Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.31, NO.6, Dec. 2021 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2021.31.6.1157

블록체인과 분산 스토리지를 활용한 프록시 재암호화 기반의 사용자 중심 재해 복구 시스템*

박 준 후,[1†] 김 근 영,[2] 김 준 석,[2] 류 재 철[3‡]
[1]아이오트러스트 (연구원), [2,3]충남대학교 (대학원생, 교수)

User-Centric Disaster Recovery System Based on Proxy Re-Encryption Using Blockchain and Distributed Storage*

Junhoo Park,[1†] Geunyoung Kim,[2] Junseok Kim,[2] Jaecheol Ryou[3‡]
[1]IoTrust (Researcher), [2,3]Chungnam National University (Graduate student, Professor)

요 약

재해 복구는 자연 재해와 같은 비상 상황에 대비하여 서비스의 연속성을 보장하고, 리소스와 재정의 손실을 최소화하기 위한 정책 및 절차를 의미한다. 특히, 클라우드 서비스 제공자에 의한 재해 복구 방법은 관리의 유연성과 고가용성, 비용효율성과 같은 장점을 지닌다. 하지만, 이러한 방법은 서비스 사업자에 대한 의존성을 가지며, 개인의 데이터에 대해 사용자가 관여할 수 없는 구조적 한계를 가진다. 본 논문에서는 블록체인과 분산 스토리지를 활용하여 사용자의 데이터를 백업함으로써 서비스 제공자에 대한 의존성을 제거하고, 데이터의 기밀성을 위해 프록시 재암호화를 이용한 프로토콜을 제시한다. 제안된 방법은 이더리움과 IPFS 환경에서 구현되었으며, 백업 및 복구 운영에 필요한 성능과 비용을 제시한다.

ABSTRACT

The disaster recovery refers to policies and procedures to ensure continuity of services and minimize loss of resources and finances in case of emergency situations such as natural disasters. In particular, the disaster recovery method by the cloud service provider has advantages such as management flexibility, high availability, and cost effectiveness. However, this method has a dependency on a service provider and has a structural limitation in which a user cannot be involved in personal data. In this paper, we propose a protocol using proxy re-encryption for data confidentiality by removing dependency on service providers by backing up user data using blockchain and distributed storage. The proposed method is implemented in Ethereum and IPFS environments, and presents the performance and cost required for backup and recovery operations.

Keywords: Blockchain, Disaster Recovery, Smart Contract, Proxy Re-Encryption, IPFS

I. 서 론

IT 시스템과 인프라가 발전하면서 많은 서비스들에서 사용자에게 수준 높은 서비스를 제공하기 위하여 실시간성과 높은 가용성의 보장 요구가 증가하고 있다. 특히, 금융 시스템을 비롯한 중요 데이터를 처리하는 시스템에서는 백업 및 복구 체계를 갖추는 것이 필수적이다. 서비스를 제공하는 기업은 사용자

Received(08. 19. 2021), Modified(10. 07. 2021), Accepted(10. 30. 2021)
* 이 연구는 충남대학교 학술연구비에 의해 지원되었음.

† 주저자, jh.park@iotrust.kr
‡ 교신저자, jcryou@cnu.ac.kr(Corresponding author)

에게 신뢰성을 보장하고, 발생할 수 있는 비상 사태에 대비하는 계획을 수립하는 것이 중요하다[1].

재해 복구(Disaster Recovery)는 IT 서비스가 재해 유형으로부터 신속하게 복구하기 위한 준비 및 계획에 관한 절차로, 기후와 같은 자연 재난과 고의적 및 의도된 중단, 정전과 같은 서비스 손실, 시스템 장애 및 해킹에 이르기까지 다양한 환경에서 서비스의 연속성을 보장하는 것을 목표로 한다. 서비스 제공자 입장에서 재해 복구 계획은 시스템으로 인한 손실 및 복구의 최소화를 통해 비즈니스 운영 중단으로 인한 재정적 손실 및 평판 손상을 최소화하기 위한 절차이다. 특히, 데이터 복구 계획은 가능한 한 빠르게 정상 작동으로 복원하는데 필요한 모든 솔루션과 단계를 제공하여 연속성을 유지하는 것에 목적이 있다. 장애로 인한 운영의 연속성을 보장하기 위한 내결함성과는 달리, 재해 복구는 비즈니스 서비스의 심각한 손상 및 장기적인 중단에 초점을 둔다.

재해 복구 시스템은 대규모의 기업에서 자체적으로 운영하는 형태뿐만 아니라, CSP(Cloud Service Provider)를 통해 제공하는 형태로 발전하였다[2-7]. 클라우드 컴퓨팅은 전 세계적으로 분산된 리소스를 공유하며 실제로 이용한 리소스만큼의 비용을 지출하도록 구성된다. 이로 인해, 재해 복구에 있어 클라우드 컴퓨팅의 적용은 유연성과 비용 효율성, 안전성 및 확장성과 같은 장점을 가진다.

CSP에 의해 제공되는 재해 복구 서비스의 형태는 Fig. 1과 같다. 사용자가 클라이언트 시스템을

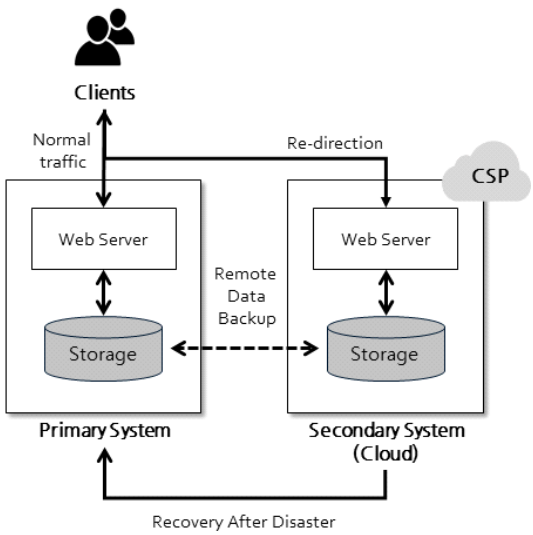

Fig. 1. Cloud based Disaster Recovery System.

통해 서비스를 제공받는 주 시스템(Primary System)에서 필요한 서버와 스토리지 자원 등을 관리하면서, 주기적으로 클라우드 환경에 백업을 시도한다. 재해로 인한 장애가 발생할 경우, 클라우드 환경으로 리디렉션을 시도하거나 클라우드 환경의 데이터를 다시 주 시스템으로 복원하여 사용자에게 계속해서 서비스를 지원할 수 있도록 구성되어있다.

하지만, 클라우드 컴퓨팅에 기반한 재해 복구 시스템은 구조적으로 다음과 같은 한계점을 가진다. 첫 번째는, CSP에 대한 의존성 문제다. CSP가 제공하는 재해 복구는 서비스 기업이 시스템과 데이터의 복제를 저장한 CSP에 대해서 신뢰 여부와 지속성을 염두에 두어야만 한다. 두 번째는 데이터 관리 측면에서 개인 정보와 같은 데이터에 대해 소유주인 사용자의 개인 정보 보호와 기밀성 문제에 관여할 수 없다는 한계가 있다[8].

재해 복구 환경에서 플랫폼에 대한 의존성을 없애고, 사용자 중심의 데이터 관리를 위해 블록체인을 활용할 수 있다. 또한, 블록체인의 적용으로 탈중앙화된 인프라를 통해 CSP의 의존성을 제거하고, 사용자가 데이터를 직접 제어 및 관리할 수 있도록 스마트 컨트랙트를 활용할 수 있다. 스마트 컨트랙트는 사용자가 데이터의 소유주인지와 데이터에 접근 가능한 엔티티를 식별하는 접근 제어의 역할을 하도록 구성될 수 있다.

추가적으로, 데이터를 저장하기 위해서 분산 스토리지를 활용한다. 분산 스토리지는 P2P 환경에서 참여자들의 저장 공간을 활용하여 데이터를 공유하는 시스템으로, 블록체인에 데이터를 직접 저장하지 않음으로써 저장 비용을 낮추고 확장성을 높일 수 있다. 제안하는 방법에서는 재해 복구 과정에서 필요한 사용자의 데이터를 분산 스토리지에 저장하여 복구에 활용한다.

개인의 데이터를 분산 저장하기 위해 필요한 기밀성에 대한 요구사항은 프록시 재암호화(Proxy Re-Encryption)를 활용한다. 프록시 재암호화는 양자 간의 데이터 교환에서 기밀성을 제공하는 암호체계로, 비밀 키를 공유하지 않고 암호화된 데이터를 교환 가능하도록 한다. 데이터 공유에 필요한 프록시는 탈중앙화 환경에서 평판 시스템에 기반한 모델을 통해 구현하여 다수로 운영함으로써, 단일 지점 장애를 방지한다.

본 논문의 기여는 다음과 같다. 첫 번째는 기존에 제시되지 않았던 클라우드 의존성을 제거한 블록체인

기반의 재해 복구 시스템 아키텍처를 제시한다. 두 번째는 제안하는 방법을 구현하고, 재해 복구 환경의 운영을 위해 필요한 분석 및 평가를 제시한다.

본 논문의 구성은 다음과 같다. 2절에서는 재해 복구 및 제안하는 환경의 배경 지식에 대해서 설명한다. 3절에서는 제안하는 방법에 대한 구성요소와 재해 복구 시스템에 필요한 백업 및 복구 절차를, 4절에서는 제안된 시스템의 구현에 대해 설명한다. 5절에서는 구현된 결과로부터 평가를 진행하고 6절에서 관련 연구와의 차이점을 비교 후, 7절에서 결론을 맺는다.

II. 배경 지식

2.1 재해 복구 시스템

재해 복구 시스템은 재난 발생 전 서비스를 유지하기 위한 주 시스템(Primary 혹은 Main System)과 재난 상황에 대비하기 위한 서브 시스템(Secondary 혹은 Serve)으로 구성된다. 각 시스템은 서비스를 위한 환경에 해당하는 서버와 데이터를 저장하는 스토리지로 구분할 수 있다. 주 시스템과 서브 시스템은 물리적으로 분리되어있는 것이 바람직하며, 클라우드 기반의 재해 복구의 경우, 하드웨어와 소프트웨어를 분리하여 서버에 해당하는 OS나 소프트웨어 도구 등을 가상화 환경으로 구성하여 안전하게 마이그레이션할 수 있다.

특히 클라우드 컴퓨팅 기술이 발전함에 따라, 재해 복구 환경에서 클라우드 시스템을 활용하는 방법이 제안되었다. 클라우드 시스템을 활용함으로써 가질 수 있는 장점이 비클라우드 기반 접근 방식에 비해 비용효율적인 전략이기 때문이다. 직접 재해 복구 환경을 구축하는 것에 비해, 클라우드 방식은 가상화를 통해 예비 시스템의 하드웨어 사양과 무관한 독립성을 가지며, 데이터나 운영체제, 소프트웨어 도구등을 쉽게 마이그레이션 할 수 있는 장점을 가진다. 또한, 클라우드에서 사용되는 연산 비용과 데이터의 보관 및 관리에 드는 비용의 유연성도 장점이다.

이러한 장점에 기반하여 [2]와 [3]에서는 클라우드 서비스에 기반한 재해 복구아키텍처를 제안하였다. [2]은 클라우드 환경의 높은 가용성과 낮은 비용에 목적을 두고 아키텍처가 설계되었으며, [3]데이터 백업 과정에서 압축과 암호화를 수행하는 것을 포함한 트래픽 전송 비용의 감소 방법을 제안하였다.

하지만, 다른 시스템과 마찬가지로 클라우드 컴퓨팅 시스템도 종속성, 장애 감지, 보안, 인명 피해, 자연 재해와 같은 위험에 노출된다. 이러한 모든 위험은 클라우드 서비스 중단으로 인한 데이터 손실로 이어질 수 있기 때문에 분산 관리 메커니즘이 필요하다. 예를 들어, Amazon S3, Google GFS, Apache HDFS와 같은 클라우드 플랫폼에서 사용되는 분산 스토리지 시스템은 3-replicas data redundant mechanism 을 적용하였다. 그러나 데이터 센터에 장애가 발생하거나, 센터 간에 공유되는 공통 원인으로 인해 복수의 데이터 센터 오류가 발생할 위험이 여전히 존재하며, 정전 및 공공 서비스 중단 사례가 발생할 수 있다[9].

[4-7]은 이러한 문제를 해결하기 위하여 복수의 클라우드 서비스를 이용한 재해 복구 아키텍처를 제안한다. [4]에서는 데이터 백업을 수행할 때 클라우드 제공 업체에 대한 의존성을 제거하는 것을 주요 목표로 하여, 하나의 CSP에서 다른 CSP로 마이그레이션하는 절차와 비용을 제시하였고, [5]는 복수의 클라우드 서비스 공급자가 존재하는 환경에서 서로 연동하여 백업 및 복구할 수 있는 프로세스를 제공한다. [6]은 데이터 백업이 분산된 위치에 있는 다중 사이트 재해 복구를 위한 배포 계획과 보호 수준 및 제약을 기반으로 시나리오를 제시하였다. [7]에서는 주 시스템과 예비 시스템 모두를 클라우드 환경에서 제공하고 업데이트하는 웜 대기 접근 방식을 제시하였다.

이와 같은 복수의 클라우드 환경에 기반한 재해 복구 시스템은 단일 CSP에 대한 의존성 문제를 해결함으로써, 서비스 기업에게 신뢰성을 제공할 수 있는 방법이지만, 데이터에 대한 개인 정보 보호 및 기밀성을 고려하지 않는다. CSP에 의한 개인정보 유출 사례는 지속적으로 발생해 왔으며, 클라우드 기반에서 사용자의 데이터가 실제 위탁한 서비스 기업의 서버가 아닌 제 3자 기업에 존재하는 환경에서 공격자의 타깃이 되고 있으며, 고객은 이에 대한 데이터 암호화 및 접근 제어에 관여할 수 없는 구조적 한계를 가지고 있다. 특히, 개인의 데이터에 대한 법과 관리에 대한 문제가 대두되고 있는 가운데, 규정 준수와 데이터의 정보 보증 여부는 오로지 신뢰할 수 있는 CSP의 선택에만 의존해야 하지만, 가장 널리 알려진 클라우드 서비스에서도 사건 및 사고가 발생하고 있다[9].

블록체인은 다수의 참여자가 노드를 형성하여 합

의를 통해 운영되기 때문에 단일 지점 장애가 없으며, 사용자 스스로 자산 및 데이터에 대한 소유권을 보장할 수 있는 환경을 제공한다. 또한 스마트 컨트랙트의 도입으로 자동화된 접근 권한 설정을 신뢰할 수 있는 제 3자 없이 구성할 수 있다.

이러한 특성을 활용하여, CSP에 기반한 재해 복구 시스템을 블록체인으로 구성할 때는 다음과 같은 장점을 얻을 수 있다. 첫 번째는 CSP에 대한 중앙화와 의존성의 제거이다. 기존에 제안된 클라우드 기반 재해 복구는 데이터를 제공하는 기업이 CSP의 운영 및 존속 여부를 고려해야 하며, CSP가 신뢰할 수 있는 사업자인지, 내부자에 대한 위협과 또다른 장애 여부의 상황을 고려하여 데이터 보호 전략을 세워야만 한다. 두 번째는 개인 데이터에 대해 사용자가 소유권을 가지고 데이터를 통제하고 제공 할 수 있는 환경을 제공할 수 있다. 고객은 대부분의 서비스 기업에게 정보를 위탁하며, 해당 데이터에 대한 CSP의 복제 여부에 대해 관여할 수 없는 환경으로, 원하지 않는 데이터 유출과 손실에 대한 위협에 잠재적 노출을 방지할 수 있다.

2.2 이더리움과 스마트 컨트랙트

비트코인[10], 이더리움[11]과 같은 암호 화폐에서 사용되는 블록체인은 네트워크 참여자들의 합의 프로토콜에 의해 탈중앙화 환경에서 원장을 구성한다. 누구나 합의에 참여할 수 있는 무허가형(Permissionless) 블록체인은 단일 지점 장애가 없으며, 과거 내역에 대한 위변조가 어려운 비가역성을 가진다. 따라서 블록체인에 저장된 트랜잭션과 상태에 대한 신뢰성을 확보할 수 있다.

스마트 컨트랙트는 블록체인에서 실행되는 프로그램으로, 이더리움의 스마트 컨트랙트 언어인 솔리디티[12]와 같은 언어로 개발할 수 있다. 솔리디티는 튜링 완전 언어로, 컴파일된 바이트코드를 EVM(Ethereum Virtual Machine)에서 참여자들이 실행하여 상태를 업데이트한다.

EVM에서 스마트 컨트랙트 실행 시 반복을 통한 서비스 거부 공격의 대비책으로 부과되는 비용을 gas라고 한다. 이를 위해 사용자들은 스마트 컨트랙트를 실행하기 위한 트랜잭션을 블록체인으로 전송할 때 연산 비용에 대한 수수료를 함께 제시해야 한다. 따라서, 스마트 컨트랙트로 구성된 탈중앙화 어플리케이션에서 가스 비용은 서비스를 평가할 수 있는 주요 지표이다.

블록체인에 저장되는 데이터의 크기가 클수록 저장 비용으로 gas가 많이 부과되며, 이를 최소화하기 위하여 분산 스토리지를 활용할 수 있다. IPFS(InterPlanetary File System)[13]와 같은 분산 스토리지 플랫폼은 저장되는 파일에 대한 해시 값을 나타내는 CID(Content ID)를 활용하여 데이터를 식별한다. 스마트 컨트랙트에 CID 값을 저장하여 분산 스토리지와 연동함으로써, 저장 비용을 감소시킬 수 있다.

2.3 프록시 재암호화

분산 스토리지는 블록체인 플랫폼에 비용 효과적인 데이터 저장소를 제공하지만, 개인의 데이터를 공유하기 위해 필요한 기밀성을 탈중앙화 환경에서 구현하기 어려움이 있다. 본 논문에서는 프록시 재암호화를 활용하여 분산 스토리지의 기밀성을 제공한다. 기밀성을 달성할 수 있는 수단 중, 프록시 재암호화는 복호화 권한을 위임할 수 있는 알고리즘이기 때문에 데이터 복구 과정에서 사용자의 개입 없이 절차를 구성 할 수 있는 장점이 있다.

프록시 재암호화 알고리즘은 Fig. 2와 같이 데이터 소유주 Alice의 공개키로 암호화한 데이터를 수신자 Bob의 개인키로 복호화 할 수 있도록 구성하기 위한 알고리즘이다. 이를 위해서 암호화된 데이터를 재암호화하여 Bob에게 전달하는 중개자인 프록시가 필요하다. 프록시에게 전달되는 데이터는 Alice의 공개키로 암호화되어 있고, 재암호화된 데이터는 Bob의 개인키로만 복호화할 수 있기 때문에 데이터 전달과정에서 기밀성이 유지된다.

프록시 자체를 하나의 서버 및 시스템으로 구성할 경우 단일 지점 장애 문제가 생길 수 있기 때문에, 본 논문에서는 평판 기반의 프록시 운영을 제안한다. 평판 시스템은 스마트 컨트랙트에 의해서 구현가능하

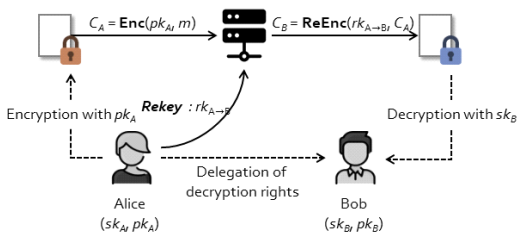

Fig. 2. Proxy Re Encryption Algorithm.

며, 기본적인 예치 및 처벌 구조로 구성되어 시스템에 등록된 노드는 예치금을 걸어두고, 재암호화 수행을 통해 이익을 증가시킨다. 이와 같은 평판 시스템 모델은 오라클 시스템을 구현한 Augur[14]나 Chainlink[15]와 같은 플랫폼과 유사하다.

프록시 재암호화 알고리즘의 유형은 A와 B의 암호화 방향 여부에 따라 단방향 및 양방향으로 구분할 수 있다. 제안하는 방식은 유형에 상관없이 프록시 재암호화 알고리즘을 적용할 수 있다.

III. 제안하는 방법

3.1 구성 요소

제안하는 시스템은 Fig. 3과 같이 구성된다. 각 구성요소에 대한 설명은 다음과 같다.

User(U): 데이터의 소유주인 사용자는 서비스 서버에 데이터를 제공한다. 또한, 개인정보와 관련된 내역들은 암호화하여 분산 스토리지에 보관한다. 사용자는 이더리움 주소 $addr_U$를 가지고 있으며, 데이터에 대한 접근 정책을 구성하기 위해 데이터 복구에 동의하기 위해 스마트 컨트랙트를 호출하는 트랜잭션을 작성한다.

Server(S): 서비스 제공자는 사용자에게 제공하는 서비스를 위한 환경을 가지고 있다. 사용자에게 제공받은 데이터를 가지고 있으며, 재해 복구 대비책을 갖고자 하는 주체에 해당한다. 서비스 제공자가 운영하는 서버는 사용자와 마찬가지로, 이더리움 주소 $addr_S$를 가지고 있으며, 재난 발생 시 스마트 컨트랙트를 호출하여 복구를 요청한다.

Smart Contract(SC): 스마트 컨트랙트는 이더리움 내에서 구현되어 있다. 스마트 컨트랙트는 사용자의 정책과 서비스 서버의 요청을 관리한다. 추가적으로, 프록시가 재암호화 수행을 올바르게 할 수 있도록 하기 위한 평판 관리가 있다. 스마트 컨트랙트의 구성은 3.5에서 자세하게 설명한다.

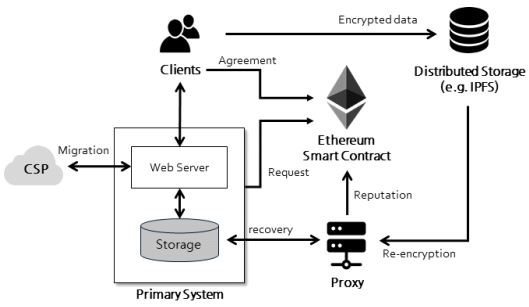

Fig. 3. Blockchain Based DR System.

Proxy(P): 프록시 노드는 분산 스토리지의 사용자 데이터를 재암호화하여 서비스 제공자에게 전달하는 역할을 한다. 프록시 노드는 이더리움 주소를 가지고 있으며, 노드에서 분산 스토리지에 접근 할 수 있는 클라이언트가 구축되어 있어야 한다.

Distributed Storage(DS): 분산 스토리지에는 암호화된 사용자의 데이터가 저장된다. 스토리지에 저장된 데이터는 CID를 고유한 식별자로 가진다. 식별자를 통해 누구나 암호화된 데이터에 접근할 수 있으나 복호화할 수는 없다.

위에서 제시한 구성 요소를 두고, 제안하는 방식은 크게 3가지 목적을 중심으로 설계되었다. 복구 단계에서 사용자의 개입을 요구하지 않도록 자동화가 가능하도록 구현되었으며, 이더리움의 트랜잭션 비용을 최소화하여 구성하였다. 또한, 스마트 컨트랙트가 접근 제어에 대한 역할을 담당함으로써, 모든 요청과 관리가 신뢰성을 가질 수 있도록 구성하였다.

3.2 준비 과정

프로토콜을 구성하는 사용자와 서버, 프록시는 이더리움 계정을 가지고 있다. 이더리움 계정은 각 주체가 생성한 이더리움 키 쌍의 공개키 해시 값에 해당한다. 또한, 이들은 서로 간의 계정으로 송금을 하거나 식별할 수 있도록 주소를 공유하고 있다. 이외에도 프로토콜에 필요한 스마트 컨트랙트는 배포되어 있으며, 스마트 컨트랙트의 주소를 서로 공유하고 있다.

반면, 이들 간의 공유하고 있는 대칭키는 없으며, 기밀성은 프록시 재암호화에 의해서 유지된다. 프록시 재암호화의 대상이 되는 사용자의 데이터는 사용자의 공개키로 암호화되며, 서버의 개인키로 복호화할 수 있도록 구성된다. 이를 위한 프록시 재암호화 키 쌍을 사용자와 서비스 제공자가 각각 생성한다.

1) 사용자 키쌍 생성(U) : 사용자는 자신의 대칭키를 암호화하기 위한 프록시 재암호화 개인키, 공개키 쌍 sk_U, pk_U을 생성한다.

2) 서비스 제공자 키쌍 생성(S) : 서비스 제공자는 데이터 복호화를 위한 개인키, 공개키 쌍 sk_S, pk_S을 생성한다.

3) 사용자별 프록시 등록(U) : 데이터 백업에 앞서, 재암호화를 수행할 프록시의 목록에 대한 이더리

움 주소 $addr_{P'}$를 사용자가 등록하는 절차가 필요하다. 분산된 환경의 프록시는 평판 시스템에 의해 운영되기 때문에, 사용자는 평판에 따라 특정 프록시만 재암호화에 참여하도록 구성할 수 있다. 프록시의 계정 주소를 스마트 컨트랙트에 등록 한 후, 등록된 프록시에 한해서만 재암호화 수행에 대한 보상이 지급되도록 구성한다. 이를 위해 사용자별로 사전에 프록시를 설정한다.

3.3 데이터 백업

데이터 백업은 서버가 재해 발생 이후 데이터를 복구할 수 있도록 사용자가 암호화된 데이터를 저장하고, 서버의 복구에 동의하는 절차로 구성된다. 상세 절차는 Fig. 4와 같으며 설명은 다음과 같다.

1) 복구 동의 요청($S \rightarrow U$) : 서버는 사용자의 데이터를 재해 발생 이후 복구할 수 있도록 사용자에게 동의를 요청한다. 이 때, 사용자가 동의할 경우 재암호화 키를 생성할 수 있도록 서버는 공개키 pk_S를 전달한다.

2) 데이터 암호화(U) : 사용자가 서버의 백업 요청에 동의할 경우, 사용자는 데이터 m을 사용자의 공개키로 암호화한 데이터 $c_1 = Enc(pk_U, m)$와 재암호화 키 $rk_{U \rightarrow S} = ReKeyGen(sk_U, pk_S)$를 생성한다.

3) 데이터 업로드 ($U \rightarrow DS$) : 사용자는 랜덤 값 r을 생성 후 암호화된 데이터의 해시 값 $h_c = Hash(c_1)$와 결합한 해시 값 $s = Hash(h_c \| r)$을 생성하고, 분산 스토리지에 저장할 번들 $bundle = c_1, h_c, r$을 구성하여 업로드 한다. 여기서 r은 가장 먼저 재암호화를 수행할 프록시를 선택하기 위한 값으로 사용된다. 분산 스토리지에 저장된 데이터는 식별 값 $cid_{bundle} = Hash(bundle)$를 반환한다.

4) 스마트 컨트랙트 호출($U \rightarrow SC$) : 사용자는 업로드한 데이터의 cid_{bundle}와 서버의 이더리움 계정 주소 $addr_S$, 재암호화키 $rk_{U \rightarrow S}$ 와 s를 파라미터로 스마트 컨트랙트를 호출하는 트랜잭션을 제출한다. 호출된 스마트 컨트랙트는 데이터 복구 시 필요한 권한을 정책으로 작성한다. 스마트 컨트랙트로 제출한 트랜잭션이 실행되면, 결과를 사용자와 서버가 확인하여 등록된 결과를 스마트 컨트랙트의 로그에 해당하는 이벤트로 확인할 수 있다.

3.4 데이터 복구

백업된 데이터는 재해 발생 후, 서버의 개인키로 암호화 될 수 있는 상태로 반환되어야 한다. 사용자가 서버의 데이터 활용에 대한 동의를 취소하기 위해 스마트 컨트랙트의 상태를 변경하지 않는 한, 서버에게 데이터 복구를 동의한 것으로 간주하여 프록시는 재암호화를 수행하여 전달한다. 상세한 절차는 Fig. 5와 같으며 설명은 다음과 같다.

1) 데이터 복구 요청($S \rightarrow SC$) : 서비스 제공자는 스마트 컨트랙트에 데이터 복구를 요청하는 트랜잭션을 제출한다. 트랜잭션의 송신자인 서버의 이더리움 주소 $addr_S$가 스마트 컨트랙트에 등록된 주소일 경우, 스마트 컨트랙트에서는 프록시가 확인할 수 있는 이벤트를 발생시킨다. 스마트 컨트랙트에서 프록시로 전달되는 이벤트에는 요청자 정보 $addr_S$과 분산 스토리지의 식별 값 cid_{bundle}, 재암호화 키 $rk_{U \rightarrow S}$로 구성된다.

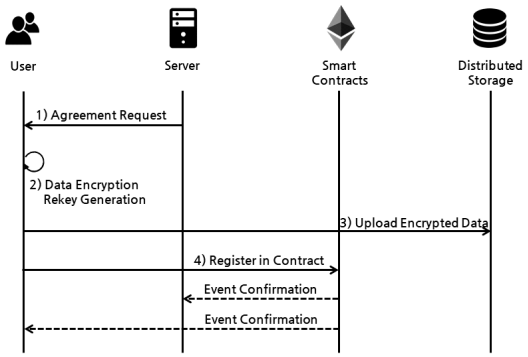

Fig. 4. Data Backup Process.

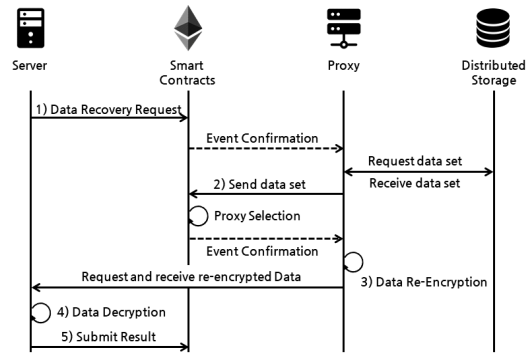

Fig. 5. Data Recovery Process.

2) 재암호화 프록시 매칭 ($P \rightarrow SC$) : 서버의 데이터 복구 요청 이벤트를 확인한 프록시는 이벤트로 확인한 cid_{bundle}로 분산 스토리지에 접근하여 암호화된 데이터 c_1와 해시 값 h_c을 찾아 함께 저장된 랜덤 값 r을 확인 후, 스마트 컨트랙트로 제출한다. 스마트 컨트랙트에 저장된 s와 프록시가 제출한 r 값으로 계산된 $s_P = Hash(h_c \| r)$가 일치하면 가장 먼저 제출된 트랜잭션에 의해 재암호화를 수행할 프록시가 선택된다. 선택된 프록시는 이벤트를 통해 확인할 수 있다.

3) 재암호화 데이터 전송($P \rightarrow S$): 선정된 프록시는 암호화된 데이터를 재암호화한 $c_2 = ReEnc(rk_{U \rightarrow S},\ c_1)$ 를 서버에 전달한다.

4) 데이터 복호화(S): 재암호화된 데이터 c_2를 받은 서버는 개인키로 데이터를 복호화하여 원본 데이터 $m = Dec(sk_S,\ c_2)$을 획득한다.

5) 최종적으로 서버는 프록시에 대한 평판을 스마트 컨트랙트에 작성하여 제출한다.

3.5 스마트 컨트랙트 구성

본 논문에서 제시하는 스마트 컨트랙트는 사용자 데이터에 대한 정책 관리, 데이터 복구를 위한 요청 관리, 프록시에 대한 평판 관리 기능을 가진다. 순서에 따라 호출하는 대상은 사용자, 서버, 프록시로 구분된다. 각 부분에 대한 설명은 알고리즘으로 나타낸 Fig. 6~8과 함께 확인할 수 있다.

3.5.1 데이터 복구 권한에 대한 사용자 동의

사용자에 의해 호출되는 스마트 컨트랙트는 서버의 데이터 복구에 동의하는 절차와 취소하는 절차로 구성된다. 이에 앞서 스마트 컨트랙트에서는 사용자를 등록하고, 사용자 별로 지정된 프록시를 지정하는 준비가 필요하다. 초기 등록과정에서 호출된 스마트 컨트랙트는 호출자를 owner로 지정하여 호출한 당사자만이 데이터를 등록 및 취소할 수 있도록 권한을 부여한다. 사용자가 프록시를 등록할 때에는 예치금을 지불하여 등록된 프록시에 한해서 지정이 가능하다. 프록시 목록에는 적어도 하나 이상의 프록시가 포함되어야 한다.

데이터 복구에 대한 동의 시, 사용자의 트랜잭션에는 cid_{bundle}, 서버의 이더리움 주소 $addr_S$, 재암호화 키 $rk_{U \rightarrow S}$, 랜덤 해시 값 s를 파라미터로 스마트 컨트랙트를 호출한다. 각 파라미터들은 사용자가 관리하는 정책에 값으로 저장되어, 이 후에 데이터 복구를 요청하거나 응답할 때 실행하는 권한을 확인할 때 활용된다. 사용자가 서버에 대해 데이터 활용 동의를 취소하고자 할 때는 스마트 컨트랙트를 호출하여 기존에 등록한 요청에 대한 데이터를 삭제한다. 삭제 권한은 정책을 생성한 소유주인 사용자만이 가지고 있다. 권한을 삭제함으로써, 추후 데이터 요청이나 응답을 할 때 필요한 권한을 제거할 수 있다.

Algorithm 1: Smart contract for recovery agreement

```
function initUser() public :
    u ← userList[msg.sender];
    require(!u.exists);
    u.owner ← msg.sender;
    u.exists ← true;
    return

function addProxy(pa) public onlyOwner :
    u ← userList[msg.sender];
    require(numOfProxy < MAX_NUM_PROXY);
    u.proxyList[pa].registered ← true;
    u.numOfProxy++;
    return

function removeProxy(pa) public onlyOwner :
    require(u.numOfProxy != 0);
    u ← userList[msg.sender];
    delete u.proxyList[pa];
    u.numOfProxy--;
    return

function recoveryAgree(cid, sa, rk, s) public onlyOwner :
    u ← userList[msg.sender];
    require(!u.policy[cid].registered);
    u.policy[cid].serverAddress ← sa;
    u.policy[cid].rekey ← rk;
    u.policy[cid].select ← s;
    u.policy[cid].registered ← true;
    return

function cancel(cid) public onlyOwner
    u ← userList[msg.sender];
    require(u.policy[cid].registered);
    delete u.policy[cid];
    return
```

Fig. 6. Smart contract for recovery agreement.

3.5.2 서버에 의한 데이터 복구 요청 관리

서버는 데이터 복구를 요청할 때 스마트 컨트랙트를 호출한다. 요청한 서버의 이더리움 주소가 요청한

Algorithm 2: Smart contract for request management

```
function request(ua, cid) public
    u ← userList[ua];
    r ← requestList[cid];
    require (u.policy[cid].serverAddress = msg.sender);
    r.requester ← msg.sender;
    r.userAddress ← ua;
    r.startTime ← now;                    // block.timestamp
    emit request(cid, sa, rk);        // monitoring by Proxy
    return

function response(cid, hc, rp) public onlyProxy
    r ← requestList[cid];
    u ← r.userAddress;
    p ← proxyList[msg.sender];
    require(u.proxyList[msg.sender].registered)
    if u.select == Hash(hc, rp) then
       r.selectedProxy ← msg.sender;
       r.endTime ← now;                   // block.timestamp
       p.numOfResponse++;
    else
       p.rep -= RES_ERROR;
    return
```

Fig. 7. Smart contract for request management.

데이터의 cid_{bundle}에 대해 복구 권한이 허가된 주소일 경우, 요청 리스트를 생성한다. 요청 리스트의 값에는 요청자와 사용자의 주소, 요청한 시간이 기록된다. 요청 함수의 실행이 완료되면, cid_{bundle}와 호출자인 서버의 이더리움 주소 $addr_S$, 할당된 재암호화 키 $rk_{U\to S}$를 파라미터로 하는 이벤트에 의해, 모니터링 하는 프록시가 확인하여 응답을 준비한다.

프록시가 재암호화를 수행을 위해 분산 스토리지에서 발견하여 요청 cid_{bundle}대한 h_c와 r을 스마트 컨트랙트로 제출한다. 이후, 호출된 함수에 의해서 사용자에 의해 등록된 프록시인지, 요청에 대해 이미 선정된 프록시가 없는지 확인한다. 최종적으로 스마트 컨트랙트 내에서 프록시가 제출한 값을 대상으로 스마트 컨트랙트에서는 $s_P = Hash(h_c \| r)$ 값을 계산하여 사용자가 등록한 s값과 일치하는지 확인한다. 일치할 경우, 프록시가 선정되고 응답 시간 결정 후 응답 횟수를 추가한다. 응답이 올바르지 않을 경우, 잘못된 응답에 대한 평판을 감소시킨다.

3.5.3 프록시 선정 및 평판 관리

프록시는 재암호화를 제공하기에 앞서 평판 관리를 위해 등록이 필요하다. 등록을 위해 프록시는 예치금을 정해진 금액 이상 걸어두어야 하며, 프록시 리스트에 등록된 후 평판을 초기화한다. 등록된 프록시는 사용자가 리스트에 추가할 수 있는 상태가 된다.

Algorithm 3: Smart contract for proxy management

```
function registerProxy(cid) public payable
    p ← proxyList[msg.sender];
    require(!p.registered && msg.value > MIN_DEPOSIT);
    p.deposit ← msg.value;
    p.registered ← true;
    p.rep ← 0;
    p.resTime ← 0;
    return

function submitReputation(cid, rep) public payable
    r ← requestList[cid];
    require (r.requester == msg.sender)
    if msg.value < MIN_REWARD then
       payable(msg.sender).transfer(msg.value)
    else
       payable(r.selectedProxy).transfer(msg.value)
       proxyList[r.selectedProxy].rep ← rep;
       delete r
    return
```

Fig. 8. Smart contract for proxy management.

데이터 복구를 완료한 후, 서버는 프록시에 대한 평판을 제출하기 위해 스마트 컨트랙트를 호출한다. 복구를 요청한 cid_{bundle} 에 대해 재암호화를 수행한 프록시에 대한 평판을 전송할 때 관련 비용을 지불하고 요청을 삭제한다.

IV. 구 현

본 절에서는 제안하는 시스템의 구현에 대해서 설명한다. 구현 범위는 프록시 재암호화를 구성하는 각 개체의 시스템과 블록체인에 배포된 스마트 컨트랙트로 나뉜다.

프록시 재암호화 알고리즘으로는 [16]에서 제시한 알고리즘을 사용한다. 해당 알고리즘은 Identity 기반의 프록시 재암호화로, rust 언어로 구현된 recrypt API[17]를 제공한다. recrypt API에서 구현된 알고리즘은 재암호화 속도 측면에서 장점을 가진다. 분산 스토리지로는 IPFS의 go언어 버전인 go-ipfs[18]를 활용하였다. 테스트 수행 프로그램은 프록시 재암호화 알고리즘과 동일하게 rust 언어로 구현되었고, 컴파일러는 rustc 1.56.0 버전을 사용하였다. 프록시 재암호화가 수행되는 환경은 Intel(R) Core(TM) i7-9700F @3.00GHz CPU와 32GB

메모리이며, 운영체제는 Ubuntu 18.04.5 LTS이다. 3.5에서 설명한 스마트 컨트랙트는 이더리움 플랫폼에서 가장 많이 활용되는 Solidity로 구현하였으며, 컴파일러는 solc 0.8.4 버전을 사용하였다. 스마트 컨트랙트 배포 이더리움 테스트넷인 Kovan Network[19] 하였고, 베를린 하드포크 기준 EVM을 활용하여 비용을 측정하였다.

V. 실험 및 평가

본 절에서는 제안하는 시스템의 성능과 비용, 보안성에 대해 실험한 결과와 평가를 나타내었다.

5.1 성능 평가

재해 복구 시스템은 데이터의 재난 발생 전 백업 절차와 재난 발생 후의 복구 절차를 구성해야한다. 데이터의 백업 과정에서는 주 시스템에서 서브시스템으로 데이터의 복제가 발생한다. 데이터가 복제되는 주기인 RPO(Recovery Point Objective)는 Fig. 9에서 보는 것과 같이 백업 주기를 나타낸다. 재난이 발생할 경우 마지막 백업 시점까지 복구가 가능하다.

복구 절차는 가장 최근에 백업된 시점의 데이터를 주 시스템으로 복구하는 시간이 중요하다. RTO (Recovery Time Objective)는 재난 발생 후 복구되기까지 걸리는 시간을 의미하며, 재해 복구 시스템에서 중요하게 여겨지는 지표이다. 재해 발생 시 대응 시간을 나타내는 RTO의 수준에 따라 Table 1과 같이 핫/웜/콜드 시스템으로 구분한다[5]. 복구에 필요한 시간인 RTO의 수준이 높을수록 비용이 높게 책정된다. 본 시스템에서는 데이터 요청 시점부터 복구까지의 시간을 RTO로 정의한다. RTO는 다음 3가지 값의 총 합에 해당한다.

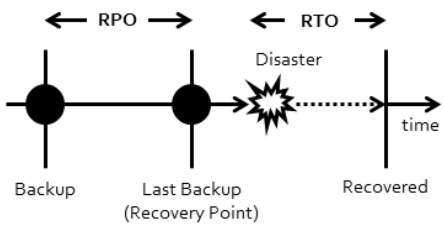

Fig. 9. RPO and RTO in DR System.

Table 1. Type of Disaster Recovery System

Opt.	RTO Coverage	Cost
Hot Site	5 min - 4 hrs	High
Warm Site	4-24 hrs	Medium
Cold Site	Days 1-7	Low

$$RTO = T_{S \to SC} + T_{P \to SC} + T_{P \to S} \quad (1)$$

$T_{S \to SC}$는 서버의 데이터 복구 요청 트랜잭션 발생부터, 프록시가 이를 확인하기까지의 시간을 나타낸다. $T_{P \to SC}$는 프록시가 요청을 확인 후 분산 스토리지에 접근하여 스마트 컨트랙트로부터 선정되는 이벤트를 확인하기까지 소요되는 시간을 나타낸다. $T_{P \to S}$는 선정된 프록시가 재암호화를 수행하여 서버에 전송 후, 서버가 데이터를 복호화 하는 시간을 측정한다.

$T_{S \to SC}$과 $T_{P \to SC}$는 이더리움 네트워크의 블록 생성 시간과 프록시가 IPFS로부터 저장되어 있는 암호화된 데이터를 다운로드 시간이 포함되기 때문에 블록 생성 시간과 데이터 다운로드 시간을 더하여 측정하였고, $T_{P \to S}$는 재암호화 후 데이터를 전송하여 복호화하기까지 걸리는 시간이 포함되므로 recrypt API를 이용해 128bit의 보안 강도를 기준으로 성능을 측정하였다. 이더리움 네트워크의 블록 평균 생성 시간은 데이터의 무관하며 최근 100개 블록의 평균 생성시간은 13.42초이다. 파일 다운로드, 재암호화, 복호화에 대한 시간 측정 결과는 Table 2와 같이 나타내었으며 측정된 각 결과는 동일한 파일에 대해 100회 수행에 대한 평균값을 나타낸다. IPFS 다운로드는 캐싱된 상태에서 측정한 결과이다.

실험 결과를 바탕으로 재난 복구 성능 지표인 RTO를 계산해보면 데이터 크기가 0.1KB일 때

Table 2. Measurement of times by data size

	0.1KB	1KB	10KB
IPFS Download	0.13s	0.21s	0.25s
ReEnc	0.68s	2.05s	21.3s
Dec	0.87s	2.62s	27.3s
RTO	15.1s	18.3s	62.27s

15.1초, 1KB일 때 18.3초, 10KB일 때 62.27초임을 알 수 있다. 측정 시간은 384byte를 기준으로 선형적으로 증가한다. 따라서 대용량 암호화가 필요한 데이터의 경우, 하이브리드 암호화를 활용하여 대칭키 값에 대해서 프록시 재암호화를 쓰는 것이 효율적이다.

5.2 비용 분석

스마트 컨트랙트에서 비용은 트랜잭션에 부과한 가격으로 설정된다. Table 3에서 제시한 값은 EVM에서 산정하는 함수별 실행 gas 비용을 나타낸다. gas 비용이란, 이더리움 노드들에서 트랜잭션별로 호출된 함수를 실행한 후, EVM 내에 정의된 명령에 따라 부과되는 비용을 의미한다. 해당 비용은 동일한 EVM 내에서 고정된 비용이며, 실제 트랜잭션에 사용되는 gas 비용은 사용자가 부과한 gas 가격에 따라 달라질 수 있다.

재암호화 서비스를 위한 컨트랙트를 하나로 구성하여 배포할 경우, 필요한 gas 비용은 2,789,868로 측정되었다. 각 호출자(Caller)에 따라 수행하는 함수에 대한 비용은 대부분 100,000 gas 아래로 나타난다. 가장 많은 비용을 처리하는 프록시가 응답을 호출하는 비용은 가스 가격을 27Gwei 기준으로 하였을 때, 약 $10.92만큼 소모된다. 또한, 가장 낮은 취소 트랜잭션은 $1.72 만큼의 비용을 지출한다. 트랜잭션 비용을 고려하면 보다 더 측정될 수 있지만, 전체적으로 개인 당 $1~10 수준의 범위에서 서비스를 수행할 수 있다.

5.3 보안 분석

1) 가용성: 제안된 시스템의 가용성은 크게 두 가지로 구분할 수 있다. 첫 번째는 단일 지점 장애에 대비한 시스템 가용성이며, 두 번째는 분산 환경의 프록시 운영에 대한 가용성이다.

제안하는 방법의 주된 목적 중 하나인 재해 복구 상의 데이터를 처리할 때 발생하는 클라우드 의존성과 사용자 중심으로 데이터를 관리하는 것이다. 블록체인은 탈중앙화 네트워크로 작동하며, 접근 제어를 수행하는 스마트 컨트랙트와 데이터가 저장되는 분산 스토리지의 도입은 단일 지점 장애가 발생하지 않는 환경을 제공한다.

프록시의 목적은 크게 2가지로 분류할 수 있는데, 첫 번째 목적은 데이터의 전달(Reporting)이다. 프록시를 통한 데이터의 전달은 프록시 재암호화가 사용자의 권한을 위임받는 과정을 통해, 복구 절차에서 사용자를 배제함으로써 RTO를 줄이는데 영향을 미친다. 두 번째는 서버의 데이터 복호화 기능 제공으로, 서버가 복구 절차에서 사용자와 통신하여 키 교환을 하지 않아도 위임된 프록시에 의해 데이터를 전달받아 복호화를 진행할 수 있도록 한다.

이러한 환경에서 프록시는 복구 절차의 가용성에 큰 영향을 미치는 요소이다. [20]과 [21]같은 블록체인 기반의 시스템에서는 프록시를 단일 노드로 환경을 구축하여 운영하며, 이러한 시스템은 단일 지점 장애를 가진다. 이러한 문제를 해결하기 위해서, 스마트 컨트랙트로 구현된 평판 시스템의 도입은 사용자에게 프록시에 대해 공개된 과거 이력과 현재의 상태를 통해 종합적인 신뢰도를 제공하여 선택을 유도할 수 있다. 본 논문에서는 프록시로 참여하고 평판을 측정하는 기본적인 환경을 구성하였으나, 분산 프록시 환경의 가용성 보장을 위해 다음과 같은 기능 추가를 고려할 수 있다.

평판 정책: 평판 관리는 프록시의 행위 정보로 구성된다. 이에 대한 파라미터로는 프록시가 예치한 금액, 처리에 성공한 요청 및 응답 수, 데이터 평균 응답 시간, 패널티에 의한 금액, 미완료 건 내역 등이 추가 될 수 있다.

인센티브 메커니즘: 탈중앙화 환경의 프록시의 운영에서 신뢰할 수 있는 운영환경을 위해서 인센티브 메커니즘을 활용할 수 있다. 인센티브 메커

Table 3. Gas Cost about Smart Contract Implementation

Function Name	Caller	Cost(gas)	Function Name	Caller	Cost(gas)
Deployment	-	2,789,868	cancel	User	20,555
userInit	User	45,460	request	Server	95,967
addProxy	User	70,991	response	Proxy	130,007
removeProxy	User	21,711	registerProxy	Proxy	56,819
recoveryAgree	User	94,881	submitReputation	Server	38,517

니즘은 선정된 프록시의 정확한 행위에 대한 보상과 잘못된 행위에 대한 처벌을 강제한다. 이를 통해 프록시가 올바로 시스템에서 보장된 활동을 하도록 동기를 제공하고 합리적인 선택을 강제함으로써 보안성을 확보할 수 있다.

진위성 판정 도구: 진위성(Authenticity)을 판정하는 도구는 인센티브 메커니즘을 통해 판정할 때 발생하는 비용을 줄이기 위한 목적으로 활용된다. 인센티브 시스템을 통한 결정은 올바로 작동할 것을 가정하고, 이에 대해 합리적인 참여자에 대해 올바르게 작동하도록 유도하지만, 비정상 작동이 발생할 경우 추가적인 판정 단계를 요구한다. 판정 단계에서 추가적인 투표를 진행하여 2중 투표 시스템으로 진행하는 방법[14]이나, 데이터에 대한 결과를 스마트 컨트랙트에서 검증할 수 있도록 프로토콜을 도입한[15]와 같은 방식을 활용할 수 있다.

2) 기밀성: 제안하는 방법에서는 분산 스토리지에 프록시 재암호화로 암호화된 상태의 데이터를 저장한다. 따라서, 사용자의 정보가 포함된 원본 데이터는 분산된 환경에서 드러나지 않으며, 프록시가 재암호화를 수행하는 동안에도 여전히 기밀성이 유지된다. 데이터는 재암호화 키를 생성 할 때 참여한 서버의 개인키로만 복호화 할 수 있으며, 사용자의 허가를 통해 복구가 가능하다.

3) 무결성: 블록체인의 합의는 기록된 트랜잭션에 비가역성을 부여하여 무결성을 유지한다. 특히, 스마트 컨트랙트에 구현된 로직은 신뢰할 수 있는 실행을 보장한다.

4) 인증: 블록체인에서 인증은 개인키의 소유권으로 관리된다. 사용자와 서버, 프록시는 각 계정으로 생성한 개인키를 통한 트랜잭션으로 등록 및 취소 과정을 수행하며, 한번 등록된 이 후 스마트 컨트랙트에 의해 트랜잭션에 따라 인증 절차는 자동적으로 수행될 수 있다.

VI. 관련 연구

[2-7]에서는 클라우드 환경의 재해 복구 시스템에 대해 제안하였다. 기존의 연구 방식은 클라우드 환경의 재해 복구에서 주요 이슈중 하나인 의존성을 제거하기 위해 추가적인 백업 및 복구 프로세스를 제시하거나[4][5], 단일 지점 장애에 대응하기 위해 분산된 환경에서 클라우드 기반의 재해 복구를 위한 프로세스를 제시한다[6][7]. 기존의 방식은 클라우드 기반의 재해 복구 시스템을 운영하면서 문제를 해결하는 반면, 제안된 방식은 블록체인 기반으로 제시되어 클라우드 서비스 제공자의 의존성을 제거하였다.

블록체인에서 기밀성을 위해 프록시 재암호화를 방식은 블록체인에 활용한 사례로는 [20][21]이 있다. 해당 논문에서는 IoT 환경에서 데이터를 공유할 때 발생하는 기밀성 문제를 프록시 재암호화를 통해 해결한다. 하지만, [20] 에서는 프록시를 신뢰할 수 있다고 가정하고 단일 노드로 운영하고 있어, 단일 지점 장애가 발생할 수 있다. 또한, [21]에서는 데이터를 보관하고 있는 CSP가 프록시로 활동하는 방식으로 운영된다. 이 같은 방법은 CSP가 데이터의 재암호화를 수행하기 때문에 기밀성이 유지되지만, CSP가 블록체인 환경에 참여해야 되는 문제로 이어지기 때문에 비즈니스 전략이 고려되어야 한다. 반면, 제안된 방법은 프록시를 신뢰하거나 CSP의 참여를 유도하지 않는 환경을 고려하여 제시되었다.

VII. 결 론

본 논문에서는 클라우드 환경에서 제공되는 재해 복구 시스템에 대해 CSP 의존성을 제거하고 사용자 중심의 데이터 관리가 가능한 블록체인 기반의 재해 복구 시스템을 제안하였다. 제안된 방법은 분산 스토리지에 데이터를 저장하고 백업 및 복구함으로써 CSP의 데이터 저장을 대체하고, 스마트 컨트랙트를 통해 신뢰할 수 있는 제 3자 없이 사용자가 데이터 활용 내역에 대해 제어할 수 있도록 구현하였으며, 데이터의 기밀성을 위하여 프록시 재암호화를 사용하였다. 제안된 방법은 이더리움과 IPFS, recrypt 알고리즘을 사용하여 구현하고 재해 복구에 활용하기 위해 필요한 성능과 비용을 함께 제시하였다. 제안된 방법을 활용하여 재해 복구 환경을 구축하면, 사용자가 데이터의 제어권을 가질 수 있고, 서비스 기업에서는 CSP에 의존하지 않고 재해 복구 계획을 구현할 수 있는 장점을 가진다. 추가적인 연구로 탈중앙화된 환경의 프록시 운영에 필요한 평판 정책의 구성과 인센티브 메커니즘, 판정 도구와 같은 연구와의 결합을 통해 향상된 신뢰성 및 가용성을 확보할 수 있을 것으로 예상한다.

References

[1] Brooks, Charlotte, et al, "IBM system

storage business continuity solutions overview," IBM Redbooks, Feb, 2007.
〔2〕 Pokharel, Manish, Seulki Lee, and Jong Sou Park, "Disaster recovery for system architecture using cloud computing," 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet, pp. 304-307, Jul, 2010.
〔3〕 Jian-Hua, Zhang, and Zhang Nan, "Cloud computing-based data storage and disaster recovery," 2011 International Conference on Future Computer Science and Education, pp. 629-632, Aug, 2011.
〔4〕 Javaraiah, Vijaykumar, "Backup for cloud and disaster recovery for consumers and SMBs," 2011 Fifth IEEE International Conference on Advanced Telecommunication Systems and Networks (ANTS), pp.1-3, Dec, 2011.
〔5〕 Gu, Yu, Dongsheng Wang, and Chuanyi Liu, "DR-cloud: Multi-cloud based disaster recovery service." Tsinghua Science and Technology, vol.19, no.1, pp.13-23, Feb, 2014.
〔6〕 Sengupta, Shubhashis, and K. M. Annervaz, "Planning for optimal multi-site data distribution for disaster recovery," International Workshop on Grid Economics and Business Models, pp.161-172, Dec, 2011.
〔7〕 Lenk, Alexander, and Stefan Tai, "Cloud standby: disaster recovery of distributed systems in the cloud," European Conference on Service-Oriented and Cloud Computing, Springer, Berlin, Heidelberg, pp.32-46, Sep, 2014.
〔8〕 Abualkishik, Abedallah Zaid, Ali A. Alwan, and Yonis Gulzar, "Disaster recovery in cloud computing systems: An overview," International Journal of Advanced Computer Science and Applications, vol.11, no.9, pp. 702-710, Sep, 2020.
〔9〕 SC MEDIA, https://www.scmagazine.com/news/-/open-aws-s3-bucket-exposes - info-on-50000-honda-india, 2021. 11.01.
〔10〕 Nakamoto, S, "Bitcoin: A peer-to-peer electronic cash system," White Paper, 2008.
〔11〕 Buterin, V, "Ethereum: A next-generation smart contract and decentralized application platform.", White Paper, 2014.
〔12〕 Solidity documentation, https://docs.soliditylang.org/en/v0.8.10/, 2021.11.01.
〔13〕 Benet, Juan, "IPFS-content addressed, versioned, P2P file system (DRAFT 3)," IACR ePring, 1407-3561, Jul, 2014.
〔14〕 Peterson, Jack, and Joseph Krug, "Augur: a decentralized, open-source platform for prediction markets," IACR ePring, 1501-01042, Jan, 2015.
〔15〕 Ellis, Steve, Ari Juels, and Sergey Nazarov, "Chainlink: A decentralized oracle network," White Paper, Mar, 2017.
〔16〕 H. Wang and Z. Cao, "A Fully secure unidirectional and multi-use proxy re-encryption scheme," proceedings of the ACM Conference on Computer and Communications Security (CCS), Poster Session, Nov, 2009.
〔17〕 recrypt-rs, https://github.com/IronCoreLabs/ recrypt-rs, 2021.11.01.
〔18〕 go-ipfs, https://github.com/ipfs/go-ipfs, 2021.11.01.
〔19〕 Etherscan Kovan Testnet Explorer, https://kovan.etherscan.io/, 2021.11.01
〔20〕 Obour Agyekum, K. O. B., Xia, Q., Sifah, E. B., Gao, J., Xia, H., Du, X., & Guizani, M, "A secured proxy-based

data sharing module in IoT environments using blockchain," Sensors, vol.19, no.5, 1235. Mar, 2019.

[21] Manzoor, A., Liyanage, M., Braeke, A., Kanhere, S. S., & Ylianttila, M. , "Blockchain based proxy re-encryption scheme for secure IoT data sharing," In 2019 IEEE International Conference on Blockchain and Cryptocurrency, pp. 99-103, May, 2019.

〈저 자 소 개〉

박 준 후 (Junhoo Park) 정회원
2014년 2월: 충남대학교 정보통신공학과 학사
2016년 2월: 충남대학교 컴퓨터공학과 석사
2016년 3월~현재: 충남대학교 컴퓨터공학과 박사과정
2020년 1월~현재: ㈜아이오트러스트
〈관심분야〉 블록체인, 보안 프로토콜, 암호 응용

김 근 영 (Geunyoung Kim) 학생회원
2018년 2월: 충남대학교 컴퓨터공학과 학사
2018년 3월~현재: 충남대학교 컴퓨터공학과 석·박통합과정
〈관심분야〉 블록체인, DID, 인증

김 준 석 (Junseok Kim) 학생회원
2021년 2월: 충남대학교 컴퓨터공학과 학사
2021년 3월~현재: 충남대학교 컴퓨터공학과 석사과정
〈관심분야〉 블록체인, 인증, 암호 응용

류 재 철 (Jaecheol Ryou) 종신회원
1985년 2월: 한양학교 산업공학과 졸업
1988년 5월: Iowa State University 전산학 석사
1990년 12월: Northwestern University 전산학 박사
1991년 2월~현재: 충남학교 컴퓨터공학과 교수
〈관심분야〉 모바일 보안, 금융보안, 블록체인