번호 1]>
<통화 1>
<개시일 [날짜-시간 2]>
<종료일 [Blank]>
<ID [모바일 번호 1]>
<통화 2>
<개시일 [날짜-시간 2]>
<종료일 [날짜-시간 5]>
<ID [모바일 번호 2]>
<통화 2>
<개시일 [날짜-시간 4]>
<종료일 [Blank]>
사용자의 이름, 새로운 계정 및 정보

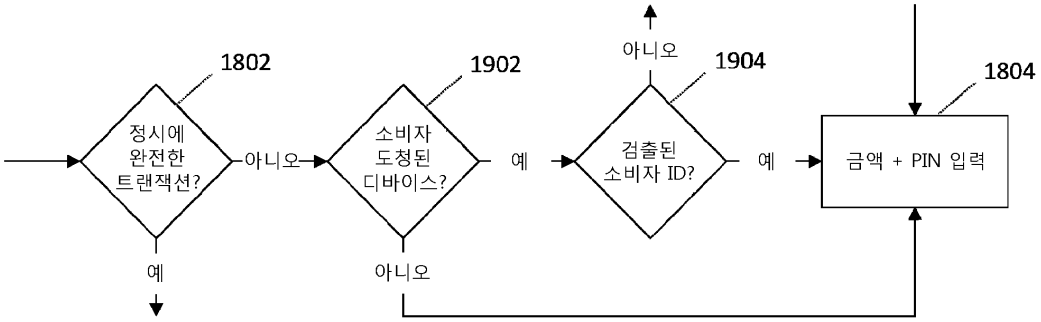
도면17a


216
<ID [모바일 번호 1]>
<서버 202b>
<통화 1>
<개시일 [날짜-시간 1]>
<종료일 [날짜-시간 3]>
<ID [모바일 번호 1]>
<서버 202b>
<통화 2>
<개시일 [날짜-시간 1]>
<종료일 [날짜-시간 5]>
<ID [모바일 번호 1]>
<서버 202c>
<통화 1>
<개시일 [날짜-시간 2]>
<종료일 [Blank]>
<ID [모바일 번호 2]>
<오래된 ID [모바일 번호 1]>
<서버 202b>
<통화 2>
<개시일 [날짜-시간 4]>
<종료일 [Blank]>
202b
202c
<ID [모바일 번호 1]>
<통화 1>
<개시일 [날짜-시간 1]>
<종료일 [날짜-시간 3]>
<ID [모바일 번호 1]>
<통화 2>
<개시일 [날짜-시간 1]>
<종료일 [날짜-시간 5]>
<ID [모바일 번호 1]>
<통화 2>
<개시일 [날짜-시간 4]>
<종료일 [Blank]>
사용자의 이름, 오래된 통화 1
계정 및 정보 (closed)
사용자의 이름, 통화 2
계정 및 정보
218
<ID [모바일 번호 1]>
<통화 1>
<개시일 [날짜-시간 2]>
<종료일 [Blank]>
사용자의 이름, 새로운 통화 1
계정 및 정보

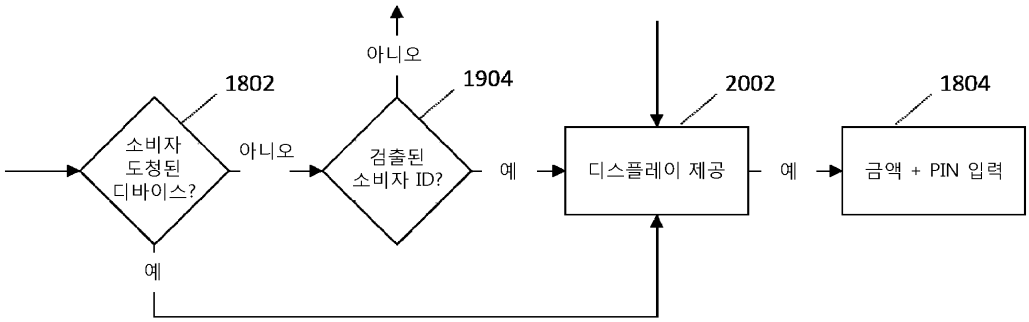
도면18


1802
정시에 완전한 트랜잭션?
예
아니오
1804
금액 + PIN 입력
1806
올바르게 입력된 PIN?
예
아니오

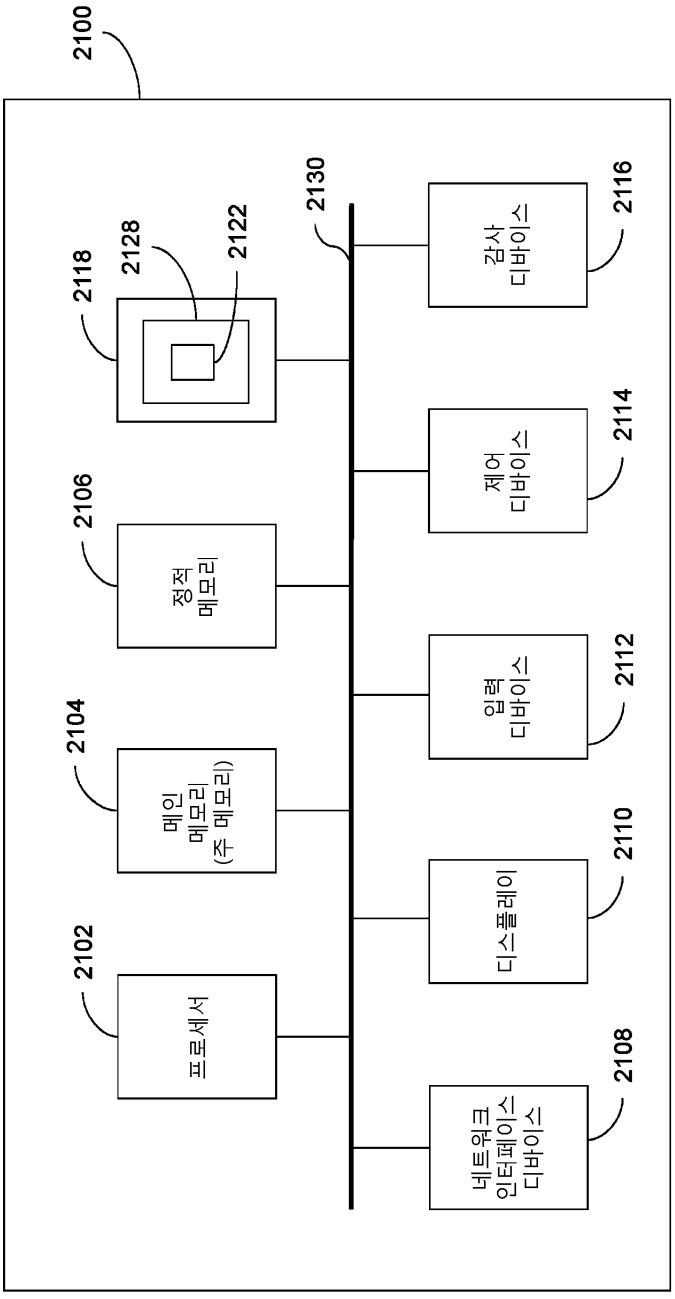
도면19


1802
정시에
완전한
트랜잭션?
아니오
1902
소비자
도청된
디바이스?
예
아니오
1904
검출된
소비자 ID?
예
아니오
1804
금액 + PIN 입력
예

도면20


1802
소비자
도청된
디바이스?
아니오
1904
검출된
소비자 ID?
예
아니오
2002
디스플레이 제공
예
1804
금액 + PIN 입력
예

도면21


2100
2130
2118
2128
2122
2106
2104
2102
2116
2114
2112
2110
2108
프로세서
메인 메모리 (주 메모리)
정적 메모리
네트워크 인터페이스 디바이스
디스플레이
입력 디바이스
제어 디바이스
감사 디바이스