Journal of KIIT. Vol. 20, No. 3, pp. 79-93, Mar. 31, 2022. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2022.20.3.79

블록체인 기반 비상상황 역할 기반 접근제어 시스템

백경동*, 박동규**

Emergency RBAC System based on Blockchain

Kyeong-Dong Baek*, Dong-Gue Park**

이 논문은 순천향대학교 연구비에 의하여 연구하였음.

요 약

역할 기반 접근제어 모델(RBAC, Role-Based Access Control)에서 사용자는 미리 결정된 역할 및 권한에 따라 자원에 대한 접근이 가능하다. 그러나 RBAC은 작업에 대한 권한이 사전에 정의되어 있기 때문에, 비상상황 시에 사용자는 정상적인 상황에서는 인가되지 않은 자원에 대한 접근이 가능하지 못하게 된다. 이를 해결하기 위하여 비상상황에서 시스템을 관리 시에 RBAC의 유연성을 강화하기 위하여 비상 역할기반 접근제어 모델(ERBAC, Emergency RBAC)이 연구되었다. ERBAC에서 사용자는 비상상황을 해결할 수 있는 권한을 부여받게 되고 동시에 비상상황 시의 모든 활동을 기록 검토함으로써 비상상황 시의 작업에 대한 불법에 대하여 책임을 지게 된다. 따라서 ERBAC에서는 비상상황 시의 모든 활동을 기록하고, 불변하도록 유지하는 것은 매우 중요하다. 본 논문에서는 ERBAC 정책을 블록체인 기반으로 구현하여 비상상황 시의 모든 활동을 기록 하고, 블록체인의 특성을 사용하여 활동 기록이 불변하도록 유지함으로써 비상상황 시의 사용자의 접근을 추적할 수 있도록 한다. 마지막으로 본 논문에서 제안된 모델을 이더리움 상에 구현하여 블록체인에서 실행함으로써 제안된 방식의 효율성을 증명한다.

Abstract

The users in Role-Based Access Control(RBAC) can gain access to resources by predetermined roles and permissions. But, in emergency situations users sometimes do not gain access to resources not authorized in normal situations, because the permission of access in RBAC already is defined. To solve this problem, Emergency RBAC(ERBAC) to enhance the flexibility of RBAC for managing the system in emergency situation is studied. The user in ERBAC can be assigned permissions to resources to solve emergency situations and simultaneously is responsible for illegality by auditing the activities performed in emergency situations. Therefore the recording the activities performed in emergency situations and maintaining them invariably in ERBAC is very important. In this paper we audit in blockchain the activities performed in emergency situations by implementing ERBAC policy based on blockchain and can trace the user's accesses in emergency situations by maintaining them invariably according to the characteristics of blockchain. Finally we can validate the effetiveness of the proposed model in this paper by implementing it on Ethereum and executing it in blockchain.

Keywords
ERBAC, emergency, Blockchain, Ethereum

* 순천향대학교 정보통신공학과 석사과정
- ORCID: https://orcid.org/0000-0002-3918-5511
** 순천향대학교 정보통신공학과 교수(교신저자)
- ORCID: http://orcid.org/0000-0002-5864-8825

· Received: Dec. 21, 2021, Revised: Feb. 23, 2022, Accepted: Feb. 26, 2022
· Corresponding Author: Dong-Gue Park
Dept. of Information and Communication Engineering Soonchunhyang Univ.
Tel.: +82-41-530-1347, Email: dgpark@sch.ac.kr.

Ⅰ. 서 론

요즈음 인터넷, 특히 모바일 인터넷이 인간의 생활양식과 사회를 극적으로 변화시킨다는 관점이 널리 받아들여지고 있다. 기술의 발전은 종종 보안 위협을 동반한다. 그래서 데이터 오용, 데이터 기물 파손 및 데이터 유출 등을 방지하기 위한 신뢰할 수 있는 조치가 필요하다. 접근제어 시스템은 컴퓨터 시스템에 저장된 개인정보 및 기업의 기밀문서와 같은 민감한 정보들에 대한 접근을 제어하기 위하여 사용된다.

역할 기반 접근제어(RBAC, Role Based Access Control)는 컴퓨터 시스템 보안에서 권한이 있는 사용자들에게 시스템의 접근을 통제하는 방법으로 사용된다. 이 방법에서 역할은 접근 권한에 매핑되고 사용자는 자원에 접근할 수 있도록 적절한 역할이 할당된다[1]. 그러나 기존의 RBAC은 작업에 대한 권한이 사전에 정의되어 있으므로, 오류 및 예상하지 못한 비상상황이 발생하는 경우 비상상황에서의 접근제어로 적합하지 않게 된다[2]-[6].

비상상황을 해결하기 위하여 사용자는 인가되지 않은 자원 접근에 대한 권한이 필요하게 되며, 비상상황에서 RBAC의 유연성을 높이기 위하여 BTG (Break The Glass)-RBAC이 도입되었다. BTG-RBAC에서는 사용자에게 비상상황을 해결할 수 있는 권한을 부여하는 것과 동시에 비상상황 시의 모든 활동을 기록 검토함으로써 비상상황 시의 작업에 대한 책임을 생성한다. 따라서 BTG-RBAC에서는 비상상황 시의 모든 활동을 기록 하고, 불변하도록 유지하는 것은 매우 중요하다[2]-[6].

불변성, 감사 가능성 및 신뢰성과 같은 블록체인 특성은 블록체인을 비상상황 접근제어 시스템의 도구로 고려하게 만든다. 실제로 [7]에서 설명한 것처럼 블록체인의 분산된 특성은 단일 실패 지점, 변경 불가능한 기록 로그에 대한 액세스 문제를 해결하며, 스마트 계약은 권한 생성을 돕기 위해 모니터링을 수행할 수 있다.

본 논문에서는 권한기반 ERBAC(Emergency RBAC) 정책[6]을 블록체인 기반으로 구현하여 비상상황 시의 모든 활동을 기록 하고, 불변하도록 유지하여 비상상황 시의 모든 활동을 기록함으로써 비상상황 시의 사용자의 작업에 대한 책임을 추적할 수 있도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구, 3장에서는 본 논문에서 제안하는 접근제어 시스템, 4장에서는 시스템 설계를 설명하고 5장에서는 시스템 구현 결과 및 기존 연구와의 비교평가를 기술하며 마지막으로 6장은 결론으로 구성된다.

Ⅱ. 관련 연구

2.1 역할 기반 접근제어

RBAC 모델에는 사용자, 역할 및 권한의 세 가지 기본 요소가 포함되어 있다. RBAC모델은 사용자가 아닌 역할에 권한을 부여한다. 따라서 사용자는 역할 할당을 통해 작업 권한을 얻는다[1]. 시스템은 시스템의 작업에 따라 적절한 역할을 생성한다. 역할이 생성된 후 시스템은 권한을 부여한다. 마지막으로 사용자는 시스템의 보안 요구 사항에 따라 역할에 할당된다. 사용자와 권한은 역할에 따라 연결된다. 따라서 역할을 통해 사용자는 객체에 접근할 수 있다[1].

2.2 비상 역할 기반 접근제어

실제 세계에서 사용자는 때때로 정상적인 상황에서 승인되지 않은 자원에 접근해야 하는 상황이 발생하게 되고, 이 문제를 해결하기 위하여 RBAC에서는 사용자가 제어된 방식으로 접근제어 정책을 재 정의할 수 있도록 하는 BTG 메커니즘이 제안되었다[8]. 따라서 사용자는 일반적으로 정상적으로는 접근할 수 없는 권한에 비상상황에서 접근할 수 있게 된다. 하지만 사용자가 무단으로 다른 역할에 접근을 방지하는 것과 사용자에게 책임을 부여하기 위하여 BTG 상황 시 새로운 권한을 할당받아 일을 해결하는 동안 사용자가 한 행위에 대한 모든 일을 기록하여, 비상상황 시의 작업에 대한 책임을 생성해야 한다[2][4]-[6][9].

다양한 비상상황을 해결하기 위해 사용자의 권한을 기반으로 하는 비상 역할 기반 접근제어 ERBAC

모델이 제안되었다[6].

그림 1은 권한기반 ERBAC 모델을 나타내고 있다[6]. 이 연구에서는 RBAC 정책에 사용자의 권한을 침해하지 않고 최소 권한 원칙을 고수하기 위하여 정상 상황과 비상상황 시에 의무분리와 바인딩 제약조건을 준수하며, 비상상황 시 행하여야 하는 의무 수행 여부에 따라서 시스템의 상태를 제어 모드, 비 제어 모드 두 가지의 경우로 나누어 비 제어 모드로 비상상황이 종료된 후에도 관리자가 직접 감사 기록을 수행하도록 하여 비상상황 처리가 가능하게 하고 있다[6]. 따라서 비상상황 시에 사용자가 새로운 권한을 할당받아 일을 해결하는 동안 사용자가 한 행위에 대한 모든 일을 기록하고, 해당 기록이 훼손되지 않도록 유지하는 것은 사용자가 무단으로 다른 역할에 접근을 방지하기 위하여 매우 중요한 작업이 된다. 불변성, 감사 가능성 및 신뢰성과 같은 특성을 가지고 있는 블록체인은 비상상황 접근제어 시스템의 기반 도구로 매우 적합하게 고려될 수 있다.

2.3 블록체인

블록체인은 저장 및 컴퓨팅을 위한 신뢰할 수 있는 플랫폼을 제공하여 접근제어 메커니즘에 더 많은 보안과 투명성을 제공한다. 이 영역에서 많은 접근 방식이 연구되었으며 블록체인을 사용한 접근제어에 대한 이전 연구 중에 비트코인 블록체인을 사용한 연구가 수행되었다. [10][11]은 해당 연구는 사용자 정의 스크립트를 사용하여 접근제어 정책을 실현하고 스크립트를 트랜잭션에 포함시키는 것이다. 비트코인 프레임워크는 범용 프로그래밍 언어를 지원하지 않기 때문에 맞춤형 스크립트는 유연하고 강력한 로직을 수행할 수 없다는 단점이 있다.

위의 연구와 달리 블록체인에서 실행되는 스마트 계약을 활용하는 연구도 활발히 진행되고 있다[12]-[15]. 논문 [12]는 사물 인터넷의 접근제어를 연구하는 것으로 다양한 접근제어 스마트 계약을 포함한다. [13]는 RABC 메커니즘을 이더리움 플랫폼에서 구현한 것으로 조직 간 RBAC 문제에 대처하기 위하여 연구되었다. [14]는 DApp를 위해 SC-RBAC라는 스마트 계약 기반 RBAC 모델을 제시하였으며, 해당 모델에서도 안전하고 감사 가능하며 투명한 접근제어를 제공하기 위하여 스마트 계약을 사용하고 있다. 그러나 해당 모델들은 동적 특성이 미흡하고 세밀한 권한 레벨에서의 제어기 부족한 단점을 가지고 있다. [15]는 위치 인식 RBAC 모델을 스마트 계약을 사용함으로써 블록체인 상에 구현하여 문맥 인식 및 동적으로 역할을 관리하는 연구를 수행하였다. 그러나 이 연구에서도 사용자 인증 방식이 제외되어 있으며, 비상상황 처리를 고려하지 못한 단점이 존재한다.

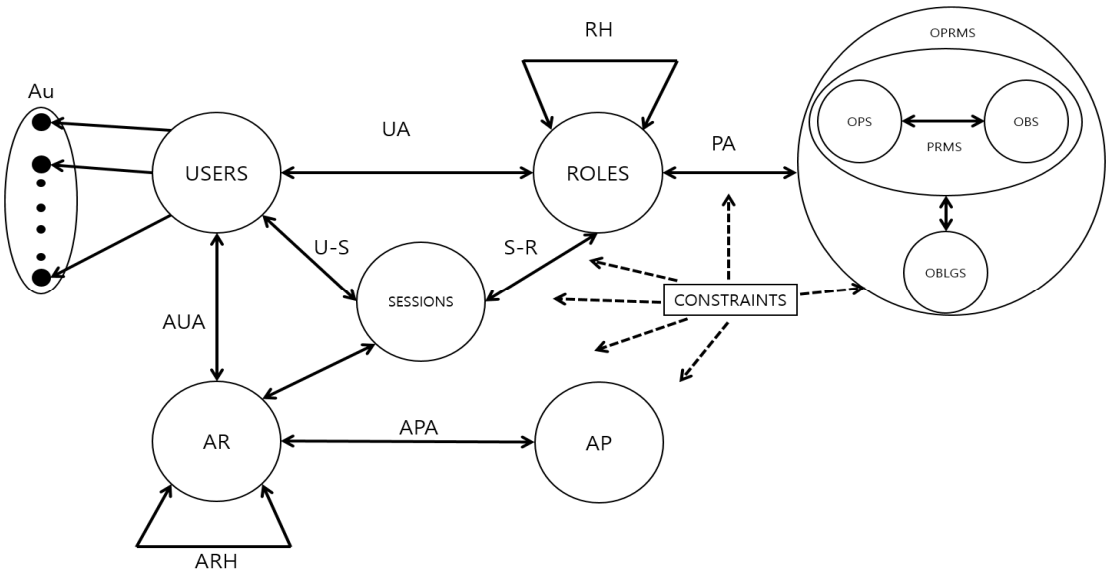

그림 1. 권한 기반 ERBAC 모델
Fig. 1. Permission-based ERBAC model

2.4 신원 기반 서명(IBS, Identity-Based Signature)

본 논문에서는 IBS를 사용하여 엔티티 식별을 제공한다. IBS를 사용하면 엔티티가 이메일 주소, 도메인 이름 또는 물리적 IP 주소와 같은 알려진 ID (ID)에서 공개키를 생성할 수 있다. 개인키 생성기(PKG)라고 하는 신뢰할 수 있는 제 3자가 ID에 해당하는 개인키를 생성한다. PKG는 자신에게 속한 사용자들의 ID에 맞는 개인키를 관리하는 주체이기 때문에, 자신에게 속한 사용자들이 주고받는 모든 데이터를 복호화할 수 있고 서명을 위조 및 변조할 수 있다. 따라서 PKG는 자신이 관리하는 ID를 발급 받은 모든 사용자(사람, 호스트)가 신뢰 할 수 있어야 하며, 어떠한 공격에라도 안전해야 한다.

Kiltz 등[16]은 완전한 IBS 체계를 형성하는 4개의 알고리즘 세트를 정의했다.

설정 : 이 알고리즘은 마스터 키 쌍 (mpk, msk)을 포함하는 전체 IBS 환경을 만들기 위해 PKG(개인키 생성기)에서 한 번 실행된다.

공개 파라미터 : 마스터 공개키 (mpk: master public-key), 마스터 비밀 키 msk (master secret-key)

KeyDer (서명키 생성) : msk 및 엔티티의 ID를 입력으로, 서명키를 생성한다. 이 알고리즘은 ID에 대한 비밀 서명 키 Isk(Id based signature key)를 생성하기 위해 PKG에서 실행된다.

Sign (서명) : 서명키 Isk 및 메시지 M을 입력으로, 이 알고리즘은 M의 서명 σ를 반환한다.

Vf (증명) : mpk, ID, M 및 서명 σ을 입력으로, 이 알고리즘은 서명 σ가 ID 및 M에 대해 유효하면 1을 반환하고 그렇지 않으면 0을 반환한다.

III. 제안된 시스템의 개요

3.1 시스템 모델

본 논문에서 제안한 시스템 모델은 그림 2에서 볼 수 있듯이 주로 시스템 관리자, 관리 시스템(MS) 및 블록체인(BC)의 세 부분으로 구성된다.

시스템 관리자 : 시스템 관리자는 접근제어 정책을 관리하는 관리자로 사용자 역할 및 비상상황 접근 정책을 스마트 콘트랙트를 사용하여 블록체인에 업로드 한다. 또한, 비상상황 시에 수행된 모든 행위들을 블록체인에 업로드 하여 추후의 비상상황 시에 수행된 사용자의 불법행위에 대하여 추적할 수 있도록 한다.

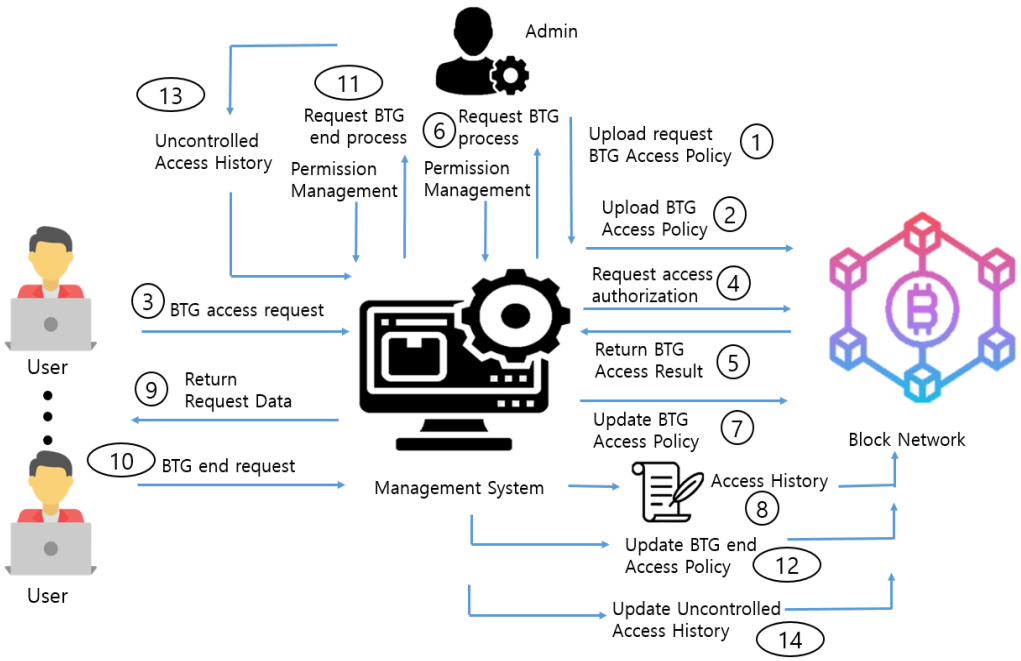

그림 2. 제안된 시스템 모델

Fig. 2. Model of the proposed system

관리 시스템 : 관리 시스템(Management system)은 접근제어를 관리하는 시스템으로 접근 서비스를 제공 할 책임이 있다, 또한 비상상황 시에 수행된 모든 행위들을 블록체인에 업로드 하여 추후의 비상상황 시에 수행된 사용자의 불법행위에 대하여 추적할 수 있도록 한다. 관리 시스템은 블록체인을 유지한다.

블록체인 : 블록체인은 관리 시스템을 감독하는데 사용된다. 블록체인은 사용자 역할, 비상상황 접근제어 정책/요청/접근결정을 포함하는 감사 레코드를 기록한다. 악의적인 공격자를 방지하기 위해 합의 알고리즘을 사용하여 블록체인 원장이 일관 되도록 한다.

제안된 시스템 모델에서 개인 키 생성기(PKG, Private key generator)는 ID가 있는 엔티티 (관리 시스템, 사용자 및 관리자)에 대한 비밀 서명 키 Isk를 생성한다.

제안된 시스템 모델의 처리과정은 다음단계 1~단계 14와 같으며, 표 1은 제안된 시스템에서 사용되는 약어를 설명한다.

표 1. 제안된 시스템에서 사용되는 약어표
Table 1. Symbols of the proposed system

Symbol	Description
H(.)	Collision resistant hash function
Sign(.), Vef(,)	Signature algorithm
$Enc_{key}(.)$, $Dec_{key}(.)$,	Symmetric encryption algorithm
$Enc_{pk}(.)$, $Dec_{sk}(.)$	Asymmetric encryption algorithm
$\{pk_{MS}, sk_{MS}\}$	Management system(MS)'s key pair
$\{pk_a, sk_a\}$	Administrator's key pair
$\{pk_u, sk_u\}$	User's key pair
Isk_a	administrator ID based signature key
Isk_u	User ID based signature key
aID	Administrator ID
uID	User ID
patientID	Patient ID
data	Resource data requested by users

1단계 : 시스템 관리자는 사용자 역할, 역할의 권한, 의무 분리 및 바인딩 등과 같은 BTG 접근제어 정책[6]을 블록체인에 업로드 하기 위하여 관리시스템에 BTG 접근제어 정책 업로드를 요청한다.

2단계 : 관리시스템은 시스템 관리자를 확인하고, 스마트 콘트랙트를 사용하여 블록체인에 BTG 접근제어 정책 업로드를 요청한다.

3단계 : 사용자는 관리시스템에게 비상상황 시의 접근 요청(access_request)을 수행한다. 사용자는의 접근요청시 사용자의 IBS 서명키로 서명하고, 관리 시스템의 공개키로 암호화하여 전송한다.

4단계 : 사용자에게서 접근 요청을 받은 관리 시스템은 자신의 비밀 키(개인키)로 접근 요청을 복원하고, uID로 IBS에 적용하여 접근을 요청한 사용자를 확인한다. 관리 시스템은 사용자를 확인한 이후에, 사용자의 역할을 검증하기 위하여 사용자가 속한 블록체인을 찾아서 스마트 콘트랙트를 사용하여 사용자의 역할을 인증한다.

5단계 : 4단계에서 인증된 사용자의 역할 및 접근제어 요청사항을 가지고 블록체인에 있는 BTG 접근제어 정책을 검토하여 스마트 콘트랙트를 사용하여 사용자 접근제어 요청사항의 인가 및 실패를 결정한다.

6단계 : 5단계에서의 결과가 실패인 경우에 비상상황이기 때문에 관리 시스템은 시스템 관리자에게 BTG 접근제어 정책의 변경을 요청하게 된다. 이 요청에 의하여 시스템 관리자는 사용자가 비상상황을 해결할 수 있는 권한을 결정하여 BTG 접근제어 정책을 수정하여 관리 시스템에게 BTG 접근제어 정책 갱신을 요청한다.

7단계 : 관리시스템은 시스템 관리자를 확인하고, 스마트 콘트랙트를 사용하여 블록체인에 BTG 접근제어 정책의 갱신을 요청한다.

8단계 : 변경된 BTG 접근제어 정책으로 비상상황 시 사용자가 속한 역할이 사용자가 원하는 접근요청을 수행할 수 있는 권한을 가지게 됨으로써, 관리 시스템은 스마트 콘트랙트를 사용하여 사용자의 접근요청에 대하여 블록체인상의 BTG 검사 기록(Audit)에 기록한다.

9단계 : 관리 시스템은 사용자에게 비상상황 시 해당 사용자가 요청한 접근제어 요청의 결과를 반환한다. 관리 시스템은 사용자에게 전달할 데이터를 암호화하는 세션 키를 생성하여 세션 키로 데이터를 암호화한다. 그리고 관리 시스템은 사용자의 공개키로 세션 키를 암호화한 다음, 전달할 데이터의

무결성을 검증하기 위하여 데이터를 해시하고, 해시한 데이터에 관리 시스템의 개인키로 서명을 하여 사용자에게 반환한다.

10단계 : 사용자는 관리시스템으로부터 받은 비상상황 시에 필요한 데이터를 사용하여 비상상황을 해결하고, 비상상황이 종료되면 사용자는 관리시스템에게 비상상황 종료(BTG_end)를 전달한다. 사용자의 비상상황 종료 요청은 사용자의 IBS 서명키로 서명하고, 관리 시스템의 공개키로 암호화하여 전송한다.

11단계 : 사용자에게서 비상상황 종료 요청을 받은 관리 시스템은 자신의 비밀 키로 비상상황 종료 요청을 복원하고, uID로 IBS에 적용하여 비상상황 종료를 요청한 사용자를 확인한다. 관리 시스템은 사용자를 확인한 이후에, 정상 상황으로 복귀하기 위하여 시스템 관리자에게 사용자가 비상상황 시에 받은 권한을 취소하도록 요청한다. 시스템 관리자는 사용자가 비상상황 시에 받은 권한을 취소하도록 BTG 접근제어 정책을 수정하여 관리 시스템에게 수정된 BTG 접근제어 정책 갱신을 요청한다.

12단계 : 관리시스템은 시스템 관리자를 확인하고, 비상상황 시에 받은 권한을 취소하도록 스마트 콘트랙트를 사용하여 블록체인에 BTG 접근제어 정책의 갱신을 요청한다.

13단계 : 시스템 관리자는 관리 시스템으로 부터 받은 비상상황 종료 처리를 수행한 이후에 비상상황 상태가 비 제어 모드(Uncontrolled)인 경우에 수동으로 비상상황 시의 활동들을 감사 기록에 남기기 위하여 비상상황 시의 수행된 모든 관련 활동들의 기록을 관리 시스템에게 요청한다.

14단계 : 관리시스템은 시스템 관리자를 확인하고, 13단계에서 요청받은 기록들을 스마트 콘트랙트를 사용하여 블록체인상의 BTG 검사 기록(Audit)에 기록한다.

3.2 관리자의 비상상황 접근제어 정책 변경

시스템 관리자는 처음에 사용자 역할과 역할의 권한, 의무분리, 바인딩 등 비상상황 접근제어 정책(BTG_policy)을 관리시스템을 통하여 블록체인에 업로드 한다. 시스템 관리자는 자신의 ID와 BTG_policy 그리고 해당 내용을 자신의 ID를 사용한 서명키를 사용하여 서명하고, 식 (1)과 같이 관리시스템 공개키로 암호화하여 관리시스템에 전송한다.

$$Enc_{pkMS}\ (aID \parallel BTG\_policy \parallel Sign_{Iska}(aID \parallel BTG\_policy)) \quad (1)$$

관리시스템은 시스템 관리자로부터 받은 데이터를 식 (2)와 같이 자신의 개인키로 복호화하여 aID ‖ BTG_policy ‖ $Sign_{Iska}$(aID ‖ BTG_policy)를 복호한다.

$$Dec_{skMS}\ (Enc_{pkMS}(\ aID \parallel BTG\_policy \parallel Sign_{Iska}(aID \parallel BTG\_policy))) \quad (2)$$

그리고 관리 시스템은 aID로 IBS에 적용하여 접근을 요청한 시스템 관리자를 확인한 다음, 비상상황 접근제어 정책을 스마트 콘트랙트(uploadBTGpolicy 함수)를 사용하여 블록 네트워크에 기록한다.

사용자의 접근 권한에 대한 변경은 시스템 관리자에 의해서만 진행될 수 있다. 비상상황 접근제어 정책 변경이 요구되는 경우, 시스템 관리자는 관리시스템에 비상상황 접근제어 정책 변경(admin_request)을 요청한다. 시스템 관리자는 자신의 ID와 관리요청(admin_request) 그리고 해당 정보를 자신의 서명키(Isk_a)를 사용하여 서명한 다음, 식 (3)과 같이 관리시스템 공개키로 암호화하여 관리시스템에 전송한다.

$$Enc_{pkMS}\ (aID \parallel admin\_request \parallel Sign_{Iska}(aID \parallel admin\_request)) \quad (3)$$

관리시스템은 시스템 관리자로 부터 받은 식 (3)을 식 (4)와 같이 자신의 개인키로 복호화하여 aID ‖ admin_request ‖ $Sign_{Iska}$(aID ‖ admin_request)를 복호한다. 그리고 관리시스템은 aID로 IBS에 적용하여 접근을 요청한 시스템 관리자가 실제 서명한 시스템 관리자가 맞는지 서명을 확인한다.

$$Dec_{skMS}\ (Enc_{pkMS}(aID \parallel admin\_request \parallel Sign_{Iska}(aID \parallel admin\_request))) \quad (4)$$

관리 시스템은 시스템 관리자를 인증하고, 비상

상황 시 사용자가 원하는 행위를 수행할 수 있도록 변경된 비상상황 접근제어 정책을 uploadBTGpolicy 함수를 사용하여 블록 네트워크에 기록하고 블록 네트워크는 수행 결과를 관리 시스템에 전달한다. 관리 시스템은 결과를 시스템 관리자에게 전달한다.

그림 3은 시스템 관리자에 의해 비상상황 접근제어 정책을 변경하는 과정을 보이는 순서도이다.

3.3 사용자의 비상상황 데이터 접근 요청

사용자는 정상 상황에서는 요청할 수 없는 환자의 데이터에 대하여 비상상황 시에 해당 데이터에 접근을 요청할 수 있다. 예를 들어 비상상황이 발생하여 의사가 자신이 권한 밖의 환자 데이터를 읽으려고 한다면, 의사는 관리시스템에 비상상황으로 데이터 접근 요청을 한다.

사용자의 데이터 접근 요청은 사용자 ID(uID), 환자 ID(patientID), 접근 요청(access_request)과 해당 내용에 대한 서명으로 이루어진다. 사용자는 자신의 Isk_u(ID 기반 서명키)를 사용하여 서명하며, 해당 요청사항을 식 (5)와 같이 관리시스템의 공개키로 암호화 하여 관리시스템으로 보내게 된다.

$$Enc_{pkMS}(uID \parallel patientID \parallel access\_control\_permission \parallel Sign_{Isku}(uID \parallel patientID \parallel access\_control\_permission)) \quad (5)$$

관리시스템은 사용자의 요청사항을 받아서 식 (6)과 같이 자신의 개인키로 복호화하여 uID ‖ patientID ‖ access_request ‖ 서명을 복원하게 된다.

$$Dec_{skMS}(Enc_{pkMS}(uID \parallel patientID \parallel access\_control\_permission \parallel Sign_{Isku}(uID \parallel patientID \parallel access\_control\_permission))) \quad (6)$$

관리시스템은 uID로 IBS에 적용하여 접근을 요청한 사용자가 실제 서명한 사용자가 맞는지 확인하고, 사용자의 요청에 대한 접근 여부를 결정한다. 관리 시스템은 사용자의 역할을 검증하기 위하여 사용자의 블록체인을 찾아서 스마트 콘트랙트(getURole)를 사용하여 사용자의 역할을 인증하고 가져온다. 그리고 관리시스템은 사용자의 인증된 역할 및 요청사항을 가지고, 블록체인에서 자신(관리 시스템)의 공개키를 키로 사용하여 getPolicy를 호출하여 비상상황 접근 정책을 검색한 이후에 비상상황 접근 정책을 가져온다. 블록체인에 있는 BTG 접근제어 정책을 검토하여 스마트 콘트랙트를 사용하여 인가 및 실패를 결정한다.

관리시스템은 비상상황 접근정책 검토 결과가 실패가 나오는 경우에 비상상황 시의 사용을 위하여 시스템 관리자에게 비상상황 접근 권한 갱신을 요청한다. 관리 시스템은 데이터를 암호화하는 세션키(key)를 생성하여 세션 키로 데이터를 암호화하고, 시스템 관리자의 공개키로 세션 키를 암호화한다.

그리고 관리 시스템은 식 (7)과 같이 비상상황 접근 권한 갱신 요청을 해시하고, 해시한 데이터에 관리 시스템의 개인키 sk_{MS}로 서명하여 시스템 관리자에게 요청한다.

$$Enc_{key}(BTG\text{-}update\text{-}request) \parallel Enc_{pka}(key) \parallel Sign_{skMS}(H(BTG\text{-}update\text{-}request)) \quad (7)$$

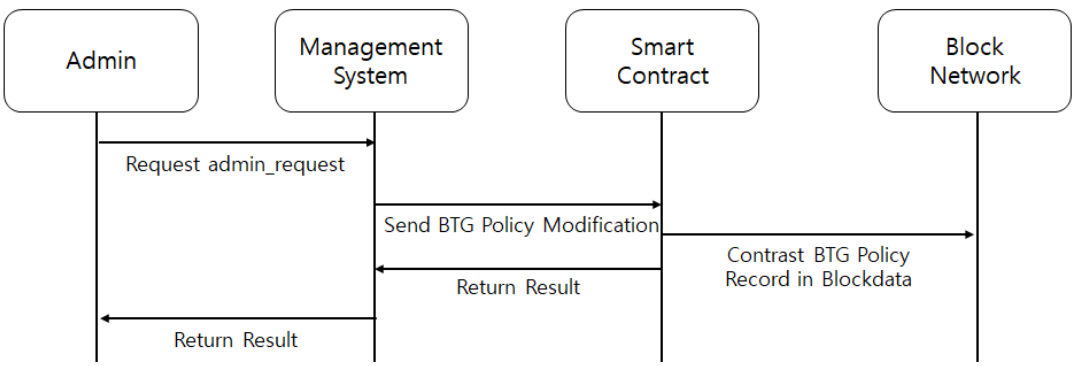

그림 3. 시스템 관리자의 BTG 정책 변경 과정
Fig. 3. Update process of BTG policy of administrator

시스템 관리자는 식 (8)과 같이 자신의 개인키로 세션 키를 복호하여 세션 키를 구한 다음, 해당 세션 키로 비상상황 접근 권한 갱신 요청을 복호한다. 그 이후에 시스템 관리자는 비상상황 접근 권한 갱신 요청을 해싱하고 수신 받은 서명된 해시 값을 관리시스템 공개키로 복호화한 값과 비교함으로써 수신 받은 비상상황 접근 권한 갱신 요청의 무결성을 보장받는다.

$$Dec_{ska}(Enc_{pka}(key)),\ Dec_{key}(Enc_{key}(BTG\text{-}update\text{-}request)),\ Vef_{pkMS}(Sign_{skMS}(H(BTG\text{-}update\text{-}request))) \quad (8)$$

시스템 관리자는 권한 기반 ERBAC [6]에 따라서 비상상황 해결을 위하여 사용자의 역할에 추가할 권한을 결정하고, 식 (3)와 같이 비상상황 정책 수정을 요청한다. 관리 시스템은 시스템 관리자에게서 받은 요청을 식 (4)와 같이 관리 시스템의 개인키를 사용하여 복호한 다음, 시스템 관리자를 인증하고, 스마트 콘트랙트를 사용하여 수정된 비상상황 접근 제어 정책을 블록체인의 해당 블록에서 갱신한다. 그 이후에 해당 사용자는 비상상황 시에 본인이 요청한 권한을 사용할 수 있게 된다. 따라서 관리시스템은 비상상황 시의 사용자 행위에 대한 추후 감사를 수행할 수 있도록, 스마트 콘트랙트(Recordaudit 함수)를 사용하여 사용자의 요청 내용을 자신(관리 시스템)의 블록체인 상의 BTG 검사 기록(Audit) 블록에 기록한다.

그리고 관리 시스템은 사용자에게 해당 사용자가 요청한 데이터를 반환한다. 관리시스템은 데이터를 암호화하는 세션 키를 생성하여 세션 키로 데이터를 암호화하고, 사용자의 공개키로 세션 키를 암호화한다. 그리고 관리시스템은 데이터를 해시하고, 해시한 데이터에 관리 시스템의 개인키 sk_{MS}로 서명하여, 식 (9)와 같이 사용자에게 반환한다.

$$Enc_{key}(data) \parallel Enc_{pku}(key) \parallel Sign_{skMS}(H(data)) \quad (9)$$

사용자는 식 (10)과 같이 자신의 개인키로 복호하여 관리시스템 세션 키를 구하고, 해당 세션 키를 사용하여 암호화한 data를 복호하고, 복호한 data를 해시함수에 적용하여, 수신 받은 서명된 해시 값을 관리시스템 공개키로 복호화한 값과 비교함으로써 수신 받은 데이터의 무결성을 보장받는다.

$$Dec_{sku}(Enc_{pku}(key)),\ Dec_{key}(Enc_{key}(data))\ Vef_{pkMS}(Sign_{skMS}(H(data))) \quad (10)$$

그림 4는 비상상황 시에 사용자의 데이터 요청을 처리하는 과정을 보여준다.

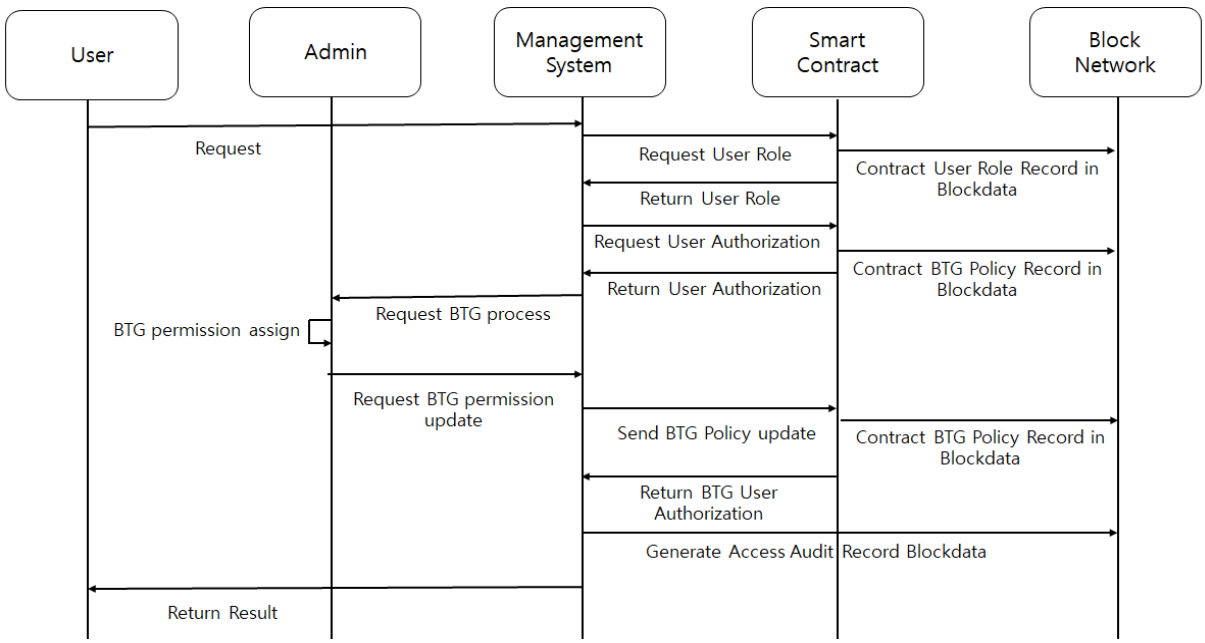

그림 4. 비상상황 시 사용자의 데이터 요청 처리 과정
Fig. 4. Process of user access request in emergency situation

Ⅳ. 시스템 설계

이 절에서는 블록체인 및 비상상황 접근제어를 기반으로 하고 있는 제안된 시스템의 동작을 보여준다. 시스템 설계를 설명하기 전에 먼저 시스템의 트랜잭션 유형과 스마트 계약들을 설명한다.

4.1 트랜잭션

블록체인 시스템에서 트랜잭션은 트랜잭션 헤더와 페이로드의 두 부분으로 구성된다[17]. 시스템에서 요청 트랜잭션은 사용자 ID(uID), 환자 ID (patientID), 접근 요청 및 서명으로 구성된다. 즉, TXrequest = uID ‖ patientID ‖ access_request ‖ 서명이다.

예를 들어 비상상황이 발생하여 의사 Bob이 환자 Alice의 데이터를 읽으려고 한다고 가정하면, Bob은 다음과 같은 요청을 관리시스템에게 해야 한다. Bob은 ID_{Bob} ‖ ID_{alice} ‖ record_read ‖ $Sign_{IskBob}$ (ID_{Bob} ‖ ID_{alice} ‖ record_read)을 관리 시스템에 전송한다. 여기서 ID_{Bob}은 Bob의 ID, ID_{alice}은 Alice의 ID, record_read 는 "레코드 읽기"의 접근 요청 및 Sign은 디지털 서명 체계를 사용하는 서명을 나타낸다.

4.2 스마트 계약

스마트 계약은 일부 실행 가능한 논리를 정의한다. 스마트 계약은 블록체인에 설치되며 트랜잭션에서 호출되는 인터페이스를 제공하는 기능을 정의한다.

제안된 시스템에서 신뢰할 수 있는 접근제어 프로세스를 제공하기 위해 시스템 관리자는 관리 시스템을 통하여 접근제어 스마트 계약을 사용하여 사용자의 역할, 역할의 권한, 의무 분리 및 바인딩 등 비상상황 접근 정책을 블록체인에 업로드 한다. 또한 관리 시스템은 블록체인에 기록된 역할 및 비상상황 접근 정책을 기반으로 접근 결정을 내리기 위해 스마트 계약인 AccessDecision 함수를 사용한다.

신뢰할 수 있는 감사를 제공하기 위해 우리는 사용자의 요청과 accessDecison의 접근 결정을 포함하여 비상상황 시의 모든 활동에 대한 감사 기록을 기록하는 스마트 계약을 설계한다. 스마트 계약은 주로 다음과 같은 다양한 기능 인터페이스를 제공한다.

uploadURole(uID, uRole, sig) : 이 함수는 그림 5와 같이 관리 시스템에 의해 호출되며 사용자의 역할을 블록체인에 업로드 하는데 사용된다. 관리 시스템은 사용자의 ID(uID), 역할 및 관리 시스템의 개인키로 서명한 서명을 전송하여 이 기능을 호출한다. 이 함수는 관리 시스템이 서명한 서명이 해당 관리 시스템이 서명한 것인지를 인증하기 위하여 서명을 확인한다. 그런 다음 블록체인은 사용자의 역할을 검색하기 위하여 uID에 대한 사용자 공개키 (uPK)를 키로 사용하며, 해당 역할을 검색한 이후에 사용자의 역할을 저장하거나 업데이트한다.

```
(1) Input: uID, uRole, sig
(2) output: bool
(3) % invoke by MS to upload the users' role in the DO.
(4) % sig = signature_skMS(uID, uRole)
(5) if verify(pkMS, sig) = true then
(6) {
(7)     if stub.GetState(uID) ! = null then % update the user's role
(8)         stub.PutState(uID,uRole); % store key = uPK, value = rRole in
            blockchain
(9)     return 1;
(10) }
(11) else
(12)    return 0;
```

그림 5 uploadURole 알고리즘
Fig. 5. uploadURole algorithm

uploadBTGPolicy(aID, policy, sig) : 이 함수는 그림 6과 같이 관리 시스템에 의해 호출되며 비상상황 접근 정책을 블록체인에 업로드 하는데 사용된다. 관리 시스템은 비상상황 접근 정책 및 관리 시스템이 서명한 서명을 전송하여 이 기능을 호출한다.

```
(1) Input: aID, BTGpolicy, sig
(2) output: bool
(3) % invoke by MS to upload the admin access policies.
(4) % BTGpolicy = {"role1, op1&op2&op3"}
(5) % sig = signature_skMS(policy)
(6) if verify (pkMS, sig) = true then
(7) {
(8)     if stub.GetState (aPK) ! = null then
(9)         stub.PutState (aPK, policy);
(10)    return 1;
(11) }
(12) else
(13)    return 0;
```

그림 6. uploadBTGPolicy 알고리즘
Fig. 6. uploadBTGPolicy algorithm

이 함수는 관리 시스템이 서명한 서명이 해당 관리 시스템이 서명한 것인지를 인증하기 위하여 서명을 확인한다. 그런 다음 블록체인은 비상상황 접근 정책을 검색하기 위하여 관리자 공개키를 키로 사용하며, 비상상황 접근 정책을 검색한 이후에 관리 시스템 공개키를 이용하여 암호화하여 비상상황 접근 정책을 저장한다.

ChangePermission(aID, Perm_Info, sig): 이 함수는 그림 7과 같이 비상상황 시 시스템 관리자가 동적으로 비상상황 접근 정책을 제어하기 위하여 사용한다. 이 함수는 시스템 관리자의 요청에 의하여 관리 시스템에 의해 호출되며 비상상황 접근 정책 중에 권한 생성을 동적으로 블록체인에 업로드 하는데 사용된다. 관리 시스템은 권한 정보 및 관리 시스템이 서명한 서명을 전송하여 이 기능을 호출한다. 이 함수는 먼저 서명을 한 관리 시스템이 이 함수를 호출한 관리 시스템인지 인증하기 위하여 서명을 확인한다. 그런 다음 블록체인은 비상상황 접근 정책을 검색하기 위하여 시스템 관리자 공개키를 키로 사용하며, 비상상황 접근 정책을 검색한 이후에 관리 시스템 공개키를 이용하여 암호화하여 비상상황 접근 정책을 업데이트한다. 그리고 비상상황 접근 정책을 업데이트한 내용을 블록체인의 감사레코드에 기록한다.

```
(1) Input: aID, Perm_Info, sig
(2) output: bool
(3) % invoke by MS to update the admin access policies.
(4) % Perm_Info = {"permissionId", "permissionName", "service"}
(5) % sig = signature_skMS(policy)
(6) if verify (pk_MS, sig) = true then
(7) {
(8)     if stub.GetState (aPK) ! = null then
            BTGpolicy= stub.GetState (aID);
            update(Perm_Info, BTGpolicy);
(9)     stub.PutState (aPK, BTGpolicy);
            recordAudit (request, 'update_Perm_Info');
(10) return 1;
(11) }
(12) else
(13)    return 0;
```

그림 7. ChangePermission 알고리즘
Fig. 7. ChangePermission algorithm

getURole(uID) : 이 함수는 그림 8과 같이 관리 시스템에 의해 호출되며 사용자의 역할을 블록체인에서 다운로드 하는데 사용된다. 사용자의 ID를 받으면, 이 함수는 블록체인으로 부터 사용자의 역할을 검색하기 위하여 사용자 공개키를 키로 사용하여 해당 역할을 검색한 이후에 사용자의 역할을 반환한다. 사용자의 역할에 대한 검색이 실패하면 role_error를 반환한다.

```
(1) Input: uID
(2) output: string
(3) % obtain the user's role from blockchain.
(4) role = stub.GetState(uID)
(5) if role ! = null then
(6)     return role;
(7) else
(8)     return 'role_error';
```

그림 8. getURole 알고리즘
Fig. 8. getURole algorithm

getBTGPolicy(a_ID) : 이 함수는 그림 9와 같이 관리 시스템에 의해 호출되며 비상상황 접근 정책을 블록체인에서 다운로드 하는데 사용된다. 시스템 관리자의 공개키가 수신되면, 이 함수는 블록체인에서 시스템 관리자의 공개키를 키로 사용하여 비상상황 접근 정책을 검색한 이후에 비상상황 접근 정책을 반환한다. 비상상황 접근 정책에 대한 검색이 실패하면 BTGpolicy_error를 반환한다.

```
(1) Input: aID
(2) output: string
(3) % obtain BTGpolicy.
(4) BTGpolicy = Stub.GetState(a_ID);
(5) if (BTGpolicy ! = null) then
(6)     return BTGpolicy;
(7) else
(8)     return 'BTGpolicy_error';
```

그림 9. getBTGPolicy 알고리즘
Fig. 9. getBTGPolicy algorithm

AccessDecision(request, sig) : 이 함수는 그림 10과 같이 관리 시스템에 의해 호출되며 접근 결정을 내리는 데 사용된다. 접근 요청이 수신되면 접근을 요청한 사용자가 실제 서명한 사용자가 맞는지 사용자 공개키로 서명을 확인한다. 그리고 getURole을 호출하여 사용자의 역할을 가져오고, getBTGPolicy를 호출하여 비상상황 정책을 가져온다. 그런 다음 사용자 역할이 비상상황 접근 정책을 충족하는지 여부를 결정한다. 마지막으로 recordAudit를 호출하여 블록체인에 감사 레코드를 기록하고 허가 여부를 반환한다.

```
(1) Input: request, sig
(2) output: string
(3) % invoke by MS to make an access control.
(4) % sig = signature_sku(request)
(5) % request = uID || patientID || permission,
(6) if verify (pku, sig) = true then
(7) {
(8)     role = getURole (uID)
(9)     if role == 'role_error' then % role is null
(10)   {
(11)     recordAudit (request, 'role_error');
(12)     return 'role_error';
(13)   }
(14)   BTGpolicy = getBTGPolicy (aID)
(15)   if policy == 'BTGpolicy_error' then % role is null
(16)   {
(17)     recordAudit (request, 'BTGpolicy_error')
(18)     return 'BTGpolicy_error';
(19)   }
(20)   if ((role, pstientID, permission) ∈ policy) then
(21)   {
(22)     recordAudit (request, permit);
(23)     return permit;
(24)   }
(25)   else
(26)   {
(27)     recordAudit (request, 'access_error');
(28)     return 'access_error';
(29)   }
(30) }
(31) else
(32)    return 'sign_error';
```

그림 10. AccessDecision 알고리즘
Fig. 10. AccessDecision algorithm

recordAudit(request, decision) : 이 함수는 그림 11과 같이 사용자의 요청, 요청한 접근 시간 및 accessDecison의 접근 결정 즉 role_error, BTGpolicy_error, access_error, sign_error, permit을 블록체인에 기록하는 데 사용된다.

```
(1) Input: request, decision
(2) output: string
(3) % record audit on blockchain
(4) % decision = role_error/BTGpolicy_error/access_error/sign_error/permit
(5) time = time.Now();
(6) stub.PutState (request||time, decision);
```

그림 11. recordAudit 알고리즘
Fig. 11. recordAudit algorithm

4.3 시스템 구축

시스템 설정, 접근제어, 데이터 전송 및 분쟁 처리를 포함하여 시스템의 4가지 주요 단계가 있다.

시스템 설정 : 시스템 관리자는 관리 시스템을 통하여 uploadURole, uploadPolicy를 호출하여 사용자의 역할, 비상상황 접근 정책을 블록체인에 업로드 한다.

접근제어 : 사용자가 접근 요청을 시작하려는 경우 자신의 요구 사항에 따라 요청을 구성한다. 예를 들어, 비상상황이 발생하여 사용자가 환자의 데이터를 읽고 싶어 한다고 하자. 이 경우 사용자가 생성한 요청은 request = (uID, patientID, read, sig)이며 sig는 sig_{Isku}(uID, patientID, read)로써 사용자가 데이터 무결성을 위하여 서명한 것이다. 사용자는 관리 시스템에게 자신의 요청을 전달하고, 관리 시스템은 AccessDecision을 호출하여 접근 결정을 한다. 비상상황이기 때문에 비상상황 접근정책 검토 결과가 실패가 나오는 경우에 관리시스템은 시스템 관리자에게 비상상황 접근 권한 갱신을 요청한다.

시스템 관리자는 허가 기반 ERBAC 정책[6]에 따라서 비상상황 해결을 위하여 사용자의 역할에 추가할 권한을 결정하고, 비상상황 정책 수정을 요청한다. 관리시스템은 시스템 관리자에게서 받은 요청을 스마트 콘트랙트를 사용하여 블록체인의 해당 블록을 갱신한다. 그 이후에 해당 사용자는 비상상황 시에 본인이 요청한 권한을 사용할 수 있게 된다. 그리고 신뢰할 수 있는 추후 감사를 제공하기 위해 관리시스템은 사용자의 요청과 AccessDecison의 접근 결정을 포함하는 비상상황 시의 모든 활동에 대하여 스마트 콘트랙트를 사용하여 자신의 블록체인 상의 BTG 검사 기록 레코드에 기록한다.

데이터 전송 : 블록체인을 통하여 관리 시스템에 접근 결정이 수신되면 관리 시스템은 데이터 리소스에 접근 할 수 있는 사용자에게 데이터를 반환한다. 관리 시스템은 데이터 전송 시 기밀성을 유지하기 위하여 데이터를 암호화하는 세션 키(Key)를 생성하여 세션 키로 데이터를 암호화하고, 세션 키를 보호하기 위하여 사용자의 공개키로 세션 키를 암호화 한다. 그리고 관리 시스템은 데이터의 무결성을 보장하기 위하여 데이터를 해시하고, 해시한 데이터의 신원 확인을 위하여 관리 시스템의 개인키로 서명을 한 다음, 전체 데이터를 사용자에게 반환한다.

분쟁 처리 : 사용자는 관리시스템으로 온 전체 데이터 중에서 자신의 개인키로 세션 키를 복호하고, 복호한 세션 키를 사용하여 접근 요청 결과 데이터를 복호한다. 그 이후에 사용자는 복호한 데이터를 해시하여 관리시스템의 공개키로 해시 값(H(data))을 복호하여 비교함으로써, 전송된 접근 요

청 결과 데이터의 무결성을 확인할 수 있다. 그리고 관리 시스템은 블록체인상의 사용자 감사 레코드 기록을 사용하여 사용자의 접근 기록을 추적할 수 있으므로 비상상황 시의 사용자의 비정상적인 접근 동작을 감지 할 수 있다.

Ⅴ. 시스템 구현 결과 및 기존 방법론과의 비교분석

본 논문에서 제안된 시스템은 이더리움 블록체인상에서 구현하였다. 구현 시 사용한 컴퓨터의 사양은 Windows 10 Pro 64비트(10.0, 빌드 19042)환경의 Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz (12C CPUs), ~ 3.2GHz, 32GB RAM, NVIDIA GeForce GTX 1660 SUPER 이다. 이더리움은 이더리움 가상 머신(EVM, Ethereum Virtual Machine)으로 처리되는 스마트 계약을 기반으로 실행하기 때문에 가상 머신에서는 솔리디티(Solidity)라는 전용 프로그래밍 언어가 사용되고 Remix IDE를 통하여 가상으로 구현 한다. 솔리디티의 각 코드라인은 실행을 위해 일정량의 가스를 사용한다. 가스는 스마트 계약에 필요한 수수료의 일종으로 이더리움에 가치를 두고 있다. 일반적으로 1가스는 0.000000001이더 이다.

구현한 결과는 표 2와 같고 결과를 분석해 보면 처음 스마트 계약을 업로드하는 가스 값이 2758424로 가장 많은 가스 값이 소요되고 추후에 사용자의 역할 인증과 정책에 대한 접근 결정 및 비상상황 대처를 위한 권한 수정은 비교적 적은 가스 값이 소요됨을 알 수 있다.

표 2. 제안된 시스템의 스마트 계약 수행 가스 값 분석
Table 2. Analysis of execution cost of smart contracts in the proposed system

Function	Gas used
Create contract	2758424
uploadURole	57324
uploadBTGPolicy	194822
ChangePermission	137041
getUserinfo	39588
getBTGPolicy	39349
AccessDecision	47912
recordAudit	22476

따라서 본 논문에서 제안된 시스템은 스마트 계약의 업로드를 한번만 수행하기 때문에 본 논문에서 제안하는 방법이 블록체인에서 효율적으로 수행 가능한 것을 알 수 있다.

그림 12는 솔리디티 언어를 사용하여 Remix IDE에 구현한 결과이다. 비상상황 시 기존의 사용자가 상황 해결을 위한 권한을 요청하고, 시스템은 그림 10의 AccessDecision 알고리즘을 적용하여 허가를 결정한다. 허가가 결정되면 시스템은 사용자가 요청한 권한을 부여하여 사용자에게 등록된 권한이 변경됨을 확인할 수 있다.

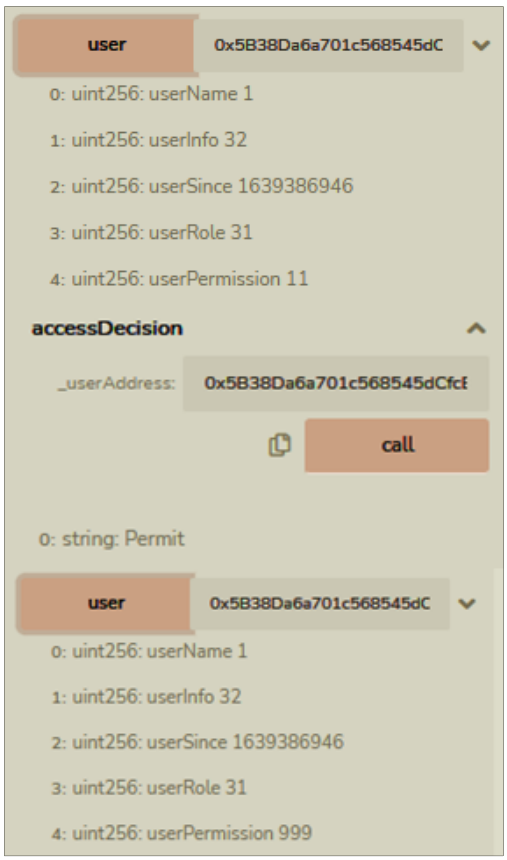

그림 12. 비상상황 권한 부여
Fig. 12. Role assignment in emergency situation

그림 13은 사용자가 등록은 되어있지만 역할과 권한이 아직 할당되지 않은 경우이다. 이 상황에서 사용자는 비상상황 해결을 위해 권한을 요청하게 된다. 시스템은 그림 10의 AccessDecision 알고리즘을 적용하여 사용자가 등록은 되어있지만 역할 구성이 되지 않았음을 확인 후 'Role Error'메시지를 출력하게 된다. 그리고 시스템은 사용자에게 요청한

권한을 부여할 수 없음을 알린다.

그림 14는 권한 기반 ERBAC의 제약조건을 충족하지 못한 상황에서 사용자가 비상상황 해결을 위한 권한을 요청하는 경우이다. 이 경우에 시스템은 'BTG Policy Error'를 출력하여 사용자에게 요청한 권한을 부여할 수 없음을 알린다.

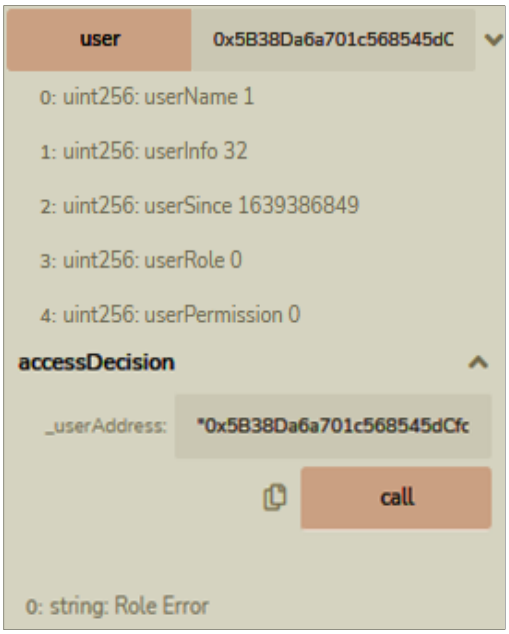

그림 13. 권한 할당 역할 에러
Fig. 13. Role error in authorization

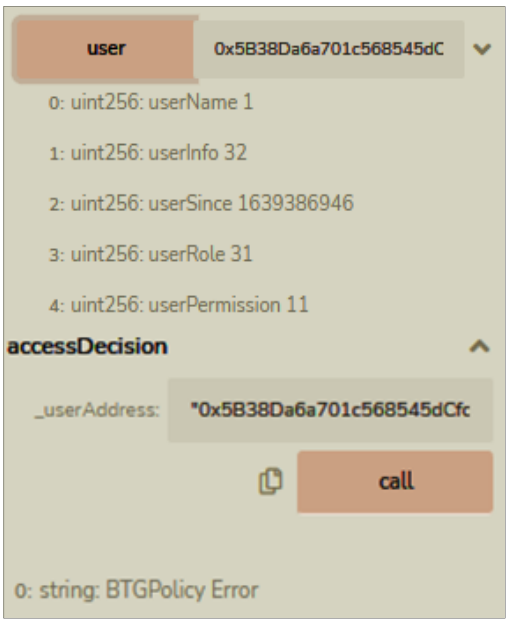

그림 14. 권한 할당 BTG 정책 에러
Fig. 14. BTG policy error in authorization

본 논문에서 제안된 시스템은 사용자의 역할과 비상상황 접근 정책을 기록하는 블록체인을 기반으로 하며, 사용자는 자신의 역할이 비상상황 접근 정책을 충족하는지 확인할 수 있다. 따라서 제안된 시스템은 허용된 사용자 접근을 거부하는 관리 시스템에 대한 공격을 방지 할 수 있다.

또한, 비상상황 시에 수행된 모든 활동에 대한 감사 기록은 블록체인에 저장되어, 수정 공격을 방지하는 해결책이 된다. 그리고 대칭 암호화 알고리즘을 사용하여 사용자가 요청한 데이터를 암호화한 다음, 비대칭 암호화 알고리즘과 요청자의 공개키를 사용하여 세션 키를 암호화하여 반환하기 때문에, 중간자 공격에 대하여 공격을 방지할 수 있다.

본 논문에서 제안된 방법과 기존의 방법들을 비교한 비교 결과표는 표 3과 같다.

표 3. 접근제어 모델 비교
Table 3. Comparison of access control models

	RBAC-SC [13]	SC-RBAC [14]	LRBAC -BC [15]	Our Soluion
Dynamic	X	X	O	O
Context awareness	X	X	O	O
User authentication	O	O	X	O
Fine-grained (permission -level)	X	X	X	O
Emergency situation	X	X	X	O

표 3에서 보이는 바와 같이 본 논문에서 제안한 방식은 권한 기반으로 접근제어를 수행하기 때문에 세밀하게 접근제어 관리가 가능하고, 비상상황 시에 권한을 수정하기 위하여 동적으로 비상상황 접근제어 정책 관리를 수행하며, 중간자 공격에 방지하기 위하여 IBS 기반의 사용자 인증을 수행한다. 또한 권한기반 ERBAC모델[6]은 제약 조건들을 포함하고 있기 때문에 문맥 인식이 가능하며, 비 제어 모드에서도 사용자의 활동에 대한 감사 기록을 수행함으로써, 비상상황 시의 사용자의 활동에 대한 감사 추적이 가능하도록 하였다.

Ⅵ. 결 론

본 논문에서는 권한기반 ERBAC 정책[6]을 블록체인 기반으로 구현하여 비상상황 시의 모든 활동을 기록 하고, 불변하도록 유지함으로써 비상상황 시의 사용자의 작업에 대한 책임을 추적할 수 있도록 한다. 제안된 방법은 불변성, 감사 가능성 및 신뢰성과 같은 블록체인 특성을 기반으로 권한 기반 ERBAC 모델을 구현함으로써 권한 기반의 세밀하고, 동적이며, 비상상황 시에 비 제어 모드 시에도 사용자의 활동을 추적할 수 있도록 한다. 제안된 모델을 이더리움 블록체인 상에 구현하여 스마트 계약을 실행함으로써 제안된 방식이 가능함을 보였다. 향후에는 서로 다른 조직에서 비상상황 시의 유연한 접근제어 방식을 연구할 예정이다.

References

[1] R. S. Sandhu, E. J. Coynek, H. L. Feinsteink, and C. E. Youman, "Role-Based Access Control Models", IEEE Computer, Vol. 29, No. 2, pp. 38-47, Feb. 1996. https://doi.org/10.1109/2.485845.

[2] Ana Ferreira, David Chadwick, Gansen Zao, Pedro Farinha, Ricardo Correia, and Rui Chilro, "How to securely break into RBAC: the BTG-RBAC model", 2009 Annual Computer Security Applications Conference, Honolulu, HI, USA, pp. 23-31, Dec. 2009. https://doi.org/10.1109/ACSAC.2009.12.

[3] Sigrid Schefer-Wenzl and Mark Strembeck, "Generic Support for RBAC Break-Glass Policies in Process-Aware Information Systems", Proceedings of the 28th Annual ACM symposium on Applied Computing, Coimbra Portugal, pp. 1441-1446, Mar. 2013. https://doi.org/10.1145/2480362.2480631.

[4] Fatemeh Nazerian, Homayun Motameni, and Hossein Nematzadeh, "Emergency role-based access control (E-RBAC) and analysis of model specifications with alloy", Journal of Information Security and Applications, Vol. 45, pp. 131-142, Apr. 2019. https://doi.org/10.1016/j.jisa.2019.01.008.

[5] Abdelkrim BOUADJEM, Mustapha Kamel ABDI, "Towards an Extension of RBAC Model", International Journal of Computing and Digital Systems, Vol. 10, No. 1, pp. 1145-1155, Nov. 2021. http://dx.doi.org/10.12785/ijcds/1001103.

[6] Kyeong-Dong Baek and Dong-Gue Park, "Permission Based ERBAC", Journal of Korean Institute of Information Technology, Vol. 19, No. 6, pp. 109-124. Jun 2021. http://dx.doi.org/10.14801/jkiit.2021.19.6.109.

[7] S. Rouhani and R. Deters, "Blockchain based access control systems: state of the art and challenges", IEEE/WIC/ACM International Conference on Web Intelligence, Thessaloniki, Greece, pp. 423-428. Oct. 2019.

[8] S. Schefer-Wenzl, H. Lovasz-Bukvova, and M. Strembeck "A review of delegation and break-glass models for flexible access control management", International conference on business information systems, Larnaca, Cyprus, pp. 93-104, May 2014. https://doi.org/10.1007/978-3-319-11460-6_9.

[9] Qais Tasali, Christine Sublett, and Eugene Y. Vasserman, "Controlled BTG: Toward Flexible Emergency Override in Interoperable Medical Systems", Security and Safety, Vol. 6, No. 22, Jul. 2018. http://dx.doi.org/10.4108/eai.13-7-2018.163213.

[10] A. Ouaddah, A. Abou Elkalam, and O. A. Ait, "FairAccess: a new blockchain based access control framework for the internet of things". Secur. Commun. Netw. Vol. 9, No. 18, pp. 5943-5964, 2016. http://dx.doi.org/10.1002/sec.1748.

[11] A. Outchakoucht, E. S. S. Hamza, and J. P. Leroy, "Dynamic access control policy based on blockchain and machine learning for the internet of things", Int. J. Adv. Comput. Sci. Appl., Vol. 8, No. 7, pp. 417-424, Jul. 2017. http://dx.doi.org/10.14569/IJACSA.2017.080757.

[12] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart contract-based access control for the

internet of things", IEEE Internet Things J., pp. 1-11, Jun. 2018. http://dx.doi.org/10.1109/JIOT.2018.2847705.

[13] J. P. Cruz, Y. Kaji, and N. Yanai, "RBAC-SC: role-based access control using smart contract", IEEE Access, Vol. 6, pp. 12240-12251, Mar. 2018. http://dx.doi.org/10.1109/ACCESS.2018.2812844.

[14] D. Yi, J. Jun, Z. Jinglun, W. Zhongyi, and Kai Hu, "SC-RBAC: A Smart Contract based RBAC Model for DApps", Human Centered Computing, Springer, pp. 75-85, Dec. 2019. https://doi.org/10.1007/978-3-030-37429-7_8.

[15] Mohsin Ur Rahman, Barbara Guidi, Fabrizio Baiardi, and Laura Ricci, "Context-Aware and Dynamic Role-Based Access Control Using Blockchain", International Conference on Advanced Information Networking and Applications, AINA Caserta, Italy, pp. 1449-1460, Mar. 2020. https://doi.org/10.1007/978-3-030-44041-1_122.

[16] E. Kiltz and G. Neven, "Identity-based signatures", IdentityтBased Cryptography, Vol. 2, pp. 31-44, 2009.

[17] L. Zhou, L. Wang, Y. Sun, and P. Lv, "BeeKeeper: a blocktchain-based IoT system with secure storage and homomortphic computation", IEEE Access, Vol. 6, pp. 43472-43488, Jun. 2018. https://doi.org/10.1109/ACCESS.2018.2847632.

저자소개

백 경 동 (Kyeong-Dong Baek)

2013년 3월 ~ 2020년 2월 : 순천향대학교 정보통신공학과(학부)
2020년 3월 ~ 현재 : 순천향대학교 정보통신공학과(석사과정)
관심분야 : 제어 시스템, 모바일 애플리케이션, 시스템 보안

박 동 규 (Dong-Gue Park)

1992년 2월 : 한양대학교 전자공학과(공학박사)
1992년 3월 ~ 현재 : 순천향대학교 정보통신공학과 교수
관심분야 : 제어시스템 보안, 네트워크보안, 시스템 보안, 모바일 보안