(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2022년08월16일
(11) 등록번호 10-2432775
(24) 등록일자 2022년08월10일

(51) 국제특허분류(Int. Cl.)
G06F 8/34 (2018.01) *G06F 21/60* (2013.01)
G06F 21/62 (2013.01) *G06F 8/41* (2018.01)
G06Q 30/06 (2012.01) *H04L 9/08* (2006.01)
(52) CPC특허분류
G06F 8/34 (2013.01)
G06F 21/602 (2013.01)
(21) 출원번호 10-2021-0039691
(22) 출원일자 2021년03월26일
심사청구일자 2021년03월26일
(56) 선행기술조사문헌
US20190303541 A1
KR1020200094618 A
KR101849912 B1
KR1020200020291 A

(73) 특허권자
충남대학교 산학협력단
대전광역시 유성구 대학로 99 (궁동, 충남대학교)
(72) 발명자
조은선
대전광역시 유성구 어은로 57 한빛아파트, 110동 1504호
김지수
대전광역시 유성구 대학로 169 301호
(74) 대리인
이은철, 김재문

전체 청구항 수 : 총 13 항 심사관 : 지정훈

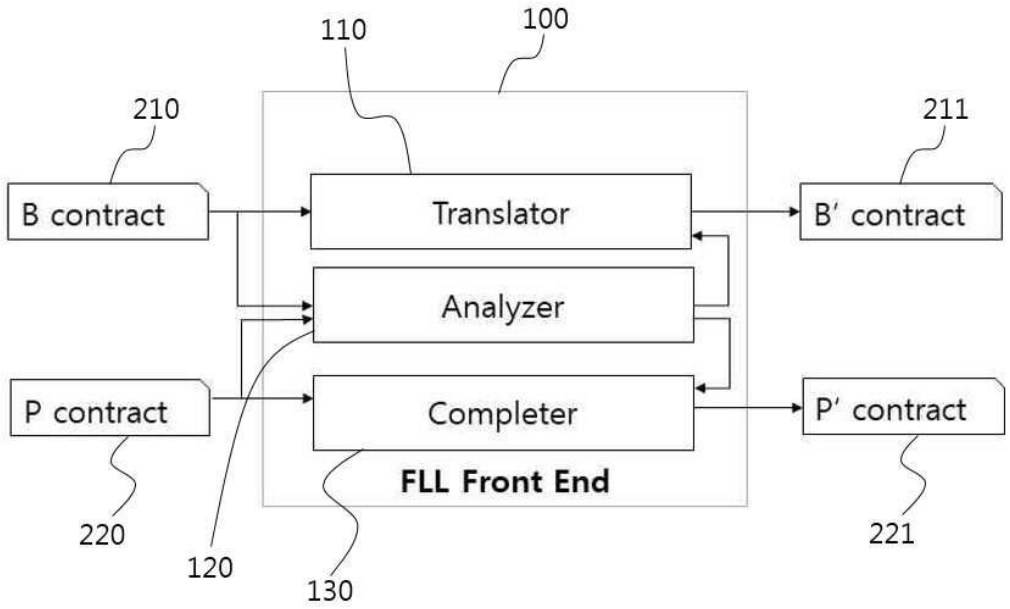
(54) 발명의 명칭 **개인정보보호를 지원하는 스마트 컨트랙트 프로그래밍 시스템 및 방법**

(57) 요 약

본 발명은 기능 중심의 개인정보 공개 및 위탁방법을 제공하는 FLL(Function level leak) 컴파일러를 이용하여 스마트 컨트랙트의 개인정보보호를 위한 정보 공개 및 위임과정에 대해 지원하는 개인정보보호를 지원하는 스마트 컨트랙트 프로그래밍 시스템 및 방법에 관한 것이다. 본 발명의 실시 예에 따른 FLL 컴파일러는 분석부, 변환

(뒷면에 계속)

대 표 도 - 도3



부 및 검증부를 포함한다. 또한, 상기 분석부는 위탁자로부터 제공되는 스마트 컨트랙트의 원래 계약 내용을 처리하는 B-컨트랙트(business logic contract)와, 상기 위탁자의 개인정보 처리를 위한 계약 내용을 포함하는 P-컨트랙트(privacy policy contract)를 분석한다. 상기 변환부는 분석부의 분석 결과를 토대로 개인정보의 공개여부가 함수단위로 정의되도록 B-컨트랙트를 원시 컨트랙트로 변환하고, 상기 P-컨트랙트에 위탁자로부터 제공되는 공개 데이터를 추가하여 삽입한다. 또한, 상기 검증부는 분석부 및 변환부의 결과 데이터를 토대로 스마트 컨트랙트의 개인정보보호 검증을 수행하기 위한 V-컨트랙트를 생성한다.

(52) CPC특허분류
G06F 21/6245 (2013.01)
G06F 8/41 (2013.01)
G06Q 30/06 (2013.01)
H04L 9/0825 (2013.01)
H04L 9/50 (2022.05)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711111605
과제번호	2020R1A2C2008864
부처명	과학기술정보통신부
과제관리(전문)기관명	한국연구재단
연구사업명	개인기초연구(과기정통부)(R&D)
연구과제명	영지식 증명을 지원하는 스마트 컨트랙트 프레임워크 연구
기 여 율	1/1
과제수행기관명	충남대학교
연구기간	2020.03.01 ~ 2023.02.28

명 세 서

청구범위

청구항 1

기능 중심의 개인정보 공개 및 위탁방법을 제공하는 FLL(Function level leak) 컴파일러를 이용하여 스마트 컨트랙트의 개인정보보호를 위한 정보 공개 및 위임과정에 대해 지원하는 스마트 컨트랙트 프로그래밍 시스템에 있어서,

상기 FLL 컴파일러는

위탁자로부터 제공되는 스마트 컨트랙트의 원래 계약 내용을 처리하는 B-컨트랙트(business logic contract)와, 상기 위탁자의 개인정보 처리를 위한 계약 내용을 포함하는 P-컨트랙트(privacy policy contract)를 분석하는 분석부;

상기 분석부의 분석 결과를 토대로 개인정보의 공개여부가 함수단위로 정의되도록 B-컨트랙트를 원시 컨트랙트로 변환하고, 상기 P-컨트랙트에 위탁자로부터 제공되는 공개 데이터를 추가하여 삽입하는 변환부; 및

상기 분석부 및 변환부의 결과 데이터를 토대로 스마트 컨트랙트의 개인정보보호 검증을 수행하기 위한 V-컨트랙트를 생성하는 검증부를 포함하는 스마트 컨트랙트 프로그래밍 시스템.

청구항 2

제1항에 있어서,

상기 변환부는 원시 컨트랙트에 reveal문을 삽입할 위치를 파악하고, 상기 원시 컨트랙트의 지정된 위치에 reveal문을 삽입하여 변환된 B-컨트랙트를 생성하는 것을 특징으로 하는 스마트 컨트랙트 프로그래밍 시스템.

청구항 3

제1항에 있어서,

상기 검증부는 수탁자가 위탁자의 정보를 위임받아서 임의로 위탁자의 정보를 처리하는 경우 오류를 발생하도록 상기 V-컨트랙트를 생성하는 것을 특징으로 하는 스마트 컨트랙트 프로그래밍 시스템.

청구항 4

제3항에 있어서,

상기 V-컨트랙트는 수탁자가 위탁자의 정보를 위임받아서 스토리지(storage) 변수에 그대로 저장하거나 또는 자신의 공개키(public key)로 암호화하여 저장하는 경우 오류를 발생하는 것을 특징으로 하는 스마트 컨트랙트 프로그래밍 시스템.

청구항 5

제3항에 있어서,

상기 V-컨트랙트는 정보의 공개 대상과 공개 위치를 확인하여 수탁자가 위탁자의 정보를 위임받아서 다른 위탁자의 공개키(public key)로 암호화하여 저장하는 경우 오류가 발생하는 것을 특징으로 하는 스마트 컨트랙트 프로그래밍 시스템.

청구항 6

제3항에 있어서,

상기 V-컨트랙트는 수탁자가 위탁자의 정보를 위임받아서 위탁자의 정보로부터 def-use 관계에 있는 값을 암호화하지 않고, 체인(chain)에서 메모리 변수나 스택 변수로 표현하여 실행하는 경우 오류가 발생하는 것을 특징으로 하는 스마트 컨트랙트 프로그래밍 시스템.

청구항 7

제3항에 있어서,

상기 V-컨트랙트는 수탁자가 위탁자의 정보를 위임받아서 all로 공개된 것이 표현식 결과가 아니라 단일 데이터 자체일 경우 체인에 있는 스마트 컨트랙트를 수행하는 노드들에 공개되지 않도록 오류를 발생하는 것을 특징으로 하는 스마트 컨트랙트 프로그래밍 시스템.

청구항 8

제1항에 있어서,

상기 P-컨트랙트는 각 주체의 역할(role)과 기능적 단위별 허용 가능한 caller가 될 수 있는 역할 집합을 명시하고, 상기 B-컨트랙트의 분석을 통해 기능적 단위에 대해 어떤 위임이 존재하는지를 추출하여 나타내는 것을 특징으로 하는 스마트 컨트랙트 프로그래밍 시스템.

청구항 9

제8항에 있어서,

상기 검증부는 역할(role)이 무엇인지에 따라 함수 호출을 허용할지에 대한 modifier를 주입하고, 상기 modifier가 주입되는 경우 B-컨트랙트 상에 모순(conflict)이 일어나는 부분이 있는지 검증을 수행하는 것을 특징으로 하는 스마트 컨트랙트 프로그래밍 시스템.

청구항 10

기능 중심의 개인정보 공개 및 위탁방법을 제공하는 FLL(Function level leak) 컴파일러를 이용하여 스마트 컨트랙트의 개인정보보호를 위한 정보 공개 및 위임과정에 대해 지원하는 스마트 컨트랙트 프로그래밍 방법에 있어서,

개발자 또는 수탁자가 위탁자로부터 제공되는 스마트 컨트랙트의 원래 계약 내용을 처리하는 B-컨트랙트(business logic contract)를 작성하는 단계(S10);

상기 개발자 또는 수탁자가 스마트 컨트랙트 위탁자의 개인정보 처리를 위한 계약 내용을 포함하는 P-컨트랙트(privacy policy contract)의 역할(role)과 함수에 허용된 역할부분을 작성하는 단계(S20);

상기 FLL 컴파일러가 B-컨트랙트 및 P-컨트랙트를 분석 변환하여 스마트 컨트랙트를 생성하는 단계(S30);

스마트 컨트랙트의 위탁자가 생성된 스마트 컨트랙트를 확인하고, 특정 함수를 호출하여 계약을 체결하는 단계(S40); 및

상기 FLL 컴파일러가 계약이 체결된 스마트 컨트랙트의 개인정보보호를 검증하기 위한 V-컨트랙트를 생성하여 컨트랙트의 검증을 수행하는 단계(S50)를 포함하는 스마트 컨트랙트 프로그래밍 방법.

청구항 11

제10항에 있어서,

상기 스마트 컨트랙트를 생성하는 단계(S30)는

상기 FLL 컴파일러가 B-컨트랙트를 분석하여 개인정보의 공개여부가 함수단위로 정의되도록 원시 컨트랙트로 변환하고, 생성된 원시 컨트랙트의 지정된 위치에 reveal을 삽입하며, 상기 P-컨트랙트에 위탁자로부터 제공되는 공개 데이터를 추가 삽입하여 스마트 컨트랙트를 생성하는 것을 특징으로 하는 스마트 컨트랙트 프로그래밍 방법.

청구항 12

제10항에 있어서,

상기 특정 함수를 호출하여 계약을 체결하는 단계(S40)에서

상기 위탁자는 위탁자의 개인정보가 수탁자 외에는 외부로 알려지지 않도록 암호화하여 제공하는 것을 특징으로 하는 스마트 컨트랙트 프로그래밍 방법.

청구항 13

제10항에 있어서,

상기 컨트랙트의 검증을 수행하는 단계(S50)는

수탁자가 위탁자의 정보를 위임받아서 이후에 스토리지(storage) 변수에 그대로 저장하거나 또는 자신의 공개키(public key)로 암호화하여 저장하는 경우 오류를 발생하는 제1 단계와, 정보의 공개 대상과 공개 위치를 확인하여 수탁자가 위탁자의 정보를 위임받아서 다른 위탁자의 공개키(public key)로 암호화하여 저장하는 경우 오류가 발생하는 제2 단계와, 수탁자가 위탁자의 정보를 위임받아서 위탁자의 정보로부터 def-use 관계에 있는 값을 수탁자가 암호화하지 않고 체인(chain)에서 메모리 변수나 스택 변수로 표현하여 실행하는 경우 오류가 발생하는 제3 단계와, 수탁자가 위탁자의 정보를 위임받아서 all로 공개된 것이 표현식 결과가 아니라 단일 데이터 자체일 경우 오류를 발생하는 제4 단계 중 적어도 하나를 포함하는 스마트 컨트랙트 프로그래밍 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 개인정보보호를 지원하는 스마트 컨트랙트 프로그래밍 시스템 및 방법에 관한 것으로서, 더욱 상세하게는 스마트 컨트랙트의 개인정보보호를 위한 정보 공개 및 위임과정에 대해 지원하는 개인정보보호를 지원하는 스마트 컨트랙트 프로그래밍 시스템 및 방법에 관한 것이다.

배 경 기 술

[0003] 최근 블록체인을 기반으로 하는 스마트 컨트랙트(Smart contract) 기술이 주목받고 있다. 스마트 컨트랙트(Smart contract)란 블록체인을 기반으로 금융거래, 부동산 계약, 공증 등 다양한 형태의 계약을 체결하고 이행하는 것을 말하는 것으로 1994년 제안되었다.

[0004] 이후 2015년에는 스마트 컨트랙트를 구현하기 위한 분산 컴퓨팅 플랫폼이자 운영체제인 이더리움(Ethereum)이 등장했다. 실제 구현된 이더리움에는 제안된 스마트 컨트랙트의 이상과 달리 여러 취약점이 존재한다. 그 때문에 DAO 해킹 사건, Parity 사고 등 금융피해들이 발생했다.

[0005] 스마트 컨트랙트의 취약점은 경제적인 부분과 직접적인 연관이 있으므로 이에 대한 대응은 필수적이다. 이에 따라 이를 위한 Solidity 등의 소스코드 레벨의 취약점 검증기들이 개발되고 있으며, 배포 전후 적용 가능한 바이트 코드 레벨의 취약점 도구도 등장하고 있다. 현재까지 이러한 도구들은 주로 개발단계에서 취약점을 검출하는 데에 초점을 두고 있어, 안전하지 않은 코드의 생산을 막는 데에 도움을 준다.

[0006] 그러나 블록체인에서 공유되는 프로그램과 데이터는 모두 공개를 원칙으로 하는 단점으로 인해 개인정보보호 문제가 대두 되고 있다. 따라서, 본 발명은 개인정보보호를 지원하는 새로운 스마트 컨트랙트 프로그래밍 언어를 제안하고자 한다.

선행기술문헌

특허문헌

[0008] (특허문헌 0001) 대한민국 등록특허 제10-2037124호(2019년 10월 29일 공고)

발명의 내용

해결하려는 과제

[0009] 본 발명이 이루고자 하는 기술적 과제는 종래의 단점을 해결한 것으로서, 민감한 개인정보를 보호할 수 있도록 지원하는 스마트 컨트랙트 프로그래밍언어 및 시스템을 제공하고자 하는데 그 목적이 있다. 또한, 스마트 컨트랙트의 안전성을 해치지 않으면서 개발자의 편리한 개발이 가능한 스마트 컨트랙트 프로그래밍 시스템 및 방법을 제공하고자 하는데 그 목적이 있다.

[0010] 또한, 스마트 컨트랙트의 수탁된 주체가 목적과 데이터범위 및 활용시간을 기술할 수 있는 기술 방법을 제시하고자 하는데 그 목적이 있다. 또한, 개인정보보호 동의에 대한 별도의 계약으로도 읽힐 수 있도록 계약 당사자들에게 가독성을 증대하고자 하는데 그 목적이 있다.

과제의 해결 수단

[0012] 이러한 기술적 과제를 이루기 위한 본 발명의 일 측면에 따른 개인정보보호를 지원하는 스마트 컨트랙트 프로그래밍 시스템은 개인정보보호를 위한 정보 공개 및 위임과정에 대해 지원하기 위해 기능 중심의 개인정보 공개 및 위탁방법을 제공하는 FLL(Function level leak) 컴파일러를 포함한다. 또한, 상기 FLL 컴파일러는 분석부, 변환부 및 검증부를 포함한다.

[0013] 또한, 상기 분석부는 위탁자로부터 제공되는 스마트 컨트랙트의 원래 계약 내용을 처리하는 B-컨트랙트(business logic contract)와, 상기 위탁자의 개인정보 처리를 위한 계약 내용을 포함하는 P-컨트랙트(privacy policy contract)를 분석한다.

[0014] 상기 변환부는 분석부의 분석 결과를 토대로 개인정보의 공개여부가 함수단위로 정의되도록 B-컨트랙트를 원시 컨트랙트로 변환하고, 상기 P-컨트랙트에 위탁자로부터 제공되는 공개 데이터를 추가하여 삽입한다. 또한, 상기 검증부는 분석부 및 변환부의 결과 데이터를 토대로 스마트 컨트랙트의 개인정보보호 검증을 수행하기 위한 V-컨트랙트를 생성한다.

[0015] 바람직하게는, 상기 변환부는 원시 컨트랙트에 reveal문을 삽입할 위치를 파악하고, 상기 원시 컨트랙트의 지정된 위치에 reveal문을 삽입하여 변환된 B-컨트랙트를 생성한다. 바람직하게는, 상기 검증부는 수탁자가 위탁자의 정보를 위임받아서 임의로 위탁자의 정보를 처리하는 경우 오류를 발생하도록 상기 V-컨트랙트를 생성한다.

[0016] 바람직하게는, 상기 V-컨트랙트는 수탁자가 위탁자의 정보를 위임받아서 스토리지(storage) 변수에 그대로 저장하거나 또는 자신의 공개키(public key)로 암호화하여 저장하는 경우 오류를 발생한다.

[0017] 이때, 상기 V-컨트랙트는 정보의 공개 대상과 공개 위치를 확인하여 수탁자가 위탁자의 정보를 위임받아서 다른 위탁자의 공개키(public key)로 암호화하여 저장하는 경우 오류가 발생한다. 또한, 상기 V-컨트랙트는 수탁자가 위탁자의 정보를 위임받아서 위탁자의 정보로부터 def-use 관계에 있는 값을 암호화하지 않고, 체인(chain)에서 메모리 변수나 스택 변수로 표현하여 실행하는 경우 오류가 발생한다.

[0018] 또한, 상기 V-컨트랙트는 수탁자가 위탁자의 정보를 위임받아서 all로 공개된 것이 표현식 결과가 아니라 단일 데이터 자체일 경우 체인에 있는 스마트 컨트랙트를 수행하는 노드들에 공개되지 않도록 오류를 발생한다. 또한, 상기 P-컨트랙트는 각 주체의 역할(role)과 기능적 단위별 허용 가능한 caller가 될 수 있는 역할 집합을 명시하고, 상기 B-컨트랙트의 분석을 통해 기능적 단위에 대해 어떤 위임이 존재하는지를 추출하여 나타낸다.

[0019] 바람직하게는, 상기 검증부는 역할(role)이 무엇인지에 따라 함수 호출을 허용할지에 대한 modifier를

주입하고, 상기 modifier가 주입되는 경우 B-컨트랙트 상에 모순(conflict)이 일어나는 부분이 있는지 검증을 수행한다.

[0020] 또한, 본 발명의 다른 측면에 따른 개인정보보호를 지원하는 스마트 컨트랙트 프로그래밍 방법은 개발자 또는 수탁자가 위탁자로부터 제공되는 스마트 컨트랙트의 원래 계약 내용을 처리하는 B-컨트랙트(business logic contract)를 작성하는 단계와, 상기 개발자 또는 수탁자가 스마트 컨트랙트 위탁자의 개인정보 처리를 위한 계약 내용을 포함하는 P-컨트랙트(privacy policy contract)의 역할(role)과 함수에 허용된 역할부분을 작성하는 단계를 포함한다.

[0021] 또한, 본 발명은 상기 FLL 컴파일러가 B-컨트랙트 및 P-컨트랙트를 분석 변환하여 스마트 컨트랙트를 생성하는 단계와, 스마트 컨트랙트의 위탁자가 생성된 스마트 컨트랙트를 확인하고, 특정 함수를 호출하여 계약을 체결하는 단계 및 상기 FLL 컴파일러가 계약이 체결된 스마트 컨트랙트의 개인정보보호를 검증하기 위한 V-컨트랙트를 생성하여 컨트랙트의 검증을 수행하는 단계를 포함한다.

[0022] 바람직하게는, 상기 스마트 컨트랙트를 생성하는 단계는 상기 FLL 컴파일러가 B-컨트랙트를 분석하여 개인정보의 공개여부가 함수단위로 정의되도록 원시 컨트랙트로 변환하고, 생성된 원시 컨트랙트의 지정된 위치에 reveal을 삽입하며, 상기 P-컨트랙트에 위탁자로부터 제공되는 공개 데이터를 추가 삽입하여 스마트 컨트랙트를 생성한다.

[0023] 바람직하게는, 상기 특정 함수를 호출하여 계약을 체결하는 단계에서 상기 위탁자는 위탁자의 개인정보가 수탁자 외에는 외부로 알려지지 않도록 암호화하여 제공한다.

[0024] 바람직하게는, 상기 컨트랙트의 검증을 수행하는 단계는 수탁자가 위탁자의 정보를 위임받아서 이후에 스토리지(storage) 변수에 그대로 저장하거나 또는 자신의 공개키(public key)로 암호화하여 저장하는 경우 오류를 발생하는 제1 단계와, 정보의 공개 대상과 공개 위치를 확인하여 수탁자가 위탁자의 정보를 위임받아서 다른 위탁자의 공개키(public key)로 암호화하여 저장하는 경우 오류가 발생하는 제2 단계와, 수탁자가 위탁자의 정보를 위임받아서 위탁자의 정보로부터 def-use 관계에 있는 값을 수탁자가 암호화하지 않고 체인(chain)에서 메모리 변수나 스택 변수로 표현하여 실행하는 경우 오류가 발생하는 제3 단계와, 수탁자가 위탁자의 정보를 위임받아서 all로 공개된 것이 표현식 결과가 아니라 단일 데이터 자체일 경우 오류를 발생하는 제4 단계 중 적어도 하나를 포함한다.

발명의 효과

[0026] 이상에서 설명한 바와 같이, 본 발명에 따른 개인정보보호를 지원하는 스마트 컨트랙트 프로그래밍 시스템 및 방법은 스마트 컨트랙트의 개인정보보호를 지원할 수 있는 효과가 있다. 또한, 스마트 컨트랙트의 안전성을 해치지 않으면서 개발자가 스마트 컨트랙트 프로그래밍을 편리하게 개발할 수 있는 효과가 있다.

[0027] 또한, 스마트 컨트랙트의 수탁된 주체가 목적과 데이터범위 및 활용시간을 효율적으로 기술할 수 있는 효과가 있다. 또한, 개인정보보호 동의에 대한 별도의 계약으로도 읽힐 수 있도록 계약 당사자들의 가독성을 증대할 수 있는 효과가 있다.

도면의 간단한 설명

[0029] 도 1은 본 발명의 실시 예에 따른 원시 컨트랙트를 나타내는 도면이다.

도 2는 본 발명의 실시 예에 따른 스마트 컨트랙트 프로그래밍 시스템을 나타내는 구성도이다.

도 3은 본 발명의 실시 예에 따른 FLL 컴파일러를 나타내는 구성도이다.

도 4 및 도 5는 본 발명의 실시 예에 따른 B-컨트랙트를 나타내는 도면이다.

도 6은 본 발명의 실시 예에 따른 P-컨트랙트를 나타내는 도면이다.

도 7은 본 발명의 실시 예에 따른 스마트 컨트랙트 프로그래밍 방법을 나타내는 순서도이다.

발명을 실시하기 위한 구체적인 내용

[0030] 아래에서는 첨부한 도면을 참고로 하여 본 발명의 실시 예에 대하여 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자가 용이하게 실시할 수 있도록 상세히 설명한다. 그러나 본 발명은 여러 가지 상이한 형태로 구현될 수 있으며 여기에서 설명하는 실시 예에 한정되지 않는다. 그리고 도면에서 본 발명을 명확하게 설명하기 위

해서 설명과 관계없는 부분은 생략하였으며, 명세서 전체를 통하여 유사한 부분에 대해서는 유사한 도면부호를 붙였다.

[0031] 명세서 전체에서, 어떤 부분이 어떤 구성요소를 "포함"한다고 할 때, 이는 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라 다른 구성요소를 더 포함할 수 있는 것을 의미한다. 또한, 명세서에 기재된 "…부", "…기", "…모듈" 등의 용어는 적어도 하나의 기능이나 동작을 처리하는 단위를 의미하며, 이는 하드웨어나 또는 소프트웨어 또는 하드웨어 및 소프트웨어의 결합으로 구현될 수 있다.

[0032] 이하, 첨부된 도면을 참조하여 본 발명의 바람직한 실시 예를 설명함으로써, 본 발명을 상세히 설명한다.

[0033] 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타낸다.

[0034] 도 1은 본 발명의 실시 예에 따른 원시 컨트랙트를 나타내는 도면이다. 여기에서, 상기 원시 컨트랙트는 기본적인 개인정보보호를 지원하는 언어로된 컨트랙트를 나타낸다. 예를 들어, 상기 원시 컨트랙트는 지케이(zkay) 컨트랙트가 될 수 있다. 즉, 도 1은 지케이(zkay)로 기술된 컨트랙트가 될 수 있다.

[0035] 조크라테스(Zokrates)는 블록체인의 로드를 줄이고 개인정보를 보호하는 방안을 제안하였고, 프로그램의 내용을 증명으로 전환하는 것을 편리하게 도와주는 새로운 도메인 언어(DSL)를 개발하였다. 그러나 상기 도메인 언어(DSL)는 종래의 이더리움 언어인 솔리디티(Solidity)로서 스마트 컨트랙트(200)를 제작하는 개발자가 사용하기 위해서는 증명 제작과정을 포함한 개인정보보호 프로토콜을 알아야 하는 부담이 있다.

[0036] 이에 따라 개발된 것이 ETH 취리히 대학의 지케이(zkay)이다. 상기 지케이는 솔리디티를 확장하여 개인정보로 보호하고, 취급되어야 하는 정보들에 대해 변수나 인자 타입에 명시하며, 이에 따라 자동으로 스마트 컨트랙트(200)와 개인정보보호의 증명 및 검증 컨트랙트를 생성해주어 매우 편리한 이점이 있다.

[0037] 예를 들어 설명하면 다음과 같다. 도 1에 도시된 지케이(zkay) 스마트 컨트랙트는 솔리디티(Solidity) 기본 타입인 uint, bool 등에 '@hospital', '@me', '@x'와 같이 소유자를 명시하도록 하고, 만일 소유자의 해당 값이 다른 변수에 직간접적으로 지정되면 오류를 내는 방식으로 개인정보보호를 지원한다.

[0038] 이를 위해 기본적으로는 각 값을 영지식 증명(zero knowledge proof) 등을 통한 암복호화를 통해 개인 정보 노출을 막고 있으며, 암호화된 값을 오프체인(off-chain)에서 계산하기 위한 런타임 프레임워크도 함께 제시하고 있다.

[0039] 또한, 도 1에서 나타나는 지케이(zkay)의 reveal문은 정보의 소유자가 지정한 다른 주체에게 정보를 명시적으로 공개하는 것을 의미한다. 즉, 도 1의 첫 번째 reveal은 소유자가 me인 r을 don이라는 소유자에게 공개함을 의미한다.

[0040] 여기서 me는 record 함수를 호출한 caller를 의미하며, 상기 caller는 병원(hospital)을 나타낸다. 즉, 함수 record는 병원에서 기처리한 정보를 don의 개인정보로 등록하여 don만 열람할 수 있도록 만드는 과정을 의미한다.

[0041] 또한, 도 1의 두 번째 reveal문은 me의 개인정보로 risk[me]와 r 두 가지가 같은지를 확인하기 위해 그 결과를 모두에게 공개하는 것을 나타낸다. 여기서 공개되는 것은 ==로 표현된 비교식의 참거짓 값이며, 계산은 상기 caller인 me의 컴퓨터, 즉 오프체인(off-chain)으로 실행되고 실제 개인정보는 감추면서도 정확히 필요한 정보를 드러내는 효과가 있다.

[0042] 그러나 이러한 개인정보 위임에 대한 표현은 아래와 같은 단점이 있다.

[0043] (1) 개발자의 입장에서 모든 개인정보가 공개되어야 하는 곳마다 정보 소유자 및 공개 대상의 명시가 필요하다. 이러한 방식으로 진행하는 경우 안전성이 높아진다는 장점은 있으나 매 표현식마다 이러한 과정을 진행해야 하는 개발자의 부담이 있으며 오류의 위험도 증가된다.

[0044] (2) 도 1의 첫 번째 reveal문의 예에서 나타난 바와 같이 개인 정보의 공개는 정보의 처리를 위탁하는 상황에서 발생되는 경우가 많다. 이러한 상황에서는 개인정보 공개의 동의를 다루는 별도의 계약(동의서)이 필요한데, 정부의 표준 개인정보처리위탁 계약서 약식과 유럽의 개인정보 처리법인 GDPR에도 나타나는 바와 같이 개인정보 활용의 목적과 범위, 보유 허용시기가 명확해야 한다.

[0045] 이와 같이 종래에는 위탁-수탁 과정에서 스마트 컨트랙트(200)의 개인정보보호를 위한 정보공개 및 위임과정에 대한 지원이 부족하다. 따라서, 본 발명은 스마트 컨트랙트(200)의 개인정보보호를 위한 정보 공개 및 위임과정

에 대해 지원하는 FLL(Function level leak) 관리시스템을 제안하고자 한다.

[0046] 도 2는 본 발명의 실시 예에 따른 스마트 컨트랙트 프로그래밍 시스템(10)을 나타내는 구성도이다. 본 발명의 실시 예에 따른 스마트 컨트랙트 프로그래밍 시스템(10)은 기능 중심의 개인정보 공개 및 위탁방법을 제공하는 FLL(Function level leak) 컴파일러(100)를 이용하여 스마트 컨트랙트(200)의 개인정보보호를 위한 정보 공개 및 위임과정에 대해 지원한다.

[0047] 본 발명은 상기 도 1에서 도시된 병원 예와 같이 빈번히 발생되는 특정 업무 처리에 대한 위탁과 이에 따른 개인정보보호에 관련된 경우에 집중하여 아래의 시나리오를 가정한다.

[0048] 즉, 수탁자가 스마트 컨트랙트(200)를 생성한다. 또한, 위탁자는 해당 컨트랙트(200)를 확인하고 특정 함수를 호출함으로써 계약에 자동 동의하여 계약을 체결한다. 이때, 상기 위탁자는 개인 정보보호에 관련된 계약도 확인하며 동의한다.

[0049] 또한, 상기 위탁자의 개인정보는 수탁자 외에는 외부로 알려지지 않도록 수탁자의 공개키 등으로 암호화된다. 또한, 정보 공개 건이 있을 때, 수탁자가 위탁자에게 공개하는 경우에만 스토리지(storage)에 저장이 허용된다. 위탁자가 수탁자에게 정보를 공개했더라도 수탁자는 해당 정보를 그대로 스토리지에 저장할 수 없다. 즉, 결과적으로 위탁자의 공개키로 암호화된 정보 외에는 수탁자가 임의로 위탁자의 정보를 블록체인에 남길 수 없어야 한다.

[0050] 본 발명은 기능 중심의 개인정보 공개 및 위탁방법을 제공하는 FLL(Function level leak) 컴파일러(100)를 통해 상기 시나리오의 목적을 달성할 수 있다. 이를 위해, 도 2와 같이 개인정보를 처리하는 모든 스마트 컨트랙트(200)는 3가지 부(Sub) 컨트랙트(210, 220, 230)가 작성된다.

[0051] 즉, 상기 부(Sub) 컨트랙트는 스마트 컨트랙트(200)의 원래 계약 내용을 처리하는 B-컨트랙트(business logic contract)(210)와, 개인정보 처리를 위한 계약 내용을 처리하는 P-컨트랙트(privacy policy contract)(220)와, B-컨트랙트(210) 및 P-컨트랙트(220)를 이용하여 개인 컴퓨터에서 실행된 계산 결과가 맞는지 여부를 증명하는 검증을 수행하는 V-컨트랙트(검증 컨트랙트, Verifier contract)(230)를 포함한다.

[0052] 이때, P-컨트랙트(220)는 지케이(zkay) 또는 솔리디티(Solidity)로 된 스마트 컨트랙트(200)와 유사한 형태로 이루어질 수 있다. 또한, 스마트 컨트랙트(200)는 B-컨트랙트(210) 및 P-컨트랙트(220)와, 자동 생성되는 V-컨트랙트(230)를 합하여 전체 기능을 수행한다. 즉, 스마트 컨트랙트(200)와 B-컨트랙트(210), P-컨트랙트(220) 및 V-컨트랙트(230)의 관계는 아래와 같이 나타낼 수 있다.

[0053] (S = < B, P, V >)

[0054] 여기에서, S는 스마트 컨트랙트(200), B는 B-컨트랙트(210), P는 P-컨트랙트(220), V는 V-컨트랙트(230)를 나타낸다.

[0055] V-컨트랙트(230)는 지케이(zkay) 또는 조크라테스(zokrates) 등의 원시적인 개인정보보호 지원 기능을 통해서도 제공되는 것으로, 개인 컴퓨터에서 실행되는 계산 결과가 맞음을 증명하는 증명을 검증한다. 이때, V-컨트랙트(230)는 FLL 컴파일러(100)를 통해 자동 생성되고, P-컨트랙트(220)의 일부도 자동 생성될 수 있다.

[0056] 도 3은 본 발명의 실시 예에 따른 FLL 컴파일러(100)를 나타내는 구성도이고, 도 4 및 도 5는 본 발명의 실시 예에 따른 B-컨트랙트(211)를 나타내는 도면이다. 즉, 도 4 및 도 5는 본 발명의 실시 예에 따라 지케이 컨트랙트(원시 컨트랙트)를 토대로 FLL 컴파일러(100)에서 제작된 B-컨트랙트(211)를 나타내는 도면이다.

[0057] 이때, B-컨트랙트(211)는 FLL 컴파일러(100)가 입력받은 B-컨트랙트(210)를 토대로 변환하여 생성되는 컨트랙트를 나타낸다(도 2 및 도 3에서 B' contract(211)). 또한, P-컨트랙트(221)는 FLL 컴파일러(100)가 입력받은 P-컨트랙트(220)를 토대로 변환하여 생성되는 컨트랙트를 나타낸다(도 2 및 도 3에서 P' contract(221)).

[0058] 본 발명의 실시 예에 따른 FLL 컴파일러(100)는 변환부(110), 분석부(120) 및 검증부(130)를 포함한다. 분석부(120)는 위탁자로부터 제공되는 스마트 컨트랙트(200)의 원래 계약 내용을 처리하는 B-컨트랙트(business logic contract)(210)와, 상기 위탁자의 개인정보 처리를 위한 계약 내용을 포함하는 P-컨트랙트(privacy policy contract)(220)를 분석한다.

[0059] 변환부(110)는 분석부(120)의 분석 결과를 토대로 B-컨트랙트(210)를 원시 컨트랙트(예를 들어, 지케이(zkay) 컨트랙트)로 변환하고, reveal문을 삽입할 위치를 파악하여 상기 원시 컨트랙트의 지정된 위치에 reveal문을 삽입하며, P-컨트랙트(220)에 위탁자로부터 제공되는 공개 데이터를 추가하여 삽입한다. 또한, 검증부(130)는 분

석부(120) 및 변환부(110)의 결과 데이터를 토대로 스마트 컨트랙트(200)에 대한 개인정보보호의 검증을 수행하기 위한 V-컨트랙트(230)를 생성한다.

[0060] FLL 컴파일러(100)를 통해 생성된 B-컨트랙트(211)는 지케이(zkay)와 유사하나, 개발자는 함수단위로 정보유출(leak) 여부를 명시하게 된다. 예를 들어, 도 1에 나타난 지케이의 첫 번째 reveal문의 경우는 도 4와 같이 변경될 수 있다. 즉, 개발자는 함수의 인자로 받는 r에 대해 타입을 bool@me에서 bool@me>don으로 변경함으로써 don으로의 공개가 일어남을 명시한다.

[0061] 도 4에서 나타나는 바와 같이 프로그램상에서는 risk[don]에 지정하는 곳에서 reveal문이 사라졌음을 확인할 수 있다. 이를 위해 FLL 컴파일러(100)는 자동으로 reveal문을 추가하게 된다.

[0062] FLL 컴파일러(100)는 B-컨트랙트(210)에서 지케이 컨트랙트 등의 원시 컨트랙트로의 변환을 담당하며, 타입과 프로그램 구문을 분석하여 reveal문을 삽입할 위치를 파악하고, 삽입하는 역할을 담당한다. 이를 통해 개인정보의 공개여부가 함수단위로 정의된다.

[0063] 상기 함수단위의 공개는 기능적인 최소단위가 함수이므로 공개 목적에 따른 공개라는 본 발명의 목적과 부합한다. 또한, 스마트 컨트랙트들의 길이가 일반적으로 10줄 이내로 짧으므로 해당 부분으로 묶어 공개를 명시해도 안전성을 해치지 않는다.

[0064] 또한, 자동으로 분석하여 reveal문을 추가하므로 신뢰성이 증가하고, 개발자의 명시가 함수 단위로 드러나므로 특정 목적(function)에 대해 데이터의 위임이 일어나기 때문에 종래의 개인정보보호와 비교한다면 안전성을 해치지 않는다.

[0065] 이때, 정보 공개 주체는 sender 즉, 함수의 caller인 me가 되므로 정보 소유자가 me인 정보에 대해서만 >를 정의할 수 있다. 도 1의 두 번째 reveal문에서는 공개 범위가 하나의 데이터가 아니라 보호대상 데이터를 포함하는 식이다. 이러한 경우 지케이 등 일반적인 개인정보보호 기능을 가지는 스마트 컨트랙트(200)에서는, 식의 결과값만을 노출하게 되며 검증 대상이 된다.

[0066] FLL 컴파일러(100)에서는 데이터 노출 범위를 표현하는 것이 중요하다. 이를 그대로 FLL(100)로 변형하는 경우, 도 4에서 risk[don] = r+1과 같은 식이 있다면 결과가 risk[don]에 지정되며 결과만 위임된다.

[0067] 프로그램 분석을 통하여 r을 포함하는 모든 식을 확인하여 그 중 동일 소유권 내에서 수행 가능한 적절한 범위의 식을 오프체인(off chain) 식으로 수행하는데, 가장 큰 범위의 식이 블록체인의 부하를 가장 줄일 수 있어 제일 선호된다.

[0068] 지정문 외에 데이터 범위의 또 다른 경계는 문장(statement) 구조 요소 즉, require 나 함수 호출 등의 인자, if 구조문의 조건 등이 있다. 도 4에서 r의 타입인 bool@me>all은 r 자체가 아니라 r을 포함하는 식이 all을 대상으로 공개될 것임을 의미한다.

[0069] 도 4의 예에서는 r을 포함하고 데이터 범위 한계 내의 가장 큰 표현식인 r == risk[me]에 대해, FLL 컴파일러(100)는 타입 시스템 및 분석으로부터 이러한 단위를 식별하고 해당 위치에 reveal문을 주입한다.

[0070] 만약 check에서 공개 대상 데이터가 인자로 존재하지 않고, require(p == risk[me])와 같은 식이 존재한다면(이때, p는 public 값), 공개 및 위임에 대한 아무런 정보가 없으므로 오류가 발생된다.

[0071] 상기 오류를 없애기 위해서는 > all risk[me]; 과 같이 >all을 명시적으로 포함하는 선언을 함수 시작 부분에서 한 번 더 해준다. 예를 들면, 아래와 같이 작성될 수 있다.

[0072]
```
function check( )
```

[0073]
```
        with risk[address! x]@x > all {
```

[0074]
```
                require(p == risk[me]);
```

[0075]
```
}
```

[0076] 또는 명시적으로 reveal과 유사하게 아래와 같이 명시할 수 있다. 이때는 지케이처럼 직접 reveal 함수를 호출하지 않고 키워드와 기호를 통한 문법적인 명시를 함으로써 분석에 도움을 주고 변수 선언과의 일관성을 유지한다.

[0077]
```
function check( ) {
```

[0078] ...

[0079] require((p == risk[me])> all);

[0080] }

[0081] FLL 컴파일러(100)를 사용하면 상기 도 1의 B-컨트랙트(210)는 전체적으로 도 5와 같이 나타낼 수 있다.

[0082] 이때, 도 5의 @creator로 명시된 부분은 컨트랙트를 생성한 사람이 외부에 드러내지 않고 보유하는 데이터를 나타낸다. 또한, 도 5의 !x나 x로 표현되는 부분은 종래의 지케이와 동일하나, !x@donner와 같이 x의 범위를 나타내는 부분이 추가된다. 의미상으로 risk는 mapping이고, key에 나타난 x는 donner 역할이며 value의 소유자이다.

[0083] 도 6은 본 발명의 실시 예에 따른 P-컨트랙트(221)를 나타내는 도면이다. FLL(100)에서의 정보처리 목적이 되는 기본 단위는 함수(function)인데, 단일 함수가 아니라 루트에 해당하는 external 함수로부터 호출되는 재귀적 호출이 없는 해당 컨트랙트 내의 call tree를 나타낸다.

[0084] 이러한 루트 함수와 call tree의 쌍을 기능적 단위(functional unit)라고 한다. 모든 external 함수는 기능적 단위의 루트함수가 될 수 있다. 하나의 함수가 복수의 call tree에 속할 수도 있고, 하나의 call tree의 루트 함수가 경우에 따라서는 다른 call tree의 일반 함수로 포함될 수 있다.

[0085] P-컨트랙트(221)는 아래의 두 부분으로 구성될 수 있다.

[0086] 하나는, 개발자가 명시하는 부분으로서, 역할(role)과 기능적 단위별 허용 가능한 caller가 될 수 있는 역할 집합을 명시한다. 두 번째는, B-컨트랙트(210)를 분석해서 추출되는 부분으로서, 기능적 단위에 대해 어떤 위임이 존재하는지를 표현하는 부분을 나타낸다. 즉, 분석부(120)가 B-컨트랙트(210)를 분석하여 기능적 단위에 대해 어떤 위임이 존재하는지를 추출하고, P-컨트랙트(221)에 명시한다.

[0087] 상기 역할(role)에는 기본적으로 컨트랙트 생성자(creator)가 있고, 변경이 불가능하다. 이는 수탁자를 의미하고, 모든 주체의 역할이 대칭적인 지케이와 다르다. External 함수마다 호출을 할 수 있는 역할들이 정해져 있다. 예를 들어, 컨트랙트의 constructor를 호출할 수 있는 역할은 컨트랙트의 생성자뿐이다. 그 밖의 역할(위탁자 및 기타)은 개발자가 P-컨트랙트(220)에 정의한다.

[0088] 외부에서 특정 루트 함수를 호출하면, 함수에 허용된 역할에 따라 호출자의 역할이 고정된다. 예를 들어, external 함수 f()를 호출할 수 있는 주체가 "@donner"라고 한다면, 상기 f()를 호출한 caller는 이후 @donner의 역할로 기능적 단위를 수행하게 된다(이때, 역할 명은 @를 앞에 붙여 명시한다).

[0089] 허용된 역할이 두 개 이상인 경우 중간에 모순이 발생하지 않는 역할로 고정된다(call path 별로 상이할 수 있으므로, 동적으로 부여할 수 있다). 루트함수와 호출 가능한 역할은 n:m 관계이고, 실제 어떤 컨트랙트/사람으로부터 역할과의 관계도 n:m 관계이다.

[0090] 예를 들어, 동일한 사람이 서로 다른 역할로 해당 컨트랙트(200)의 다른(또는 동일한) 함수를 호출할 수도 있다. P-컨트랙트(220)의 이러한 명시를 보고 런타임에 B-컨트랙트(210)를 점검, 코드 주입하는 부분이 B-컨트랙트(211)에 추가된다.

[0091] 즉, 상기 역할(role)이 무엇인지에 따라 함수 호출을 허용할지에 대한 modifier를 주입하고, 상기 modifier가 주입되었다고 가정하였을 때 현재 B-컨트랙트(211) 상에 conflict가 일어나는 부분이 있는지 검증을 하게 된다. 또한, 이러한 검증 단계를 제1 P-검증이라고 명명한다.

[0092] 상기 제1 P-검증은 정보흐름을 포함한 정적 분석, 검증을 통하여 해당 역할만 호출하고 있는지를 보고 검증 코드를 주입한다. 또한, 모순이 발생할 가능성이 있는지 정적으로 검사하고 필요시 컴파일 오류를 발생시키며, 동적으로 역할을 배정하고 검사하는 코드를 주입한다.

[0093] 또한, 제2 P-검증은 수탁자가 위탁자의 정보를 위임받아서 이후에 스토리지(storage) 변수에 그대로 또는 자신의 공개키(public key)로 암호화하여 저장하면 오류를 발생하는 제1 검증 단계와, 다른 위탁자의 공개키(public key)로 암호화하여 저장해도 오류가 발생하는 제2 검증 단계(이때, 공개 대상과 공개 위치를 확인한다)를 포함한다.

[0094] 또한, 상기 제2 P-검증은 위탁자의 정보로부터 def-use 관계에 있는 값을 수탁자가 암호화하지 않고 체인(chain)에서 메모리 변수나 스택 변수 등으로 표현하여 실행하면 오류가 발생하는 제3 검증 단계와, all로 공개

된 것이 표현식 결과가 아니라 단일 데이터 자체일 경우에도 체인에 있는 스마트 컨트랙트(200)를 수행하는 노드들에 공개되는 경우이므로 오류를 발생하는 제4 검증 단계를 포함한다.

[0095] B-컨트랙트(211)에서 함수 시그니처에 명시된 공개 관계를 확인하고 적절한 분석을 추가하면, 각 기능적 단위별로 어떠한 정보를 누구에게 공개하는지에 대한 정보를 추출 가능하다. 이때, 공개 가능한 모든 데이터(D-집합)의 원소는 <루트 함수명, 공개 범위, 공개 대상(, 지정문일 경우 공개 위치)>으로 표현된다.

[0096] 이 데이터는 루트 함수의 caller가 가진 개인 데이터가 해당 기능적 단위 수행 중에 다른 소유자에 일시적으로 공개되는 것을 요약해서 보여준다. 여기서 공개 범위, 공개 대상에서는 해당 함수 내의 scope에 나타나는 변수들과 솔리디티 산출, 논리, 비교 표현식으로 표현된다. Storage 상에 지정문을 통해 전달된 경우는 대상 스토리지(storage) 이름을 추가로 명시한다.

[0097] 수탁자의 경우 위탁자로 공개하고, 위탁자의 경우 수탁자로 공개가 아니라면 오류가 발생한다. 특히, from_role이 @creater(즉, 수탁자)일 경우에 공개 대상이 @donner(즉, 위탁자)가 아니면 오류인데, 이것이 storage 변수일 때는 더욱 중요하다. 이때, 수탁자가 storage 변수에 위탁자의 정보를 자신의 공개키(public key)로 암호화하여 넣는다고 하더라도, 나중에 열람할 수 있기 때문에 방지되어야 한다(제2 P-검증의 제1 검증 단계 및 제4 검증 단계).

[0098] from_role이 @creator인데 공개 대상과 공개 위치가 다를 때, 즉 !x와 @x로 표현되지 않고 상이한지 확인하여 만일 상이하다면 오류를 발생시킨다(제2 P-검증의 제4 검증 단계). 또한, 기타 all로 공개된 것이 표현식 결과가 아니라 단일 데이터 자체일 경우도 체인에 있는 스마트 컨트랙트(200)를 수행하는 노드들에 공개되는 경우이므로 오류를 발생한다(제2 P-검증의 제3 검증 단계).

[0099] P-컨트랙트(221)는 결과적으로 B-컨트랙트(211)를 제어할 수 있는 일반 컨트랙트로 변환된다. 개발자는 B-컨트랙트(210)를 작성하고 P-컨트랙트(220)의 역할, 함수에 허용된 역할 부분을 작성한다. 이때, 아래의 두 단계를 통해 FLL 컴파일러(100)는 B-컨트랙트(210)를 분석 변환한다.

[0100] 먼저, FLL 컴파일러(100)는 B-컨트랙트(210)를 원시 컨트랙트(예를 들어, 지케이 컨트랙트)로 변환하는데, 생성되는 원시 컨트랙트의 지정된 위치에 reveal을 삽입한다(프로그램 분석을 동반). 또한, P-컨트랙트(220)의 D-집합(공개 데이터)을 채움으로써 타입 정보를 이용하여 추론하며 적절한 분석을 실행한다.

[0101] 이러한 분석 변환 과정이 완료되면, 상기 제1 P-검증 또는 제2 P-검증을 이용하여 검증을 실시한다. 이때, 검증의 전 과정은 정보 흐름을 통해 분석되며, 변환 단계부터 흐름 분석을 통해서 임시변수를 거친 use-def 관계를 고려하여 수행한다.

[0102] 도 7은 본 발명의 실시 예에 따른 스마트 컨트랙트 프로그래밍 방법을 나타내는 순서도이다. 발명의 실시 예에 따른 스마트 컨트랙트 프로그래밍 방법은 개발자 또는 수탁자가 위탁자로부터 제공되는 스마트 컨트랙트(200)의 원래 계약 내용을 처리하는 B-컨트랙트(business logic contract)(210)를 작성하는 단계(S10)와, 상기 개발자 또는 수탁자가 스마트 컨트랙트(200) 위탁자의 개인정보 처리를 위한 계약 내용을 포함하는 P-컨트랙트(privacy policy contract)(220)의 역할(role)과 함수에 허용된 역할부분을 작성하는 단계(S20)를 포함한다.

[0103] 또한, 본 발명은 FLL 컴파일러(100)가 B-컨트랙트(210) 및 P-컨트랙트(220)를 분석 변환하여 스마트 컨트랙트(200)를 생성하는 단계(S30)와, 스마트 컨트랙트(200)의 위탁자가 생성된 스마트 컨트랙트(200)를 확인하고, 특정 함수를 호출하여 계약을 체결하는 단계(S40) 및 FLL 컴파일러(100)가 계약이 체결된 스마트 컨트랙트(200)의 개인정보보호를 검증하기 위한 V-컨트랙트(230)를 생성하여 컨트랙트(200)의 검증을 수행하는 단계(S50)를 포함한다.

[0104] 이때, 상기 스마트 컨트랙트(200)를 생성하는 단계(S30)는 FLL 컴파일러(100)가 B-컨트랙트(210)를 분석하여 개인정보의 공개여부가 함수단위로 정의되도록 원시 컨트랙트(예를 들어, 지케이(zkay) 컨트랙트)로 변환하고, 생성된 원시 컨트랙트의 지정된 위치에 reveal을 삽입하며, P-컨트랙트(220)에 위탁자로부터 제공되는 공개 데이터를 추가 삽입하여 스마트 컨트랙트(200)를 생성하는 것을 특징으로 한다.

[0105] 또한, 상기 특정 함수를 호출하여 계약을 체결하는 단계(S40)는 위탁자가 위탁자의 개인정보가 수탁자 외에는 외부로 알려지지 않도록 암호화하여 제공하는 것을 특징으로 한다.

[0106] 또한, 상기 컨트랙트(200)의 검증을 수행하는 단계(S50)는 상기 제1 P-검증 또는 제2 P-검증을 포함한다. 즉, 상기 제1 P-검증과, 제1 검증 단계와, 제2 검증 단계와, 제3 검증 단계와, 제4 검증 단계 중 적어도 하나를 포

함한다.

[0107] 본 발명의 스마트 컨트랙트 프로그래밍 시스템(10) 및 방법을 통해 개인정보보호를 지원하는 과정의 예를 들어 검증하면 아래와 같다.

[0108] 일반적으로 개인정보보호의 가장 문제는 수탁자가 위탁자의 개인정보를 raw 데이터 형태로 열람할 수 있을 때이다. 이는 처리를 위해 필연적인 것으로 두 가지 경우가 있을 수 있고, 지케이 등에서는 암호학적으로 관리하여 개인정보보호를 지원한다.

[0109] 첫 번째로, 위탁자가 수탁자에게 개인정보를 공개한 경우, 수탁자는 오프체인(off chain)으로 언제든 직접 확인 가능하지만 수탁자에게만 공개되므로 체인 위에는 공개키(public key)의 암호화된 상태로 존재한다.

[0110] 이때의 문제는 수탁자의 노드에서는 원본을 볼 수 있다는 것이다. 따라서, 본 발명의 FLL(100)은 실수 또는 고의로 상기 데이터를 부적절하게 처리하는 것을 방지하고자 한다.

[0111] 먼저, 수탁자가 고의 또는 실수로, 공개 받은 위탁자의 정보를 체인에 저장하려 할 때, 위탁자가 원래 all로 공개된 경우에는 위탁자가 의도적으로 공개한 추상화된 정보이므로 공개에 문제가 없다. 그러나 위탁자가 수탁자에게만 공개된 경우에는 제2 P-검증의 제1 검증 단계, 제2 검증 단계 및 제3 검증 단계에 의해 오류를 발생하게 된다.

[0112] 다음으로, 수탁자가 고의 또는 실수로, 공개 받은 위탁자의 정보를 자신의 소유 변수로 이전 저장하려 할 때, 위탁자가 all로 공개된 경우에는 제2 P-검증의 제4 검증 단계에 의해, 표현식이므로 제2 P-검증의 제1 검증 단계와 같은 경우로 문제가 없다.

[0113] 또한, 위탁자의 정보가 수탁자에게만 공개된 경우에는 제2 P-검증의 제1 검증 단계, 제2 검증 단계 및 제3 검증 단계를 통해 개인정보의 방어가 가능하다.

[0114] 두 번째로, 수탁자가 처리 결과를 위탁자에게 넘겨주는(공개하는) 경우, 처리 결과는 위탁자 개인정보에 해당하므로 체인에 그대로 존재하면 안된다. 체인에는 수탁자의 공개키로 암호화된 상태로 존재하며, 위탁자에게 reveal 되는 순간, 암호학적으로 위탁자의 공개키로 암호화되는 처리가 수탁자의 노드에서 오프체인으로 일어난다.

[0115] 수탁자가 위탁자에게 공개하는 정보는 주로 처리 결과로서 위탁자의 공개키로 암호화되어 있는 개념이므로 위탁자만 확인 가능하다. 즉, raw 데이터가 수탁자의 손에 있다는 것이므로 결과적으로는 상기 첫 번째 경우와 차이가 없다.

[0116] 또한, 본 발명의 실시 예에 따른 개인정보보호를 지원하는 스마트 컨트랙트 프로그래밍 시스템(10)의 상기 제2 P-검증은 그 자체가 개인정보보호의 단계를 말해주도록 이용될 수 있다. 즉, 상기 제2 P-검증의 제4 검증 단계를 수탁자에게 공개된 데이터에도 적용하는 컨트랙트인지 검사하여 위탁자에게 알리고, 위탁자가 적절한 컨트랙트(수탁자)를 선택하게 할 수 있다.

[0117] 이상으로 본 발명에 관한 바람직한 실시예를 설명하였으나, 본 발명은 상기 실시예에 한정되지 아니하며, 본 발명의 실시예로부터 당해 발명이 속하는 기술분야에서 통상의 지식을 가진 자에 의한 용이하게 변경되어 균등하다고 인정되는 범위의 모든 변경을 포함한다.

부호의 설명

[0119] 10 : 스마트 컨트랙트 프로그래밍 시스템

100 : FLL 컴파일러 110 : 변환부

120 : 분석부 130 : 검증부

200 : 스마트 컨트랙트(smart contract)

210 : B-컨트랙트 211 : B'-컨트랙트

220 : P-컨트랙트 221 : P'-컨트랙트

230 : V-컨트랙트 300 : 지케이(zkay) 시스템

310 : B"-컨트랙트

도면

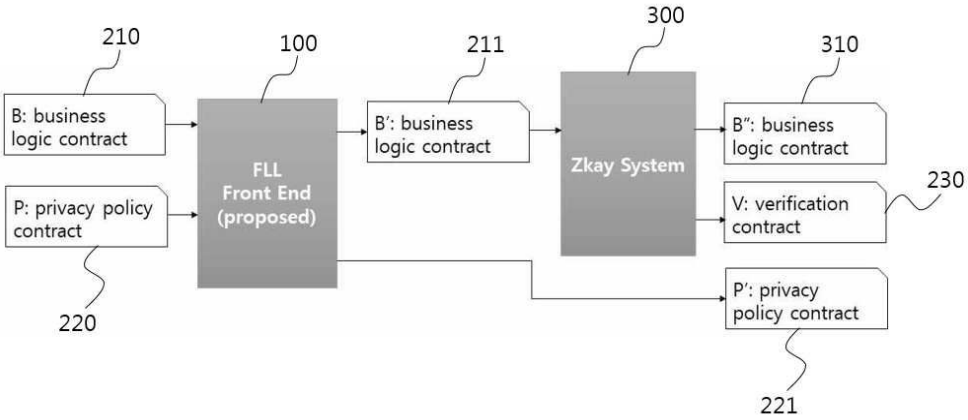
도면1

```
contract MedStats {
  final address hospital;
  uint@hospital count;
  mapping(address!x => bool@x) risk;

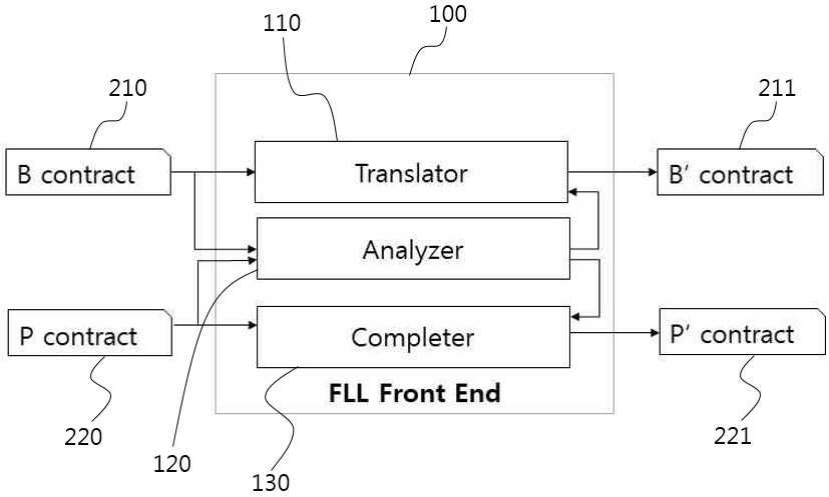
  constructor() {
          hospital = me;
          count = 0;
  }
 function record(address don, bool@me r) {
                  require(hospital == me);
                  risk[don] = reveal(r, don);
                  count = count + (r ? 1 : 0);
          }

          function check(bool@me r) {
                  require(reveal(r == risk[me], all));
          }
}
```

도면2



도면3



도면4

```
function record(address don,
                    bool@me>don r) {
                    require(hospital == me);
                    risk[don] = r;
                    count = count + (r ? 1 : 0);
          }
function check(bool@me>all r) {
                    require(r == risk[me]);
          }
```

도면5

```
contract MedStats {
  final address hospital;
  uint@creator count;
  mapping(address!x:donner => bool@x) risk;

  constructor() {
          hospital = me;
          count = 0;
  }
 function record(address don, bool@me>don r) {
                    require(hospital == me);
                    risk[don] = r;
                    count = count + (r ? 1 : 0);
          }

          function check(bool@me>all r) {
                    require((r == risk[me]);
          }
}
```

도면6

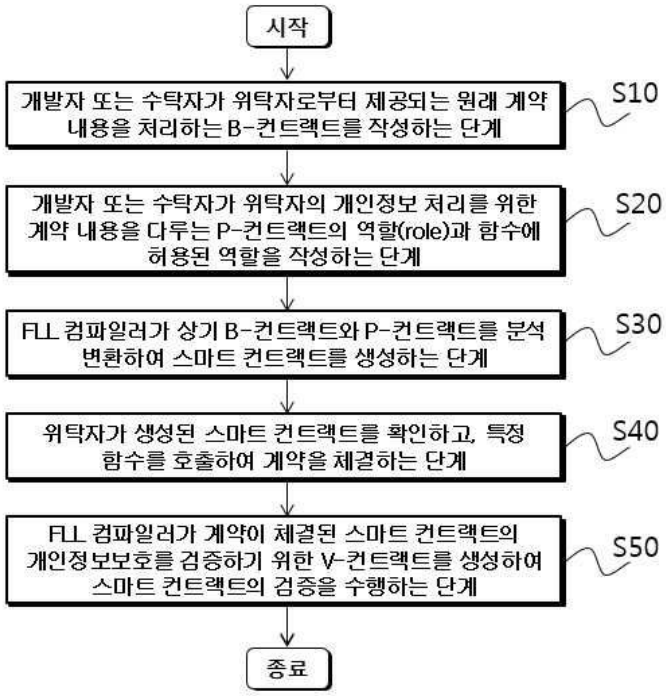
```
// 역할
role: [
@creator, // default (을)
@donner // 개인 (갑)
],

// 함수-역할
function_role: [
[fname: constructor, roles: [@creator]],
[fname: record, roles: [@creator]],
[fname: check, roles: [@donner]] ],

// 아래는 B-컨트랙트로부터 FLL컴파일러에 의해 자동 생성되는
D-집합
// 데이터를 열람하는 위치
[from_role: @creator, [
register : [<register, !x@donner, r, risk[x]>]]

[from_role: @donner, [
check : [<check, all, risk[me]==r>]]]
```

도면7


시작
개발자 또는 수탁자가 위탁자로부터 제공되는 원래 계약 내용을 처리하는 B-컨트랙트를 작성하는 단계
S10
개발자 또는 수탁자가 위탁자의 개인정보 처리를 위한 계약 내용을 다루는 P-컨트랙트의 역할(role)과 함수에 허용된 역할을 작성하는 단계
S20
FLL 컴파일러가 상기 B-컨트랙트와 P-컨트랙트를 분석 변환하여 스마트 컨트랙트를 생성하는 단계
S30
위탁자가 생성된 스마트 컨트랙트를 확인하고, 특정 함수를 호출하여 계약을 체결하는 단계
S40
FLL 컴파일러가 계약이 체결된 스마트 컨트랙트의 개인정보보호를 검증하기 위한 V-컨트랙트를 생성하여 스마트 컨트랙트의 검증을 수행하는 단계
S50
종료