JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 26 No. 4, pp. 75-84, April 2021
https://doi.org/10.9708/jksci.2021.26.04.075

InfoDID: A robust user information management serivce based on Decentralized Identifiers

Min-Ho Kwon*, Myung-Joon Lee**

*Student, Dept. of Electrical/Electronic and Computer Engineering, University of Ulsan, Ulsan, Korea
**Professor, School of IT Convergence, University of Ulsan, Ulsan, Korea

[Abstract]

In this paper, we introduce InfoDID, a robust user information management service based on DID that manages user information reliably. Since blockchain technology provides an environment in which data can be handled transparently on a decentralized basis, various services using blockchain are currently being developed As the importance of user's personal information has recently emerged, the DID technology is receiving attention. The technology allows a user to control his or her information through decentralized identifiers, and various information management services are being tried based on the technology. Using blockchain-based DID technology, InfoDID reliably controls personal information requested frequently, helping users to provide their information to other services more conveniently. In addition, to support service continuity, InfoDID uses BR2K technique that provides robust execution of blockchain application services, so that even partial service failures can be systematically recovered. To facilitate this operation, we present a replication status monitoring tool that can continuously check the replication states of blockchain application services running in association with the BR2K technique such as InfoDID.

▸ **Key words**: Blockchain Service, User information, Decentralized Identifiers, Service Replication, Personal Information Management

[요 약]

본 논문에서는 사용자 정보를 신뢰성 있게 관리하는 DID 기반의 견고한 사용자 정보 관리 서비스인 InfoDID를 소개한다. 블록체인 기술은 탈중앙화 기반으로 데이터를 투명하게 다룰 수 있는 환경을 제공하여 현재 블록체인을 활용한 다양한 서비스들이 개발되고 있다. 최근 사용자의 개인 정보에 대한 중요성이 부각되면서 사용자가 자신의 정보를 통제할 수 있는 분산 식별자를 지원하는 DID 기술이 주목받고 있으며, 이를 기반으로 다양한 정보를 관리하는 서비스들이 시도되고 있다. InfoDID는 블록체인 기반의 DID 기술을 활용하여 빈번히 요청되는 개인 정보를 신뢰성 있게 제어하고, 사용자가 자신의 정보를 보다 편리하게 다른 서비스들에게 제공하도록 지원한다. 또한 서비스의 연속성을 지원하기 위하여, InfoDID는 블록체인 응용서비스의 견고한 실행을 제공하는 BR2K 기법을 채택하여 부분적인 서비스 실패의 경우에도 체계적으로 복구될 수 있다. 이러한 작업을 용이하게 수행하기 위하여, InfoDID와 같이 BR2K 기법을 적용한 블록체인 응용서비스의 복제 상태를 지속적으로 확인할 수 있으며 결함 복구를 지원하는 복제 상태 모니터링 도구가 제시된다.

▸ **주제어**: 블록체인 서비스, 사용자 정보, 분산 식별자, 서비스 복제, 개인정보 관리

• First Author: Min-Ho Kwon, Corresponding Author: Myung-Joon Lee
*Min-Ho Kwon (alsgh458@gmail.com), Dept. of Electrical/Electronic and Computer Engineering, University of Ulsan
**Myung-Joon Lee (mjlee@ulsan.ac.kr), School of IT Convergence, University of Ulsan
• Received: 2021. 02. 18, Revised: 2021. 03. 17, Accepted: 2021. 03. 23.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

일반적으로 사용자가 디지털 기반의 서비스를 이용하기 위하여 각 서비스에 자신의 정보를 제공하고 서비스 계정을 만드는 개별 가입 방식으로 신원을 인증하게 된다. 이와 같은 방식은 사용자가 새로운 서비스를 사용할 때마다 자신의 개인 정보를 서비스에 일일이 제공하여 회원가입을 하는 번거로운 절차를 거치게 된다. 개별 가입을 하지 않을 경우에 사용자는 OpenID, OAuth[1] 등과 같은 기반 기술을 이용해 구글이나 페이스북의 기존 서비스 계정으로 여러 서비스에서 신원을 인증하는 연합 신원 인증을 사용하여야 한다. 이러한 방식은 사용자가 신원 증명을 해주는 서비스에서 다루는 사용자의 정보가 어느 수준까지 제공되는지 확인하기 어렵고 신원 증명 제공자의 서비스가 중지된다면 해당 방식의 신원 인증을 할 수 없는 문제점이 발생한다. 최근에는 무결성, 탈중앙화, 높은 보안성 등의 특징을 가진 블록체인 기술을 이용하여 중앙화된 신원 제공자, 인증 기관 등으로부터 벗어난 검증 가능한 분산 식별자 DID(Decentralized Identifiers)가[2-4,15,16] 차세대 신원 인증 기술로 등장하였고 W3C(World Wide Web Consortium)에서 표준화를 시도하고 있으며[5] 이를 이용한 여러 가지 응용서비스들이 개발되고 있다.[12-14]

본 논문은 사용자의 정보를 신뢰성 있게 관리하기 위한 DID 기반의 견고한 사용자 정보 관리 서비스인 InfoDID를 소개한다. InfoDID는 DID 기반의 블록체인 응용 서비스로 실행되며, 사용자는 해당 서비스를 통해 자신의 정보를 관리하고 DID 인증 기법을 이용하여 다른 서비스에게 손쉽게 전달할 수 있다. 또한 DID 문서를 관리하는 DID 레지스트리를 이더리움 블록체인에서 관리하고 기본 DID 레지스트리를 확장하여, 사용자 정보가 사용자의 승인 없이 임의로 변경되어 사용되지 못하도록 방지한다. 그리고 블록체인 서비스의 견고한 실행을 지원하는 BR2K 기법을[6] 해당 서비스에 적용하여, 서비스 실행에 대한 결함이 발생하여도 사용자에게 지속적인 서비스가 가능하며 어떠한 상황에서도 결함에 대한 복구를 진행할 수 있다. 이를 손쉽게 지원하기 위해서 InfoDID와 같이 BR2K 기법이 적용된 블록체인 응용서비스들의 서비스 상태를 지속적으로 체크하고 간편한 결함 복구 등의 기능을 지원하는 복제 상태 모니터링 도구를 개발한다.

본 논문의 구성은 1장 및 2장에서는 서론과 배경지식이 소개된다. 그리고 3장에서는 BR2K 기법을 적용한 사용자 정보 관리 서비스와 이를 위한 DID 레지스트리를 소개하고 4장에서는 BR2K 기법이 적용된 블록체인 응용서비스의 상태를 모니터링하고 간편한 결함 복구 등을 지원하는 모니터링 도구를 소개한다. 마지막 5장에서는 본 논문의 결론에 대하여 기술한다.

II. Background Knowledge

1. Ethereum and Smart contract

이더리움은 전자 계약인 스마트 컨트랙트 기능을 제공하며 탈중앙 블록체인 애플리케이션을 개발하고 운영할 수 있는 블록체인 플랫폼이다. 이더리움 블록체인에서는 스마트 컨트랙트 기반으로 간단한 거래뿐만 아니라 사용자가 원하는 복잡한 계약도 할 수 있다. 이더리움 블록체인 네트워크에 연결된 모든 노드는 스마트 컨트랙트를 실행시키는 이더리움 가상머신을 통해 모든 트랜잭션을 실행하여 모든 노드는 동일한 계산을 수행하고 같은 상태를 가진다.

스마트 컨트랙트란 기존 대면 방식의 계약에서 벗어나 블록체인에서 특정 조건에 따라 계약을 자동으로 수행하는 스마트 계약 기능이다. 이더리움에서 실행되는 스마트 컨트랙트는 튜링 완전성을 가지고 있어 반복문과 제어문과 같은 코드를 작성하여 복잡한 형태의 계약을 수행할 수 있으며 이더리움 블록체인 기반에서 계약이 실행되기 때문에 데이터 조작 방지 및 무결성 보장이 가능하다. 스마트 컨트랙트는 이더리움 가상 머신에서 작동하는 스마트 계약을 위하여 개발된 정적타입의 프로그래밍 언어인 솔리디티(Solidity)로[17] 작성된다. 해당 프로그래밍 언어는 스마트 컨트랙트에 다양한 비즈니스 로직을 담을 수 있고 Hyperledger와 같은 폐쇠형 블록체인 (Private blockchain) 플랫폼에서도 동작할 수 있다.

2. BR2K scheme

금융, 의료, 공공 인프라 등과 같은 분야에서 블록체인 기반의 응용서비스들이 많이 개발되고 있으며, 이러한 응용서비스들은 지속적으로 서비스를 제공할 수 있는 서비스의 연속성과 각종 재난 및 해킹과 같은 재난적인 상황이 발생하여도 서비스 실패를 극복할 수 있는 체계적인 복구 방법이 필요하다. BR2K는[6] 이러한 문제를 극복하기 위하여, [18]의 연구를 확장하여 블록체인 응용서비스를 복제 실행하고 실행된 서비스들의 상태를 지속적으로 동기화하면서 부분적인 실패에 대한 체계적인 복구를 지원하는 기법이다. 해당 기법은 Raft 컨센서스 프로토콜[8] 기반 스토리지인 ETCD를[9] 이용한 서비스 상태의 복제 실행과 실패에 대한 복구를 체계적으로 지원하는 서비스 레지

스트리로 나뉜다. 서비스 상태의 복제 실행은 분산 환경을 구성하는 각 노드에 블록체인 응용서비스를 실행하고 [6]의 상태 복제 알고리즘에 의하여 서비스 상태를 복제한다. BR2K 기법이 적용된 블록체인 응용서비스는 그림 1과 같이 분산 환경을 구성하는 각 노드에 블록체인 애플리케이션과 ETCD를 실행한다.

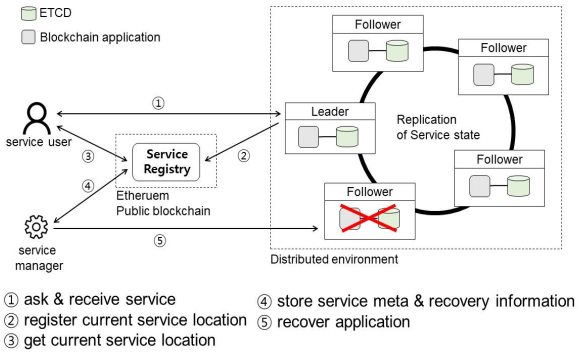

Fig. 1. Service replication with BR2K

BR2K 기법의 상태 복제 알고리즘에 의하여 각 노드의 블록체인 애플리케이션의 서비스 상태는 지속적으로 동기화되고 리더나 팔로워 중 하나의 역할을 가지게 된다. 리더 애플리케이션은 블록체인 응용서비스에서 오직 하나이며 사용자에게 서비스를 제공하는 역할을 한다. BR2K에서 리더 외 모든 애플리케이션은 팔로워 역할이며, 해당 역할은 서비스를 제공하는 리더가 실패하는 경우에 대비하여 지속적으로 서비스 상태에 대한 동기화를 수행한다. 만약 현재 리더 애플리케이션이 임의의 실패에 의해 서비스를 제공하지 못한다면, 리더는 다른 복제 애플리케이션으로 변경된다. 새로운 리더 애플리케이션은 서비스의 메타 정보와 복구 정보를 저장하는 서비스 레지스트리에 서비스 제공 위치를 자신의 위치로 업데이트하여 사용자에게 서비스를 지속적으로 이어나간다.

BR2K 기법이 적용된 블록체인 응용서비스의 사용자는 이더리움 블록체인의 스마트 컨트랙트 기반으로 작성된 서비스 레지스트리에서 현재 서비스 제공 위치를 가져와 서비스를 요청한다. 그림 1의 서비스 관리자는 서비스 결함에 대한 실행 복구를 위하여 서비스 레지스트리에 서비스 메타 정보와 복구 정보를 저장하며, 배포한 서비스에 결함이 생기면 서비스 레지스트리에 복구 정보를 가져와 결함이 있는 복제 애플리케이션을 재배포한다. BR2K 기법이 적용된 블록체인 응용서비스는 분산 환경에서 컨테이너 관리 도구인 쿠버네티스 기반으로 실행되며, 이러한 작업들을 간편하게 하기 위하여 개발된 확장 트러플 프레임워크를 통해 블록체인 응용서비스에 적용한다.

3. DID

DID(Decentralized Identifiers)는 중앙화된 신원 제공자, 레지스트리, 인증 기관 등으로부터 독립되어 검증가능한 분산 디지털 신원 기술이다. DID를 가진 주체는 해당 주체를 설명할 수 있는 DID 문서를 통해 자신의 정보에 대한 제어권을 가지게 된다. DID 문서는 대칭 암호화 키와 같이 문서 주체의 신원을 검증할 수 있는 값과 문서 주체와 상호작용할 수 있는 서비스에 대한 정보가 들어간다. 일반적으로 DID 문서는 분산 원장 기술인 블록체인 기반의 DID 레지스트리에서 관리되기 때문에, DID를 가진 주체는 블록체인 상에서 언제나 식별될 수 있으며 신뢰할 수 있는 신원 인증 기법인 DID auth를 통하여 자신의 신원을 타인에게 검증받을 수 있다. DID 문서가 저장되는 DID 레지스트리는 분산 식별자의 생성, 확인, 업데이트 및 비활성화를 제어하는 기능을 제공한다.

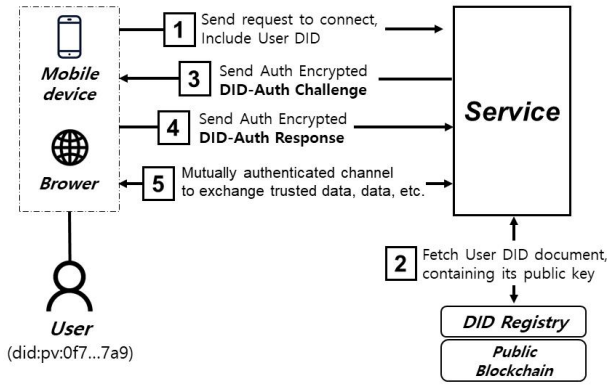

Fig. 2. Typical process of DID auth

DID 사용자는 서비스를 제공받기 위하여, 비대칭 암호화 방식의 비밀 키가(Private key) 저장된 개인 모바일 단말기나 브라우저를 통해 자신의 DID가 담긴 사용자 요청을 서비스에게 보낸다. 서비스는 사용자 DID를 이용하여 DID 레지스트리에 저장된 사용자 DID 문서에서 사용자 공개 키를 가져와 신원 인증에 필요한 값을 암호화하고 사용자에게 보낸다. (fig. 2, DID-Auth Challenge) 암호화된 값을 받은 사용자는 자신의 비밀 키를 통해 해당 값을 복호화하고 서비스에게 다시 보내 신원 인증 요청에 응답한다. (fig. 2, DID-Auth Response) 서비스는 사용자가 보낸 복호화된 인증 값이 유효하다면, 사용자에 대한 신원 인증을 완료하고 서비스를 제공하기 위한 채널을 만들어 사용자에게 서비스를 제공한다.

III. User information Management Service

본 장에서는 신원 인증과 개인 정보 관리에 사용되는 이더리움 블록체인 기반의 DID 레지스트리와 이를 이용한 사용자 정보 관리 서비스인 InfoDID의 구조와 구현 기법에 대해서 설명한다. 또한 BR2K 기법을 이용하여 이를 견고하게 서비스하기 위한 과정 및 실행 환경을 소개한다.

1. DID registry for user information mangement

사용자 정보 관리 서비스인 InfoDID의 모든 서비스는 DID를 기반으로 동작하기 때문에, 이러한 서비스를 지원하기 위한 DID 레지스트리가 필요하다. 이를 위하여 InfoDID는 개인정보의 위변조를 방지하기 위한 정보를 담을 수 있는 W3C의 DID 표준 규격을 확장한 DID 레지스트리를 제공한다. 확장된 DID 레지스트리는 이더리움 블록체인의 스마트 컨트랙트 기반으로 개발되며 사용되는 DID(Decentralized Identifier)의 구조는 RFC3986을[10] 준수하는 URI 체계를 가진다. 그림 3은 사용자 정보 관리를 위하여 확장된 DID 레지스트리를 나타낸다. 확장된 레지스트리에서 DID 문서를 만들 수 있는 주체는 이더리움 계좌를 가지고 있는 개인, 조직, 서비스 등이다. 이더리움 계좌를 가진 사용자는 오직 한번만 레지스트리의 DID 문서 생성 함수인 *create()*를(fig. 4, function of DID registry) 통해 자신의 DID와 DID 문서를 만들 수 있다.

그림 3과 같이 DID 문서에 저장되는 항목은 문서 주체의 분산 식별자인 DID(fig. 3, Id in DID document), 해당 문서의 편집 권한을 가진 주체인 컨트롤러(Controller List) 리스트, 신원 증명에 사용되는 비대칭 암호화의 공개키 리스트(Public Key List), 해당 문서 주체와 상호작용하는 서비스 리스트(Service List), 그리고 관리되는 문서 주체의 개인정보를 위한 항목인 개인정보(Privacy)가 있다.

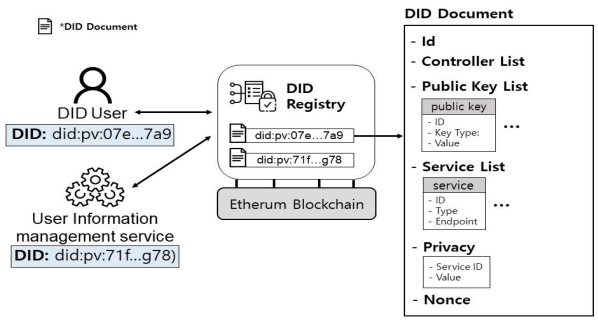

Fig. 3. Extended DID registry

확장된 레지스트리에서 관리되는 모든 DID 문서들은 문서에 대한 DID와 레지스트리의 *readDocument()* 함수를(fig. 4, function of DID registry) 이용해 누구나 읽을 수 있다. DID 문서의 항목 편집은 해당 문서의 주체 또는 컨트롤러 항목에 기록된 주체가 DID 레지스트리의 편집 함수(fig. 4, *add_*()*, *update_*()*, *delete_*()*)를 실행할 때 가능하다. 또는 그림 3과 같이 문서 주체의 이더리움 계정 비밀 키(Private key)로 서명하여 만든 시그니처(Signature)를 통해 DID 문서에 대한 항목 편집이 가능하며 그 과정은 아래와 같다.

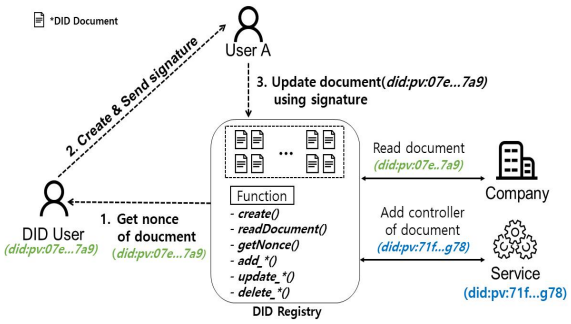

Fig. 4. Using the extended DID registry

(1) DID 문서의 임시값(fig. 3, Nonce in DID document)은 문서 주체가 문서의 편집 작업을 시그니처를 통해 진행할 때 사용되는 값이며, 문서에서 편집 작업이 완료될 때마다 문서의 임시값은 순차적으로 증가한다. DID 사용자는 시그니처를 만들 때 필요한 임시값을 자신의 문서에서 가져온다.

(2) DID 사용자는 문서의 항목 편집에 유효한 시그니처를 만들기 위하여, 이더리움 계정의 비밀 키로 항목 편집 데이터를 서명한다. 항목 편집 데이터는 항목 편집에 사용되는 레지스트리의 함수 이름, 편집할 문서의 DID, 편집할 문서의 현재 임시값 그리고 문서가 존재하는 레지스트리의 주소 값이 포함된다.

(3) DID 사용자가 시그니처를 직접 사용하여 문서에 대한 항목 편집을 하거나 그림 4와 같이 문서 주체가 아닌 타인에게 시그니처를 전달하여 문서에 대한 항목 편집을 진행할 수 있다.

2. store, update process and service scenario

InfoDID는 블록체인 기반의 사용자 정보 저장의 비용 및 용량 한계에 때문에 사용자 정보를 그림 5와 같이 LDAP(Lightweight Directory Access Protocol) 기반의 데이터베이스에 저장하여 관리한다. 또한 해당 서비스는 이더리움 블록체인에 배포된 확장 DID 레지스트리 및

LDAP 기반의 데이터베이스와 상호작용하여 사용자에게 서비스를 제공하는 서버가 있다.

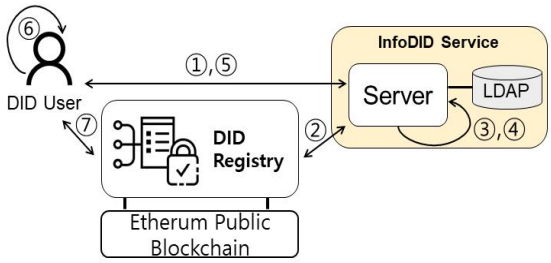

Fig. 5. Managing user information

InfoDID의 LDAP에 저장되는 사용자 정보는 표 1과 같이 사용자의 식별자인 DID, 사용자 정보 관리 시작 시간, 사용자 정보의 마지막 업데이트 시간, 사용자 정보의 유효기간(table. 1, validTime), 사용자의 성과 이름(sn and cn), 프로필 사진, 학력 목록 그리고 경력 등이 저장된다.

Table 1. user information attributes

DID	User decentralized ID for use in this service
createdTime	Time DID user last stored information
updatedTime	Time DID user last updated information
validTime	Time remaining for the service to manage DID user information
sn	Last name of DID user
cn	First name of DID user
gender	Gender of DID user
brith	Brith date of DID user
country	Country of DID user currently living
address	Current address of DID user
job	Current job of DID user
email	Email used by DID users
contact	Phone number or means of contacting DID user
....	
profile picture	Profile picture of DID user
education list	Education list of DID user
hobby	Hobbies of DID user

InfoDID에서 이러한 사용자 정보들은 DID 기반의 사용자 요청을 통해 저장되거나 업데이트 될 수 있다. InfoDID에 사용자의 정보를 저장(업데이트)하는 과정은 다음과 같다.

[step1 in fig. 5] 사용자는 InfoDID의 서버에 저장(업데이트)할 자신의 정보와 자신의 신원을 증명하기 위한 *사용자 인증 정보*를 보낸다. *사용자 인증 정보*는 사용자 DID, 사용자 DID 문서에서 저장된 공개 키의 대칭키인 비밀 키(Private key)로 사용자 정보를 서명하여 만든 시그니처(Signature) 값 그리고 시그니처를 복호화할 수 있는 사용자 DID 문서의 공개 키에 대한 공개 키 아이디(Public Key ID)가 포함된다.

[step2-3 in fig. 5] InfoDID 서버는 받은 사용자 인증 정보의 DID에 해당하는 문서를 DID 레지스트리에서 가져온다. 가져온 문서에서 사용자 인증 정보의 공개 키 아이디를 사용하여 공개 키를 추출한 뒤, 대칭키 암호의 복호화 과정을 거쳐 사용자 인증 정보의 시그니처 값이 유효한지 검증한다. 이 과정을 통해 사용자 정보 관리 서비스는 사용자의 신원 인증과 사용자 정보가 전달되는 과정에서 위조 또는 변조되었는지 확인한다.

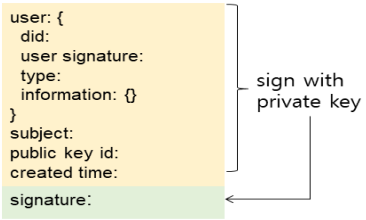

Fig. 6. Structure of management certificate

[step4-5 in fig. 5] 사용자 인증 정보의 시그니처 값이 유효하다면, 해당 서비스의 서버는 받은 사용자 정보를 자신과 연결된 LDAP 데이터베이스에 서비스의 고유 기법에 의하여 암호화하여 저장(업데이트)한다. LDAP에 사용자 정보를 저장(업데이트)했다면, InfoDID의 서버는 사용자에게 사용자 정보를 위변조 없이 저장(업데이트)하였다는 걸 보장하는 *관리증명서*(Management certificate)를 발급해 준다. 그리고 사용자 정보가 다른 서비스에게 전달될 때 LDAP에 저장된 해당 정보들이 비정상적으로 변경되었는지 확인하기 위하여, 해당 서비스의 LDAP에 자신이 발급한 관리증명서를 저장한다.

관리증명서의 구조는 그림 6과 같이 사용자의 DID(fig. 6, DID), 사용자가 보내온 사용자 인증 정보의 시그니처 값(User signature), 사용자 정보의 최근 처리 타입(type, 저장 또는 업데이트), 현재 저장된 모든 사용자 정보(Informations), InfoDID의 DID(Subject), 해당 서비스가 만드는 관리증명서의 시그니처를 복호화할 수 있는 DID 문서의 공개 키 아이디(Public key id), 보증서 생성 시간(Created time) 그리고 앞에 모든 항목을 해당 서비스의 DID 문서에서 유효한 비밀 키로 서명한 시그니처 값으로 구성된다.

[step6-7 in fig. 5] 사용자는 DID 레지스트리에서 InfoDID의 DID 문서를 가져와 관리증명서의 시그니처 값의 유효성을 검증하여, InfoDID가 사용자 정보를 임의로

변경하는 것 없이 저장(업데이트)하였는지 확인한다. 그 후, 사용자는 관리증명서를 자신의 DID 문서에 등록된 첫 번째 공개 키로 암호화하고 암호화된 값을 해시 함수인 Keccak-256을 통해 해시하여 자신의 DID 문서에 있는 개인정보(Privacy) 항목에 저장하여 보관한다. 사용자 DID 문서에 저장된 개인정보 항목의 값은 오직 사용자의 비밀 키에 의해서만 복호화할 수 있다.

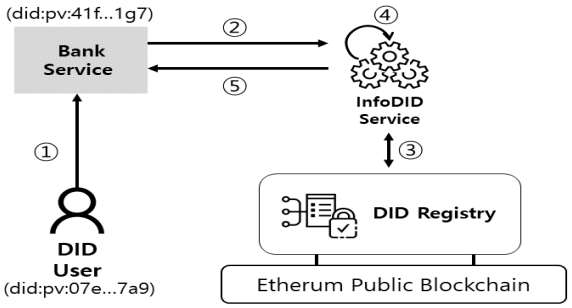

Fig. 7. Example of InfoDID usage

사용자 정보 관리 서비스인 InfoDID가 특정 사용자의 정보를 관리하고 있다면, 해당 사용자는 자신의 정보를 필요로 하는 서비스에 손쉽게 사용자 정보를 전달할 수 있으며 사용자 정보를 사용한 이력을 InfoDID를 통해 관리할 수 있다. 그림 7과 같이 DID 사용자가 사용자 관리 서비스를 통해 은행 서비스에 회원으로 가입하는 과정은 다음과 같다.

[step1-2 in fig. 7] 사용자는 사용자의 DID, 사용자 정보를 받을 서비스의 DID, 회원가입에 필요한 사용자 정보, 사용자 정보를 받을 수 있는 유효시간, 앞서 나온 항목을 사용자의 비밀 키로 서명하여 만든 시그니처 값 그리고 시그니처를 복호화할 수 있는 사용자 DID 문서의 공개 키 아이디가 포함된 *사용자 신원 인증* 정보를 은행 서비스에 전달한다. 은행 서비스는 사용자 신원 인증과 서비스 신원 인증 정보를 InfoDID의 서버에게 보낸다. *서비스 신원 인증* 정보는 은행 서비스의 DID, 은행 서비스의 비밀 키로 사용자로부터 받은 사용자 신원 인증를 서명한 시그니처 값, 시그니처를 복호화하기 위한 은행 서비스 DID 문서의 공개 키 아이디로 구성된다.

Table 2. Pseudocode for [§fig. 7, step 3]

Structure of information received form service	
user {	*service* {
DID	DID
attributes	publicKeyID
validTime	signature }
publicKeyID	
signature }	
1: **IF** *user*.validTime is not timeout*&user*.DID exists in LDAP	
2: **IF** *user & service* document exists in DID registry	
3: Get *user & service* document from DID registry	
4: Extract public keys from *service, user* document as *service*.publicKeyID, *user*.publicKeyID	
5: Verification *service*.signature using public key of *service, user*	
6: Verification *user*.signature using public key of *user, user* (except *user*.signature)	

[step3 in fig. 7] InfoDID 서버는 은행 서비스로 받은 모든 정보를 표 2에 있는 절차대로 수행하여 사용자에 대한 신원 인증, 은행 서비스에 대한 신원 인증 그리고 은행 서비스가 사용자 정보를 읽는 기능에 대한 사용자 승인 여부를 확인한다.

[step4-5 in fig. 7] InfoDID 서버가 은행 서비스로부터 받은 정보에 대한 모든 검증이 완료되면, LDAP에 저장된 사용자 정보가 비정상적으로 변경되었는지 확인한다. 위의 확인 과정은 InfoDID 서버가 해당 사용자에게 마지막으로 발급해 준 관리증명서를 사용자 DID 문서의 첫 번째 공개 키로 암호화하여 Keccak-256으로 해시한 값과 사용자 DID 문서에 저장된 개인정보 항목의 값이 서로 같은지 비교한다. 두 값이 같다면, 은행 서비스에게 사용자에 대한 정보를 발급했다는 기록을 InfoDID의 LDAP에 남기고 필요한 사용자 정보를 은행 서비스에게 전달한다.

3. service replication for robustly executes

사용자 정보 관리 서비스인 InfoDID는 네트워크 장애나 노드의 결함 시에도 지속적으로 사용자 정보를 제공하기 위하여, BR2K 기법 기반으로 상태 복제되어 실행되며 해당 기법을 손쉽게 적용할 수 있는 커맨드 라인 인터페이스 도구인 확장 트러플 프레임워크를(§2.2) 사용한다. 확장 트러플 프레임워크를 이용해 InfoDID의 모든 서비스에 대한 사용자 요청을 BR2K 기법에서 상태 복제를 지원하는 사용자 요청타입인 *Replication* 으로 변경한다.

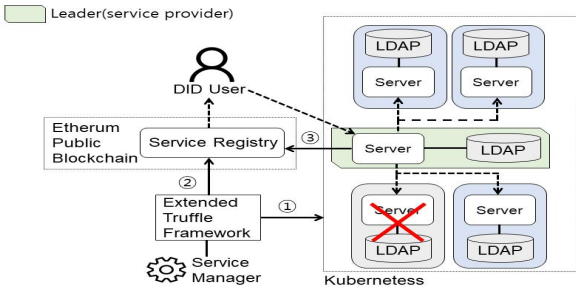

Fig. 8. InfoDID service with BR2K

InfoDID 서비스는 확장 트러플 프레임워크를 이용하여 쿠버네티스 환경에서 복제 실행되며 서비스 이름, 서비스 관리자와 같은 서비스 메타 정보 및 실행에 대한 복구 정보를 서비스 레지스트리에(§2.2) 저장한다. 서비스 상태가 복제된 InfoDID는 그림 8과 같이 쿠버네티스의 각 노드에 서버와 LDAP가 실행되며, 그 중 하나의 서버만이 BR2K 기법에 의해 사용자에게 서비스를 제공하는 리더 역할로 선출된다. 리더 역할을(§2.2) 가진 서버는 서비스 레지스트리에 자신의 위치를 현재 서비스 제공 위치로 업데이트하여 사용자에게 서비스를 제공한다.

사용자는 서비스 레지스트리를 통해서 현재 InfoDID의 서비스를 제공하는 서버의 위치를 가져와, 자신의 정보를 저장(업데이트)하는 요청을 InfoDID의 서버에게 보낸다. 사용자로부터 요청을 받은 리더 서버는 해당 요청을 먼저 실행하여 해당 요청에 대한 오류 여부를 확인하고, 실행한 요청을 각 서버에게 복제하여 리더를 제외한 각 서버가 서로 같은 서비스 상태를 가지도록 한다. 만약 서비스를 제공하는 현재 리더 서버가 어떠한 이유로 중지된다면, BR2K 기법에 의하여 새로운 리더 서버가 선출되어 지속적으로 서비스를 제공할 수 있다. 또한 쿠버네티스에서 실행이 중지된 InfoDID의 서버 및 LDAP는 실행에 대한 복구 정보를 이더리움 블록체인에서 실행되는 서비스 레지스트리에서(§2.2) 가져와 복구 작업을 수행할 수 있다.

IV. BR2K Watch

본 장에서는 InfoDID와 같이 BR2K 기법을 이용하여 상태 복제한 블록체인 응용서비스의 지속적인 상태 확인, 복제 서비스의 편리한 결함 복구 등의 기능을 제공하는 복제 모니터링 도구인 블록체인 웹 응용서비스인 BR2K Watch에 대해서 설명한다.

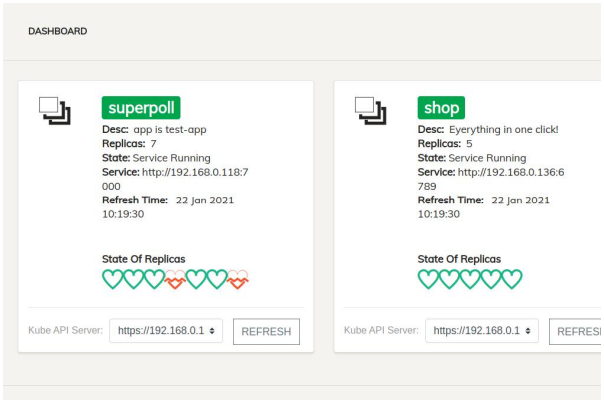

Fig. 9. BR2K Watch view after logging in

BR2K Watch는 웹 서버 프레임 워크인 Vuejs[11] 기반으로 구현되며 BR2K 기법의 서비스 레지스트리 기반으로 동작한다. BR2K Watch의 기능은 그림 9와 같이 서비스 레지스트리에 등록된 서비스 메타 정보(서비스의 이름, 설명, 실행 상태, 복제 수 그리고 서비스 접근 위치)의 출력, 쿠버네티스에서 실행중인 서비스 상태 확인, 각 복제 서비스의 로그 출력, 복제 수 조절, 그리고 간편한 결함 복구가 있다. 서비스 관리자인 BR2K Watch의 사용자는 BR2K Watch를 이용하기 위하여, 서비스 레지스트리에 서비스 정보를 등록할 때 사용한 이더리움 계정으로 로그인해야 한다. BR2K Watch의 로그인 UI 화면에 사용자가 이더리움 계정과 해당 계정의 비밀 키를 입력하여 로그인 버튼을 클릭하면, 표 3과 같은 과정을 거쳐 사용자가 배포한 서비스들의 현재 상태를 BR2K Watch의 대시보드에 출력하고 그 과정은 아래와 같다.

[line 1-2 in table.3] BR2K Watch는 사용자가 로그인할 때 입력한 이더리움 계정 정보를 이용하여 서비스 레지스트리에 등록된 서비스의 정보를 가져온다. 해당 서비스 정보에는 서비스 이름, 서비스를 관리하는 회사, 서비스가 배포되는 쿠버네티스의 접근 위치(table. 3, api-server of service), 쿠버네티스에 접근 권한 값인 액세스 토큰, 배포된 상태, 그리고 서비스의 복제 수 등이 포함된다.

[line 4-5 in table.3] BR2K Watch는 서비스 레지스트리에 기록된 각 서비스의 배포된 상태를 확인한다. 배포된 상태가 실행 상태가 아니라면, BR2K Watch는 쿠버네티스에 현재 실행중인 복제 애플리케이션이 없다고 판단하고 서비스 레지스트리에서 가져온 서비스 정보만을 저장한다.

Table 3. Pseudocode to show service states

Structure of service in service registry

service {
name, company, api-server, access-token,
..., state, replicas, access-location, start-time
}

1: account = getUserAccountInfo()
2: serviceList = readServiceList(account)
3: **for** *service* **in** serviceList
4: **IF** deployed state of *service* is not running
5: store *service*
6: **ELSE**
7: decrypt access-token of *service* using privatekey of account
8: read currrent service status from kubernetes using access-token of *service*
9: check current service status
10: store *service*, current service status
11: print dashboard using stored services information

[line 7-8 in table.3] 서비스의 배포된 상태가 실행 상태라면, 현재 쿠버네티스에서 실행되고 있는 복제 애플리케이션들의 상태를 가져온다. 복제 애플리케이션들의 상태를 가져오기 위하여, 쿠버네티스의 접근 권한 값인 액세스 토큰이 필요하고 사용자 이더리움 계정의 공개 키로 암호화되어 서비스 레지스트리에서 저장된 액세스 토큰을 사용자의 이더리움 계정의 비밀 키로 복호화한다. 그리고 복호화된 액세스 토큰 값과 쿠버네티스의 접근 위치 값을 이용하여, 서비스가 실행되는 쿠버네티스에 접근하고 현재 복제 애플리케이션의 실행 상태 정보를 가져온다.

[line 9-11 in table.3] BR2K Watch는 쿠버네티스에서 가져온 복제 애플리케이션들의 실행 상태 정보를 이용하여 서비스의 현재 복제 상태를 확인한다. 현재 복제 상태의 확인 작업은 해당 애플리케이션이 실행되는 쿠버네티스의 노드 상태, 각 복제 애플리케이션의 실행 상태 그리고 애플리케이션의 상태 복제에 사용되는 ETCD의(§2.2) 상태를 확인한다. 위의 작업이 완료되면, 서비스 레지스트리에서 가져온 서비스 정보와 서비스의 현재 상태를 BR2K Watch에 저장하고 이를 이용하여 BR2K Watch는 그림 9와 같이 서비스의 상태를 대시보드에 출력한다.

로그인 과정 후에 대시보드에 출력된 서비스의 이름을 사용자가 클릭하게 되면, 그림 10과 같이 BR2K Watch의 앱 세팅(App setting) 화면으로 이동할 수 있다. BR2K Watch의 앱 세팅 화면에서는 해당 서비스에 대한 각 복제 애플리케이션의 로그 출력, 서비스의 복제 애플케이션의 개수 조절 및 결함 복구 기능을 사용할 수 있다. 로그 출력은 사용자가 서비스에 현재 실행되고 있는 각 복제 애플리케이션을 선택할 때마다, 쿠버네티스에서 선택된 복제 애플리케이션의 실시간 로그를 출력할 수 있는 로그 스트림(Log stream)을 가져와 대시보드에 로그를 출력하는 기능이다.

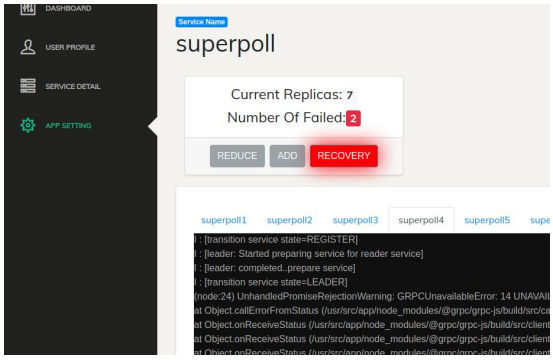

Fig. 10. Setting menu of BR2K Watch

서비스의 복제 애플케이션의 개수 조절 기능은 해당 서비스의 서비스 레지스트리에 등록된 최대 복제 수 범위만큼 복제 수를 조절할 수 있는 기능이다. 해당 기능은 복제 애플리케이션의 개수 조절 버튼을(fig. 10, REDUCE, ADD button) 통해 실행되며, 서비스 레지스트리에서 가져온 서비스 정보를 이용하여 쿠버네티스에 해당 서비스의 애플리케이션의 개수를 늘리거나 줄이고 서비스 레지스트리에 해당 서비스의 복제 애플리케이션 개수를 업데이트하는 과정을 거친다. 만약, 서비스의 현재 복제 상태에 문제가 있다면 복제 애플리케이션의 개수 조절 버튼이 비활성화되어 개수 조절 기능을 사용할 수 없게 되며 결함이 있는 서비스의 복제 애플리케이션에 대한 복구 작업이 필요하다. BR2K Watch의 결함 복구 기능은 서비스의 현재 복제 상태에 문제가 있을 때만 사용 가능하며 복구 버튼(fig. 10, RECOVERY button)을 통해 실행된다. 그림 11은 BR2K Watch에서 결함 복구의 과정을 나타낸다.

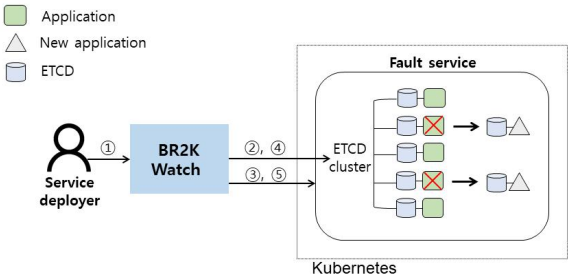

Fig. 11. Recovery process in BR2K Watch

[step 1-2 in fig. 11] 서비스 관리자인 BR2K의 사용자는 BR2K Watch의 대시보드를 통해 서비스 상태 복제에 대한 결함을 발견하고, 앱 세팅 메뉴에서 복구 버튼을 눌

러 결함이 있는 복제 애플리케이션의 복구를 실행한다. BR2K Watch는 가져온 서비스 정보를 이용하여, 쿠버네티스에서 결함이 있는 복제 애플리케이션와 연결된 ETCD를 해당 서비스의 BR2K 상태 복제 작업에 사용되는 ETCD 클러스터에서 제외시킨다.

[step 3-5 in fig. 11] 위와 같은 작업을 수행한 뒤, BR2K Watch는 쿠버네티스에서 해당 서비스의 결함이 있는 애플리케이션을 제거한다. 그리고 BR2K Watch는 서비스 레지스트리에서 가져온 서비스 복구 정보를 이용하여, 해당 서비스의 ETCD 클러스터에 재배포할 애플리케이션과 연결될 ETCD를 새로운 ETCD 멤버로 합류하고 쿠버네티스에 재배포할 복제 애플리케이션과 이와 연결되는 ETCD를 함께 배포하여 서비스의 복제 상태에 대한 결함을 복구시킨다.

V. Conclusions

본 논문에서는 사용자 정보를 관리하기 위한 DID 기반의 견고한 사용자 정보 관리 서비스인 InfoDID를 소개하였다. 본 서비스는 이더리움 블록체인의 스마트 컨트랙트로 개발된 DID 레지스트리를 연계하고 이를 DID 신원 인증 서비스에 활용하여, 사용자의 정보를 신뢰성 있게 관리하고 편리한 형태로 사용자가 자신의 정보를 다른 서비스에게 제공할 수 있도록 지원하였다. 또한 BR2K 기법을 InfoDID 서비스에 적용하여, 부분적인 서비스 실패의 경우에도 체계적으로 복구할 수 있도록 함으로써 사용자 정보를 지속적으로 다른 서비스들에게 제공하도록 하였다. 더불어 InfoDID와 같이 BR2K 기법을 적용한 블록체인 응용서비스의 복제 상태 모니터링 도구인 BR2K Watch를 개발하였다. BR2K Watch는 서비스의 상태를 편리하게 확인하는 기능을 제공하며 서비스 결함 시에 대한 용이한 복구를 지원한다.

ACKNOWLEDGEMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education(No. 2019R1I1A3A01052970)

REFERENCES

[1] Wanpeng Li, Chris J Mitchell, "User Access Privacy in OAuth 2.0 and OpenID Connect", 2020 IEEE EuroS&PW, Sept. 2020. DOI: https://doi.org/10.1109/eurospw51379.2020.00095

[2] Y. Kortesniemi, et al. "Improving the Privacy of IoT with Decentralised Identifiers (DID)", Journal of Computer Networks and Communications, Vol. 2019,pp. 1-10, Mar. 2019. DOI: https://doi.org/10.1155/2019/8706760

[3] C Brunner, et al. "DID and VC:Untangling Decentralized Identifiers and Verifiable Credntials for the Web of Trust", In Proceedings of Preprint.ACM, pp. 6, 2020. DOI: https://doi.org/10.1145/nnnnnnn.nnnnnnn

[4] C Brunner, et al. "SPROOF: A platform for issuing and verifying documents in a public blockchain", Proceedings of the 5th International Conference on Information Systems Security and Privacy, Vol. 1 pp. 15-25, 2019. DOI: https://doi.org/10.5220/0007245600150025

[5] Drummond Reed, et al. "Decentralized Identifiers (DID) v1.0", W3C Working Draft, *https://www.w3.org/TR/did-core*

[6] MH Kwon, MJ Lee. "BR2K: A Replication and Recovery Technique Using Kubernetes for Blockchain Services", Proceedings of the Korean Society of Computer information Conference, Vol. 25, No. 10, pp. 77-86, Oct. 2020. DOI: https://doi.org/10.9708/jksci.2020.25.10.077

[7] M. Wohrer, U. Zdun, "Smart contracts: security patterns in the ethereum ecosystem and solidity", Blockchain Oriented Software Engineering (IWBOSE) 2018 International Workshop on, pp. 2-8, DOI: https://doi.org/10.1109/iwbose.2018.8327565

[8] Donguan Huang, et al. "Performance analysis of the raft consensus algorithm", IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 50, No. 1, pp. 171-181, Jan. 2020. DOI: 10.1109/TSMC.2019.2895471

[9] ETCD, *https://etcd.io/*

[10] RFC3986, https://datatracker.ietf.org/doc/rfc3986/

[11] Vuejs, *https://vuejs.org/*

[12] RAON OmniOne Enterprise, *https://www.raoncorp.com/ko/main*

[13] IconLoop VisitMe, *https://www.iconloop.com/services/*

[14] Coinplug MYKEEPiN, https://coinplug.com/mykeepin#mykeepin_app

[15] MH Rhie, et al. "Vulnerability Analysis of DID Document's Updating Process in the Decentralized Identifier Systems", Information Networking (ICOIN) 2021 International Conference on, pp. 517-520, Jan. 2021. DOI:https://doi.org/10.1109/icoin50884.2021.9334011

[16] ZA Lux, et all, "Distributed-Ledger-based Authentication with Decentralized Identifiers and Verifiable Credentials", 2020 2nd Conference on Blockchain Research & Applications for

Innovative Networks and Services, Sep. 2020. DOI:https://10.1109/brains49436.2020.9223292

[17] Solidity, https://docs.soliditylang.org/en/v0.8.2/

[18] MH Kwon, MJ Lee. "A robust execution scheme for Ethereum blockchain application services", Korean Society of Computer Information, Vol. 25, No. 3, pp. 73-80, March. 2020. DOI: https://doi.org/10.9708/jksci.2020.25.03.073

Authors

Min-Ho Kwon received the B.S/B.A degrees in IT convergence/Economics from University of Ulsan, Korea, in 2020. He is currently an M.S student in Dept. of Electrical/Electronic and Computer Engineering, University of Ulsan. He is interested in blockchain technology, distributed computing, and cloud computing.

Myung-Joon Lee received the B.S. degree in Mathematics from Seoul National University in 1980, and the M.S. and Ph.D. degrees in Computer Science from KAIST in 1982 and 1991, respectively.

Dr. Lee joined the faculty of the Department of Computer Science at University of Ulsan, Ulsan, Korea, in 1982. He is currently a Professor in the School of IT Convergence, University of Ulsan. He is interested in blockchain technology, distributed computing, and mobile/cloud service.