논문 25-28-02-06
KNOM Review '25-02 Vol.28 No.02
https://doi.org/10.22670/knom.2025.28.2.51

End-Fog-Cloud 환경에서 데이터 무결성과 접근 제어를 보장하는 데이터 프로비넌스 시스템

배 태 모◆, 방 지 원*, 황 향 희**, 최 미 정°

A Data Provenance System Ensuring Data Integrity and Access Control in End-Fog-Cloud Environments

Taemo Bae◆, Jiwon Bang*, Hyang-Hee Hwang**, Mi-Jung Choi°

요 약

IoT(Internet of Things)는 제한된 연산과 저장 자원을 보완하기 위해 클라우드 컴퓨팅과 결합하여 발전해왔다. 그러나 물리적 거리로 인한 지연과 실시간 처리 한계는 여전히 문제로 제시된다. 이를 해결하기 위해 Fog 컴퓨팅이 도입되면서 End-Fog-Cloud 구조로 확장되었다. 하지만 기존 보안 기법은 IoT 환경의 제약으로 적용이 어렵고, CSP(Cloud Service Provider)에 대한 신뢰 문제까지 존재해 데이터 무결성과 접근 제어 측면에서 한계를 가진다. 본 논문은 이러한 문제를 해결하기 위해 End-Fog-Cloud 전 구간을 포괄적으로 보호하는 데이터 프로비넌스 시스템을 제안한다. 제안 시스템은 블록체인을 결합한 로그 기반 프로비넌스로 무결성을 검증하고 DID (Decentralized Identifier) 및 VC/VP (Verifiable Credential/Verifiable Presentation) 기반 신원 인증을 통해 계층 간 접근 제어를 강화한다. 마지막으로 STRIDE 위협 모델을 적용해 제안하는 시스템이 주요 보안 위협을 효과적으로 탐지·차단함을 실험적으로 검증한다.

Key Words : IoT, Cloud, Blockchain, Data Provenance, Data Integrity, Access Control

ABSTRACT

The Internet of Things (IoT) has evolved by integrating cloud computing to overcome limited computational and storage resources. However, physical distance still causes network latency and limits real-time processing. To address this, fog computing was introduced as an intermediate layer, forming the End-Fog-Cloud architecture. Yet conventional security mechanisms remain unsuitable because of IoT device constraints, persistent reliance on Cloud Service Providers (CSPs). Consequently, existing approaches fail to guarantee data integrity and access control across all layers. This paper proposes a unified data-provenance-based security system that protects the entire End-Fog-Cloud environment. The system ensures data integrity through blockchain-anchored log provenance and enforces access control using DID (Decentralized Identifier) and VC/VP (Verifiable Credential/Verifiable Presentation)-based identity verification. Finally, we experimentally verify that the proposed system effectively detects and mitigates major security threats using the STRIDE threat model.

※ 이 논문 또는 저서는 2024년 대한민국 교육부와 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2024S1A5C3A01043273)
◆ First Author : Kangwon National University Department of Computer Science and Engineering, tm3693@kangwon.ac.kr
° Corresponding Author : Kangwon National University Department of Computer Science and Kangwon National University IGP. in Medical Bigdata Convergence, Department of Data Science, mjchoi@kangwon.ac.kr
* Kangwon National University IGP. of Medical Bigdata Convergence, jiwonbang@kangwon.ac.kr
** Kangwon National University Department of Sport Science, phyhee@kangwon.ac.kr
논문번호 : KNOM2025-02-011, Received November 13, 2025; Revised December 1, 2025; Accepted December 21, 2025

Ⅰ. 서 론

IoT (Internet of Things)는 센서와 통신 모듈을 내장한 디바이스를 통해 실시간으로 데이터를 수집하고 전송하는 기술이다. 이러한 IoT 기술은 의료, 스마트 팩토리 등 다양한 분야에서 활용되고 있다 [1][2]. 그러나 IoT 디바이스는 저전력, 저사양이라는 특성 때문에 대규모 데이터를 장기 저장하거나 복잡한 연산을 수행하는 데 한계가 있다 [3][4]. 이러한 제약을 보완하기 위해 IoT 환경은 클라우드 컴퓨팅(Cloud Computing)과 결합되어 발전해왔다. 클라우드 컴퓨팅은 서버, 스토리지 등의 자원을 인터넷을 통해 제공하는 기술이다. 두 기술의 결합으로 IoT 디바이스에서 생성되는 데이터를 대규모 저장 공간과 고성능 환경에서 효율적으로 저장하고 처리할 수 있도록 지원한다 [5]. 하지만 클라우드와 IoT로 구성된 환경은 몇 가지 근본적 문제를 지닌다. IoT 디바이스와 클라우드 간의 물리적 거리로 인해 네트워크 지연이 발생하고, 실시간 처리가 어려워진다. 또한 클라우드에 대한 의존성이 커진다는 문제도 존재한다 [6][7]. 이러한 문제를 해결하기 위해 그림 1과 같이 End (IoT)와 클라우드 사이에 중간 계층을 두는 Fog 컴퓨팅이 도입되었다 [8].

End - Fog - Cloud 환경에서 Fog 계층은 End 계층과 지리적으로 가까운 위치에서 데이터를 처리한다. 이에 따라 네트워크 지연을 줄이고, 연산량이 큰 작업은 클라우드가 담당해 전체 부하를 효과적으로 분산한다. 이러한 구조는 IoT의 성능 제약을 해소하지만, 데이터 무결성과 접근 제어와 같은 보안 문제를 여전히 가진다 [9]. IoT 환경에서는 의료 데이터, 개인정보 등 민감한 데이터가 다루어진다. 이러한 데이터가 제3자에게 노출되거나 공격자에 의해 변조될 경우 심각한 피해로 이어질 수 있다. 변조된 데이터가 상위 서비스나 의사 결정 과정에 반영되면 시스템 오동작이나 장애를 유발할 수 있다 [1][2]. 그러나 IoT는 자원이 제한적이기 때문에 기존의 무거운 보안 기법을 적용하기 어렵다. 또한 수많은 디바이스가 동시에 인증받아야 하므로 중앙 PKI (Public Key Infrastructure) 기반 인증 방식은 적합하지 않다 [10]. 다른 문제로는 클라우드 컴퓨팅의 신뢰 문제도 대두된다. 클라우드 컴퓨팅에 저장된 데이터는 통제권이 데이터 소유자에서 CSP (Cloud Service Provider)로 넘어가게 된다. 만약 CSP가 악의적인 목적으로 데이터를 삭제하거나, 관리의 문제로 데이터가 소멸할 경우, 데이터 소유자는 이를 인지하기 어렵다 [11]. 따라서 End - Fog - Cloud 환경에서는 데이터 무결성을 감시할 수 있는 구조가 필요하다. 동시에 저장된 데이터는 권한을 가진 주체만 접근할 수 있도록 접근 제어 체계를 갖추어야 한다.

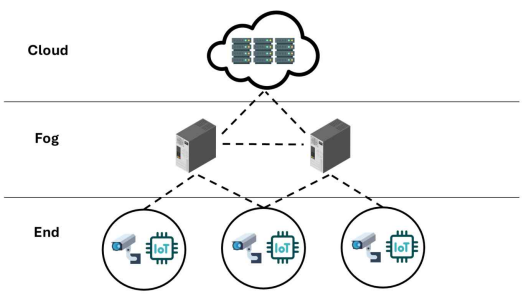

그림 1. End - Fog - Cloud 구조
Fig. 1. End - Fog - Cloud architecture

본 논문에서는 End - Fog - Cloud 환경에서 데이터 무결성과 접근 제어를 동시에 보장하는 시스템을 제안한다. 제안하는 시스템은 데이터의 생성, 변환, 전달 과정 전체를 추적하기 위해 데이터 프로비넌스(Data Provenance) 기법을 적용한다. 데이터 프로비넌스는 데이터의 출처와 처리 이력을 설명하는 메타데이터이다. 데이터의 전 생애주기를 추적함으로써 무결성 검증과 책임 추적성을 보장하는 기법이다 [12]. 본 연구에서는 로그를 기반으로 데이터 프로비넌스를 구현한다. 기록된 로그는 머클 트리(Merkle Tree) 형태로 구성되며, 루트 노드인 루트 해시를 블록체인에 저장한다. 이를 통해 클라우드에서 발생할 수 있는 데이터 위·변조를 탐지할 수 있다. 또한 DID (Decentralized Identifier), VC (Verifiable Credential)/VP (Verifiable Presentation)를 활용하여 분산 신원 인증과 토큰 기반 접근 제어를 수행함으로써 신뢰성 있는 데이터 관리 환경을 제공한다.

본 논문의 목적은 다음과 같다. 1) 블록체인, 로그 기반 데이터 프로비넌스 그리고 머클 트리를 결합하여 End - Fog - Cloud 환경에서 발생할 수 있는 데이터 위·변조 문제를 해결할 수 있는 시스템을 설계한다. 2) DID와 VC/VP 기반의 분산 신원 인증 체계를 도입하여 중앙화된 인증 서버 없이도 안전한 접근 제어가 가능한 구조를 구현한다. 3) 실험 및 분석을 통해 제안한 시스템이 데이터 무결성과 접근 제어 요구사항을 충족함을 검증한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구에서 활용한 핵심 기술인 분산 원장 기술에 대해 설명한다. 이후 기존 IoT 환경에서 데이터 무결성과 접근 제어를 보장하기 위한 연구들을 소개한다. 3장에서는 제안 시스템의 전체 구조, 동작 방식, 구성 요소에 대해 상세히 기술한다. 4장에서는 제안한 시스템의 데이터 무결성과 접근 제어 보장 능력을 분석한다. 마지막으로 5장에서는 결론과 향후 연구 방향을 제시한다.

Ⅱ. 배경지식 및 관련 연구

이 장에서는 본 논문에서 제안하는 시스템의 이해를 돕기 위해 사용된 분산 원장 기술과 배경지식을 설명한다. 이후 IoT 환경에서 보안 문제를 해결하기 위한 기존 연구를 소개한다. 마지막에는 제안하는 시스템이 이 기존 연구들과 어떤 차별점을 가지는지 설명한다.

2.1 분산 원장 기술

블록체인은 데이터를 중앙집중형 저장소가 아닌, 지리적, 국가적, 기관적으로 분산된 노드들에 동일하게 저장하는 분산 원장 (Distributed Ledger) 기술이다. 새로운 블록이 생성될 때 합의 알고리즘을 통해 정당성을 검증한다. 각 블록은 이전 블록의 해시 값을 포함하는 체인 구조로 연결된다. 만약 저장된 데이터를 수정 및 삭제하려면 모든 블록을 동시에 변경하고 참여자의 합의를 다시 획득해야 한다. 이러한 구조 덕분에 블록체인에 기록된 데이터는 불변성을 가진다. 대표적인 블록체인으로는 Ethereum과 Hyperledger Fabric 등이 있다 [13].

스마트 컨트랙트(Smart Contract)는 제3자의 개입 없이 계약 내용을 자동으로 실행하는 트랜잭션 프로토콜이다. 스마트 컨트랙트는 생성(Creation), 배포(Deployment), 실행(Execution), 완료(Completion)의 4단계 생명주기를 거쳐 동작한다. Ethereum이나 Hyperledger Fabric과 같은 다양한 플랫폼을 통해 구현된다. 스마트 컨트랙트는 블록체인의 불변성과 분산 원장을 통해 계약의 위변조를 방지하고 투명성을 보장함으로써, 기존 방식 대비 관리 비용을 절감하고 비즈니스 프로세스의 효율성을 개선하는 장점을 가진다 [14].

머클 트리는 대규모 집합의 데이터 무결성을 효율적으로 검증하기 위해 고안된 해시 기반 트리 구조이다. 모든 리프 노드에는 원본 데이터의 해시 값이 저장되며, 자식 노드 두개의 해시 값을 합친 후 다시 해시 값을 구해 상위 노드를 구성한다. 최종적으로 완성된 트리에서 루트 노드를 루트 해시라고 부른다. 루트 해시는 전체 데이터 집합의 요약 값 역할을 한다 [15].

그림 2. DID와 DID 문서 예시
Fig. 2. Example of DID, DID document

DID는 World Wide Web Consortium (W3C)에서 제안한 새로운 유형의 검증 가능한 분산 신원 인증 식별자이다. DID는 신뢰 가능한 저장 공간이 필요하다. 이러한 이유로 DID 기술은 저장된 데이터가 불변성을 가지는 블록체인과 결합되어 사용된다. 사용자가 스스로 DID를 생성하면, DID 문서가 같이 생성되며, 이는 JSON 파일로 DID 문서의 소유자의 DID, 인증 방식, 공개키 등을 저장한다. 그림 2는 DID와 DID 문서의 예시이다. “did”는 DID 스키마로 DID 체계를 따른다는 표시이다. “ethr”는 DID 메서드로 어떻게 구현되는지를 정의한다. ethr는 Ethereum을 뜻하며, web 등 여러 가지가 존재한다. 마지막 부분은 DID를 식별하는 고유의 값이다 [16].

VC는 W3C에서 표준한 데이터 모델로, 신분증, 자격증과 같이 물리적 세계의 자격 증명을 디지털 환경에서 암호화하여 표현한 것이다. VC는 기존의 중앙 집중식 신원 관리 시스템과 달리, 사용자가 자신의 신원 정보를 직접 소유하고 관리한다. VC는 다음과 같은 요소로 구성된다. Claims은 주체에 대한 이름, 나이, 소속 등 정보의 집합이다. Metadata는 자격 증명의 유형, 발급자 정보, 발급 일시, 만료 일시 등을 기록한다. 마지막 Proof는 VC의 발급자의 전자 서명이 포함된다.

VP는 자신의 신원 인증을 받아야 할 때, 하나 이상의 VC로 생성하는 데이터 포맷이다. 주요 장점으로는 사용자는 VC 중 필요한 신원 정보만 제출할 수 있다는 점이다. 신원 인증이 필요할 때, VC 그대로 제출해도 되지만, 검증자가 요구하는 특정 조건에 맞춰 데이터를 가공하여 제출할 수 있다. 이를 통해 모든 정보를 공개하지 않고, 검증에 필요한 최소한의 정보만을 선택적으로 공개할 수 있다 [17].

2.2 관련 연구

End‑Fog‑Cloud 환경에서는 IoT 환경에서 생성·전송·저장되는 데이터는 여러 계층을 거쳐 처리된다. 또한, 클라우드와 같은 외부 인프라도 사용하기 때문에 데이터 무결성, 접근 제어는 필수적인 보안 요구사항으로 간주된다. 이를 해결하기 위해 기존 연구들은 무결성 검증 기법, 블록체인 기반 데이터 프로비넌스, 분산 접근 제어 모델을 제안해 왔다. 그러나 대부분은 특정 부분에만 집중했으며, 포괄적으로 보호하는 시스템을 제안하는 연구는 드물었다. 본 장에서는 이러한 선행 연구들을 검토하고, 본 논문이 제안하는 시스템과의 차별점을 정리한다.

클라우드 기반 IoT 환경에서 데이터 무결성과 접근 제어를 보장하기 위해 다양한 연구가 진행되었다. Zang et al.은 중앙 집중형 스토리지의 단일 장애 지점, 보안 문제(무단 삭제, 변조, 오염 등)을 지적하며 Edge 환경에 적합한 탈중앙 저장 구조인 DataPond 모델을 제안하였다. 이 모델은 실제 데이터를 여러 Edge 노드에 분산 저장한다. 그 후 블록체인에는 데이터의 버전, 수정 시각, 소유자와 같은 메타데이터만 기록한다. 이러한 과정을 통해 데이터 무결성과 최신성을 검증할 수 있도록 설계되었다. 또한 Kubernetes 기반의 클라우드 네이티브 오케스트레이션을 결합해 노드 장애 대응, 동적 역할 할당, 확장성 향상 등을 지원한다 [18].

Nguyen et al.의 연구는 IoT 데이터를 공유하는 환경에서, 기존 접근 제어 방식이 상황 변화(컨텍스트)를 반영하지 못한다는 문제를 지적한다. IoT 데이터는 시간, 장소, 사용 목적, 센서 상태 등 주변 상황에 따라 중요도와 민감도가 달라진다. 하지만 기존 시스템은 이런 변화된 상황을 실시간으로 반영하지 못한다. 이에 따라 접근을 과도하게 허용하거나 불필요하게 차단되는 문제가 발생한다. 이를 해결하기 위해 Edge 환경에서 동작하는 EDSaaS (Edge-Driven IoT Data Sharing as a Service) 프레임워크를 제안했다. 이 시스템은 "지금이 허용된 시간인지", "이벤트가 발생했는지" 같은 상황 정보를 실시간으로 정책에 반영하며, 해당 조건에 맞게 접근을 자동 조정한다. 정책은 계약 형태로 작성된 후 Edge Hub에서 해석되어 즉시 적용되기 때문에, 사용자의 요청이나 상황 변화에 빠르게 대응할 수 있다 [19].

IoT 환경에서 데이터의 신뢰성을 보장하기 위해 데이터 프로비넌스 적용한 연구도 여러 연구가 진행되었다. Jaigirdar et al.의 연구에서는 IoT 환경에서는 최종 사용자가 데이터의 투명성을 확보하기 위해 전통적인 프로비넌스 기법에 보안 메타데이터를 결합한 모델을 제안하였다. Prov-DM에 보안 메타데이터를 추가하여 Prov-IoT 데이터 프로비넌스 모델을 제안하였다. Prov-DM은 W3C에서 제안한 모델로, entity, activity, agent를 기록하여 데이터 흐름을 표현한다. 데이터가 이동하는 각 단계에서 발생하는 작업과 함께 인증 실패, 미설정된 보안 모듈, 비정상 접근 시도 등 보안 관련 이벤트도 기록한다. 이를 통해 데이터가 어디서 생성되고 어디를 거쳐 전달되었는지 추적 가능하다. 더 나아가, 전달 과정 중 어떤 보안 문제가 발생했고 해당 문제가

표 1. 기존 연구와 제안 시스템 비교
Table 1. Comparison of existing research and proposal systems

Research	Data Integrity Mechanism	Access Control Mechanism	Provenance Support	Considered Layers
[17]	Blockchain-based integrity, IPFS/DHT	Blockchain + relational DB	Data Confidence (Blockchain Ledger)	Edge-Cloud (Fog layer largely not considered)
[18]	Policy violation prevention	Context-sensitive access control	Not supported	End‑Fog
[19]	Security metadata‑based integrity event detection	Not supported	Prov-DM model	End‑Cloud (Fog considerations are limited)
[20]	Zero-watermarking for hop-by-hop integrity verification	Not supported	Watermark-based lightweight provenance	End‑Multi-hop‑Gateway (Cloud not included)
Ours	Blockchain + Merkle Tree + Data Provenance	DID, VC/VP	Log-based data provenance	End‑Fog-Cloud

데이터의 신뢰성과 무결성에 어떤 영향을 미쳤는지를 한눈에 확인할 수 있다 [20].

Faraj et al.은 IoT 네트워크에서 데이터 무결성과 프로비넌스를 동시에 확보하기 위한 시스템인 ZIRCON을 제안하였다. 기존 IoT 환경에서는 멀티 홉(Multi-hop) 전송 과정에서 중간 노드가 데이터를 변조, 삭제하거나 공격자가 패킷을 삽입하더라도 이를 탐지하기 어려운 문제가 존재한다. 또한, 프로비넌스 레코드를 패킷에 추가해서 전송하는 구조는 전송할수록 패킷 크기가 선형적으로 증가한다. 이로 인해 IoT 센서에서 과부하를 일으킨다는 한계가 존재한다. ZIRCON은 이러한 문제를 해결하기 위해 패킷 내부에 프로비넌스 정보를 삽입하지 않는 zero-watermarking 방식을 적용한다. 이 기법은 데이터 생성 시 IP 주소, 타임스탬프, 시퀀스 번호, 데이터 해시를 기반으로 고유 watermark를 생성한다. 그 후 tamper-proof 네트워크 데이터베이스에 저장한다. 중간 노드는 패킷을 수신할 때마다 watermark를 계산해 원본 watermark와 비교를 진행한다. 이 과정을 통해 패킷 변조 여부를 실시간으로 판단하여 멀티홉 환경에서 위·변조 탐지가 가능하다 [21].

기존 연구들은 IoT 환경에서 데이터 무결성과 접근 제어를 보장하기 위해 다양한 방법을 제안해왔다. 일부 연구는 블록체인 기반 무결성 검증이나 컨텍스트 기반 접근 제어 등 특정 기능에 집중했다. 다른 연구들은 데이터의 신뢰성을 확보하기 위해 프로비넌스 기법을 적용하였다. 그러나 End, Fog, Cloud 중 특정 계층이나 제한된 환경만을 고려한다. 또한, 데이터 무결성, 데이터 프로비넌스, 접근 제어를 단일 시스템 안에서 통합적으로 제공하는 연구는 부족하다. 이에 본 논문에서는 End-Fog-Cloud 전체 범위에서 세 가지 기능을 모두 지원하는 시스템을 제안하며, 기존 연구와의 차이를 표 1에 정리하였다.

Ⅲ. 시스템 설계

본 장에서는 End-Fog-Cloud 환경에서 데이터 무결성과 접근 제어를 보장하기 위한 시스템 설계를 제시한다. 제안하는 시스템은 데이터 생성부터 저장, 접근, 변환에 이르는 전 과정을 로그로 기록한다. 이러한 로그를 기반으로 데이터 흐름을 검증·추적하는 로그 기반 데이터 프로비넌스 기법을 사용한다.

그림 3은 제안하는 시스템의 전체 구조를 나타낸다. End 계층은 IoT 디바이스와 게이트웨이로 구성된다. IoT 디바이스는 센서, 카메라 등에서 수집한 데이터를 게이트웨이로 전송한다. 게이트웨이는 수집 데이터를 필터링과 정규화를 진행한다. 이후 암호화를 수행한 뒤 Fog 계층으로 전달한다. 이를 통

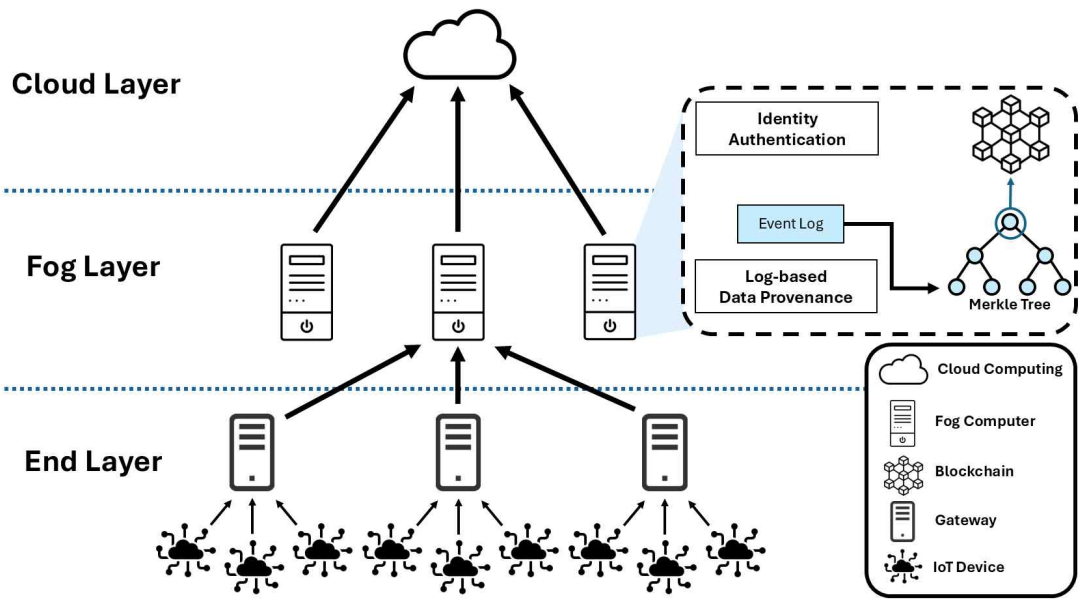

그림 3. 제안 시스템 구조
Fig. 3. Proposed system architecture

해 디바이스의 부족한 연산 능력을 게이트웨이가 보완한다. Fog 계층에는 본 시스템의 핵심 요소인 Fog 노드가 위치한다. Fog 노드는 블록체인 노드를 유지하며, 데이터 무결성을 보장하기 위한 작업을 처리한다. 또한, 이벤트가 발생하면 Fog 노드는 로그를 기록하며, 일정 주기마다 수집된 로그 데이터와 데이터를 로그 머클 트리와 데이터 머클 트리로 생성한다. 계산된 루트 해시를 스마트 컨트랙트를 이용해 블록체인에 업로드한다. DID·VC/VP 기반의 신원 인증과 토큰 기반 접근 제어를 수행한다. 더불어 클라우드 스토리지와의 데이터 송수신을 중계하는 역할도 수행한다. Cloud 계층은 클라우드 스토리지를 통해 원본 데이터를 저장하는 역할을 수행한다. 저장된 데이터는 Fog 노드에서 생성된 로그와 블록체인에 기록된 루트 해시를 이용해 무결성 검증이 가능하다.

제안하는 시스템은 End-Fog-Cloud 전 구간의 데이터 처리 과정에 신원 기반 접근 제어와 로그 기반 데이터 프로비넌스를 적용한다. 이를 통해 데이터 흐름 전반에서 발생할 수 있는 위·변조와 비인가 접근을 차단한다. End 계층은 데이터를 전처리하고 Fog 계층은 인증, 검증, 로그 생성을 담당한다. Cloud 계층은 최종 데이터를 저장한다. 또한 블록체인에 기록된 머클 루트를 활용해 저장된 데이터의 무결성을 검증할 수 있다. 이러한 구조를 통해 제안하는 시스템은 End-Fog-Cloud 환경에서 신뢰할 수 있는 데이터 관리와 접근 제어를 제공한다.

Ⅳ. 시스템 구현

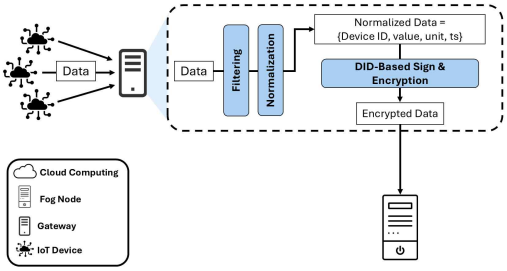

그림 4. End 계층 구현 구조
Fig. 4. End layer Implementation architecture

본 장에서는 제안하는 시스템을 어떻게 구현했는지 설명한다. 표 2는 End-Fog-Cloud 구조에서 계층별로 구현 방법을 정리했다. 이후 절에서는 해당 표를 기준으로 각 계층의 구체적인 구현 방식과 역할을 상세히 설명한다. 마지막으로는 시스템 전체가 어떻게 동작하는지 흐름을 설명한다.

4.1 End 계층 구현

End 계층은 데이터의 생성과 전처리의 역할을 가지며, 그림 4와 같이 구현한다. 구성 요소는 IoT 디바이스와 게이트웨이로 구성된다. 두 구성 요소는 DID 기반 인증을 활용해 데이터를 안전하게 처리한다. IoT 디바이스는 센서나 카메라를 통해 데이터를 수집하고 이를 게이트웨이로 전달한다. 게이트웨이는 수신된 데이터를 필터링한다. 이후 디바이스 ID, 측정 값, 단위, 타임스탬프 형태로 정규화를 진행한다. 이러한 전처리 과정을 통해

표 2. End-Fog-Cloud 계층별 구현 요약
Table 2. End-Fog-Cloud Layer Implementation Summary

Layer	Components	Primary Role	Key Functions
End	IoT Device, Gateway	Data collection, Preprocessing	Data collection, Filtering, Normalization, Encryption
Fog	Fog Node	Authentication, Access control, Data provenance	DID, VC/VP verification, Session token issuance, Log generation, Maintaining an Ethereum node, Merkle root computation & Blockchain anchoring, Cloud interaction
Cloud	ownCloud	Original data storage	WebDAV-based data put/get/remove/modify, Returning data upon Fog requests

데이터는 이기종 환경에서도 일관된 구조를 갖춘다. 정규화된 데이터는 먼저 IoT 디바이스의 DID에 대응하는 개인키로 서명되어 데이터의 출처와 무결성이 보장된다. 이후 Fog 노드의 DID 문서에 포함된 공개키로 암호화한 뒤 Fog 계층으로 전송된다. Fog 노드는 공개키 기반 서명 검증을 통해 데이터의 무결성과 출처를 확인하고, 자신의 개인키로 데이터를 복호화한다.

4.2 Fog 계층 구현

Fog 계층에서는 제안하는 시스템의 핵심인 Fog 노드가 동작된다. 본 절에서는 머클 트리를 어떻게 구현했는지를 설명한다. 이후, 로그 기반 데이터 프로비넌스의 구조에 대해 설명한다. 마지막으로 Fog 노드가 수행하는 세부 동작을 기술한다.

4.2.1 머클 트리 구현

본 논문에서 제안하는 시스템은 로그 기반 데이터 프로비넌스의 위·변조를 방지하기 위해 머클 트리를 활용한다. Fog 노드의 처리 부하와 블록체인 앵커링 비용을 고려하여, 데이터 무결성 검증 지연을 과도하게 증가시키지 않는 범위에서 30분 주기의 머클 트리 생성을 적용하였다. Fog 노드에서는 30분 주기로 로그 데이터와 수집된 데이터를 각각 로그 머클 트리와 데이터 머클 트리로 계산한다. 두 트리는 동일한 방식으로 생성된다. 각 레코드는 고정된 형식으로 직렬화된 뒤 Keccak-256 해시 알고리즘을 적용하여 리프 노드를 구성한다. 이후 인접한 두 리프를 사전순으로 정렬한다. 그 후 다시 Keccak-256 해시를 적용해 상위 노드를 생성하며, 이를 반복해 최종적으로 하나의 루트 해시만 남도록 한다. 만약 리프 노드가 홀수인 경우, 마지막 리프 노드를 복제하여 짝수 개로 맞추는 방식을 이용한다. 이렇게 얻어낸 머클 트리의 루트 해시를 스마트 컨트랙트를 이용해 Ethereum에 저장한다. 본 구현에서는 Ethereum Sepolia Testnet을 사용한다. 저장할 때, 루트 해시를 찾기 위해 Fog 노드는 Hash ID를 생성한다. Hash ID는 디바이스 ID, 수집 시작 시간, 수집 종료 시간 그리고 루트 해시의 종류로 문자열을 만든다. 루트 해시의 종류는 데이터 머클 트리인 경우, data, 로그 머클 트리인 경우, log로 기록된다. 스마트 컨트랙트를 호출해서 Hash ID를 입력하면 해당 시간에 기록된 디바이스 ID의 머클 트리를 손쉽게 얻어 올 수 있다.

4.2.2 로그 기반 데이터 프로비넌스 구현

Fog 노드는 데이터의 생성, 접근 이력, 변환 과정들을 로그로 기록한다. 로그의 세부 구조는 그림 5와 같다. event_id는 각 로그의 고유 값이다.

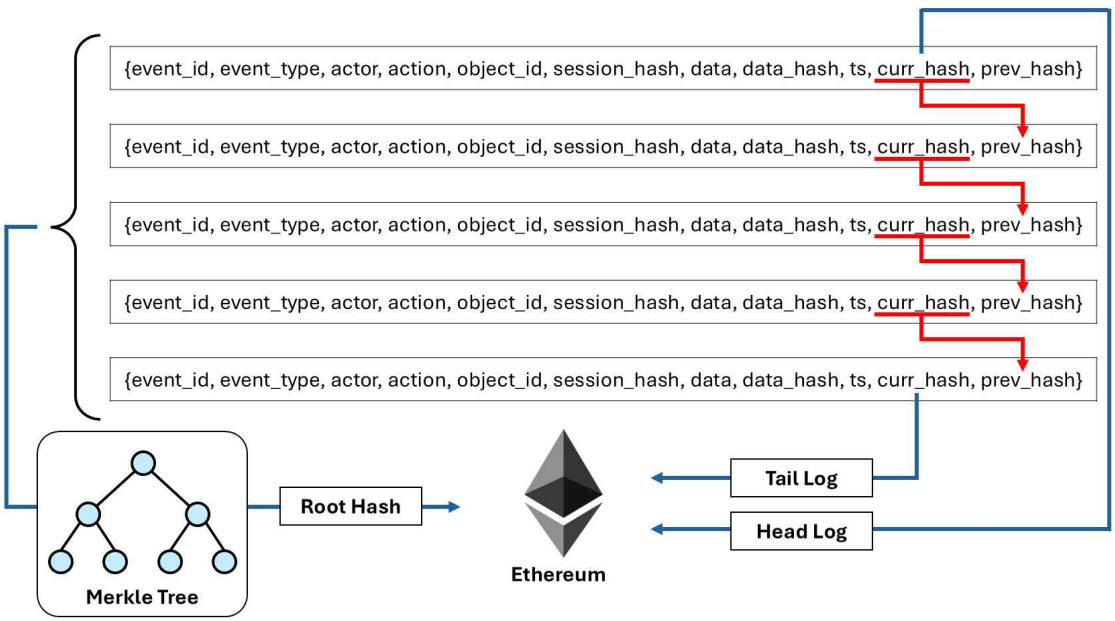

그림 5. 로그 기반 데이터 프로비넌스
Fig. 5. Log-based data provenance

event_type은 메시지의 유형을, actor는 이벤트를 발생시킨 객체를 기록한다. action은 event_type에서의 세부 동작을 명시한다. object_id는 어떤 디바이스에서 어느 시각에 생성된 데이터인지 식별하기 위한 값이다. device_1에서 2025년 11월 1일 13시 5분 30초에 생성된 데이터의 경우 object_id는 device_1:2025-11-01 13:05:30이 된다. session_hash는 접근 제어에 사용된 세션 토큰의 해시 값이다. data는 Fog 노드로 전달된 원본 페이로드이며, data_hash는 data의 무결성을 검증하기 위한 해시 값이다. ts는 로그가 기록된 시각을 기록한다. curr_hash는 해당 로그의 해시 값이며, prev_hash는 직전 로그의 해시 값을 기록한다.

기록된 로그들은 위·변조가 발생해서는 안 되므로, 본 시스템은 로그의 데이터 무결성을 보장하기 위해 다음의 작업을 수행한다. Fog 노드는 수집된 로그들을 30분 간격으로 머클 트리로 계산한다. 이후 계산된 루트 해시를 스마트 컨트랙트를 호출해 블록체인에 저장한다. 제안하는 시스템은 주기적으로 기록된 로그들을 머클 트리로 구성한다. 새롭게 계산된 루트 해시와 블록체인에 기록된 루트 해시를 비교해 로그들이 변조되지 않았는지 지속적으로 검증한다.

제안하는 방법은 주기적으로 생성되는 머클 트리를 통해 데이터 무결성을 보장할 수 있다. 그러나 머클 트리가 생성되기 전에 로그가 수정되는 경우 이를 탐지할 방법이 없다. 이를 보완하기 위해 본 논문에서는 curr_hash와 prev_hash를 연결한 체인형 구조로 구성한다. 특정 로그가 변조되면 이후 모든 로그의 해시가 연쇄적으로 불일치하게 된다. 이를 통해 변조 탐지가 가능하다. 그러나 이런 구조에서도 문제가 있다. 제일 처음 기록된 Head 로그와 머클 트리가 생성되기 전 마지막에 기록된 Tail 로그가 보호되지 않는다. 이를 해결하기 위해 본 논문에서는 Head 로그와 Tail 로그의 해시 값을 스마트 컨트랙트를 통해 블록체인에 저장한다. 결과적으로 Head와 Tail 로그가 변조되더라도 블록체인에 저장된 해시와 대조함으로써 완전한 검증이 가능해진다.

4.2.3 Fog 노드 동작 구현

Fog 노드의 전체 동작은 그림 6과 같다. 사용자 또는 게이트웨이는 요청 목적을 명시하는 msg_type과 관련 데이터를 포함하여 페이로드를 생성한다. 이후 생성된 페이로드를 TCP를 통해 Fog 노드로 전송한다. Fog 노드는 수신된 페이로드를 파싱(Parsing)하여 msg_type을 식별하고, 이에 따라 다음과 같이 분기 처리를 수행한다.

첫 번째로, msg_type이 data인 경우, Fog 노드는 페이로드 내의 device_id와 타임스탬프를 분석한다.

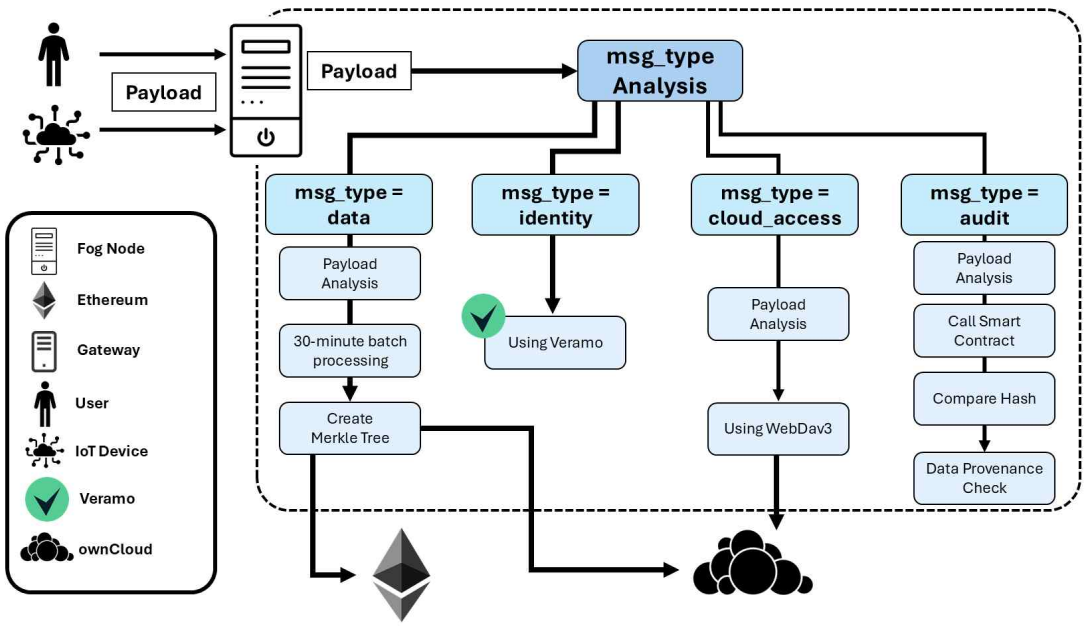

그림 6. Fog 노드 동작
Fig. 6. Fog Node Processing

이후, 데이터의 출처 및 생성 시점을 식별한다. 수집된 데이터는 30분 단위로 배치(Batch) 처리되며, Fog 노드는 해당 주기 동안 축적된 데이터에 대해 머클 트리를 구성한다. 머클 트리에서 루트 해시 얻어, 스마트 컨트랙트를 통해 블록체인에 기록한다.

두 번째로, identity 유형은 Fog 노드가 신원 인증 및 접근 제어 작업을 수행한다. Fog 노드는 페이로드 내 action 필드를 읽고 VP 검증, 세션 토큰 발급 등 인증 관련 작업을 수행한다. 본 시스템은 DID 및 VC/VP 기반 프레임워크인 Veramo [19]를 활용한다. Fog 노드는 Veramo 서버를 구동한다. 이후 HTTP API를 통해 신원 인증 및 접근 제어 관련 작업을 수행한다.

세번째로, cloud_access 유형은 클라우드에 저장된 데이터에 대한 접근을 제어한다. action 필드에는 세부 동작이 명시된다. 동작은 총 4가지로, get은 데이터 조회, put은 저장, remove는 삭제, modify는 수정을 진행한다. Fog 노드는 해당 요청을 검증한 후, Python 라이브러리인 webdav3를 이용하여 WebDAV API를 통해 ownCloud와 상호작용한다.

마지막으로, msg_type이 audit인 경우는 사용자가 CSP의 데이터 위·변조, 임의 삭제 등 부정행위를 의심하여 데이터 무결성 검증을 요청하는 상황을 처리한다. Fog 노드는 해당 페이로드를 수신하면, 클라우드에 저장된 의심 시간의 데이터를 가져온 뒤 머클 트리를 계산 후 루트 해시를 얻는다. 이후, Ethereum에 기록된 루트 해시를 비교해 불일치가 있는지 확인한다. 만약 불일치가 발생하면, Fog 노드는 감사 요청이 들어온 시간부터 데이터가 생성된 시간까지의 로그 데이터를 역순으로 추적한다. 예를 들어 감사 요청 시각이 2025년 10월 30일 12:00:00이며, 의심되는 데이터가 2025년 10월 28일 12:00부터 13:00까지의 구간이라면, Fog 노드는 감사 요청 시각의 로그 데이터부터 10월 28일 12:00까지 순차적으로 확인한다. 이 과정에서 로그에 포함된 object_id가 핵심적인 식별자로 사용된다. 클라우드에 저장된 원본 데이터들을 이용해서 cloud_object_id를 만든 후, 로그 데이터를 역방향으로 탐색하면서 해당 데이터의 가장 마지막에 기록된 로그를 찾는다. 일치하는 object_id를 발견했고, 로그에 기록된 내용이 맞으면, 해당 데이터는 위·변조가 되지 않은 데이터란 뜻이다. 하지만, cloud_object_id와 일치하는 object_id를 찾을 수 없거나, 찾았음에도 로그의 data 부분과 실제 클라우드에 저장된 데이터가 불일치가 발생한다면 이는 CSP가 데이터를 무단으로 변조했거나 삭제했다는 뜻이다. 이러한 과정을 통해 데이터 무결성 검증이 가능하다.

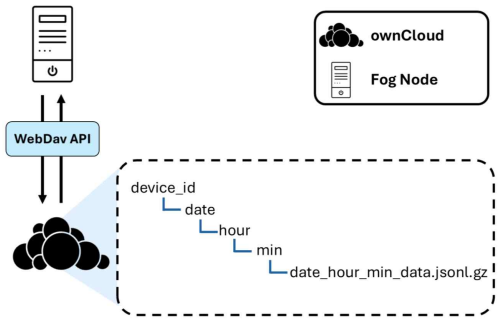

그림 7. Cloud 계층 구현
Fig. 7. Cloud layer Implementation

4.3 Cloud 계층 구현

Cloud 계층은 원본 데이터를 저장하기 위한 클라우드 컴퓨팅이 동작된다. 본 논문에서 제안하는 시스템은 ownCloud를 사용하여 그림 7과 같이 구현하였다. ownCloud는 파일 저장, 사용자 관리, 동기화 기능을 제공하는 오픈소스 클라우드 스토리지 플랫폼이다. ownCloud는 WebDAV 기반의 표준화된 인터페이스를 제공한다. 이를 통해 Fog 노드는 저장소에 대한 조회, 저장, 삭제, 수정 요청을 안전하게 수행할 수 있다. ownCloud는 Docker Compose를 이용해 구성하였다. ownCloud 컨테이너는 MariaDB, Redis, Apache 등과 함께 작동한다. 이를 통해 안정적인 스토리지 운영과 데이터 관리가 가능하다. Fog 노드는 Python의 webdav3 라이브러리를 통해 ownCloud의 WebDAV API와 직접 상호작용하며, 데이터 입출력을 처리한다. 저장되는 데이터는 디바이스별·날짜별·시간대별로 체계적으로 관리된다. 디렉터리 구조는 device_id/날짜/시간/분/파일 형태로 구성된다. 그에 대한 예시로, device_1에서 2025년 11월 1일, 13시 30분에 수집된 경우, device_1/2025-11-01/13/30/2025-11-01_13_data.jsonl.gz와 같이 저장된다. 이러한 구조를 통해 시간 단위로 데이터를 관리한다.

4.4 시스템 동작

본 절에서는 제안하는 시스템의 전체적인 동작을 설명한다. 동작은 총 4가지로, 등록, 저장, 사용

그리고 감사 과정으로 구분된다. 각 동작은 End-Fog-Cloud 환경에서 서로 다른 구성 요소들이 어떤 역할을 수행하며 데이터를 처리하는지를 단계별로 보여준다.

첫 번째로 등록 단계는 그림 8과 같은 흐름을 가진다. 등록 단계는 IoT 디바이스나 사용자가 시스템에 참여하기 위해 Fog 노드로부터 접근 권한을 획득하는 과정이다. 등록 주체는 총 세 가지로, 데이터 생성자인 IoT 디바이스, IoT 디바이스를 소유 중인 IoT 디바이스 소유자 그리고 수집된 데이터를 사용하는 데이터 사용자로 분류된다. 등록 절차는 주체의 유형에 따라 다음과 같이 수행된다. 먼저 IoT 디바이스 소유자와 데이터 사용자는 정부, 기업, 교육 기관 등 신뢰할 수 있는 기관에 자신의 DID를 제출하여 VC를 발급받는다. 이후, 획득한 VC들을 조합하여 자신이 정당한 권한을 가진 사용자임을 증명하는 VP를 생성한다. 반면, IoT 디바이스의 경우, 제조 또는 초기 설정 단계에서 DID와 VP가 ROM에 저장된 상태로 운용된다. 각 주체가 생성된 VP를 포함해 msg_type이 identity인 페이로드를 Fog에 제출하면, Fog 노드는 VP의 유효성을 검증한다. 검증이 성공하면 해당 주체의 역할에 부합하는 세션 토큰을 생성하여 발급한다. 이후 발생하는 모든 페이로드에서 주체는 발급받은 세션 토큰을 포함하여 전송함으로써, 반복적인 DID 검증을 할 필요 없이 자신이 사전에 승인된 사용자임으로 효율적으로 인증한다. 결과적으로 본 시스템은 VP 기반의 신원 인증과 세션 토큰 기반의 접근 제어를 결합하여 IoT 환경에서의 데이터 기밀성을 보장한다.

두 번째는 저장 단계로 IoT 디바이스에서 수집한 데이터가 게이트웨이와 Fog 노드를 거쳐 클라우드에 안전하게 저장되는 과정이다. 흐름은 그림 9와 같다. 먼저 IoT 디바이스는 센서, 카메라 등 물리적 모듈에서 수집한 데이터를 게이트웨이로 전송한다. 게이트웨이는 데이터들을 필터링과 정규화를 거쳐 전처리를 수행한다. 이후 암호화를 수행한 후, IoT 디바이스가 사전에 발급한 토큰을 포함하여 Fog 노드로 전송한다. Fog 노드는 수신된 페이로드의 msg_type이 data인지를 확인하고, 포함된 세션 토큰의 유효성을 검증한다. 검증이 완료되면 전달 받은 데이터를 임시 저장하며, 이러한 내용을 로그로 기록하여 저장한다. Fog 노드는 30분 주기로 수집된 데이터에 대해 머클 트리를 구성하여 루트 해시를 Ethereum에 업로드한다. 이를 통해 데이터 무결성을 확보하고, 암호화된 원본 데이터는 클라우드 스토리지에 저장한다.

세 번째는 사용 단계로, IoT 디바이스 소유자, 데이터 사용자 그리고 다른 IoT 디바이스가 수집된 데이터에 접근하는 과정을 설명한다. 그림 10은 사용 단계의 흐름을 보여준다. 데이터 접근을 희망하는 주체는 msg_type을 cloud_access로 설정하고, 수행하고자 하는 동작, get, put, remove, modify를 action에 명시한다. 이후 접근하고자 하는 데이터의 정보, 디바이스 ID, 수집된 날짜와 시간 등을 명시한 페이로드를 생성하여 세션 토큰과 함께 Fog 노드에 전송한다. 요청을 수신한 Fog 노드는 페이로드와 함께 온 세션 토큰을 검증한다. 요청자의 신원을 확인한 후 역할 기반 접근 제어 정책에 따라 권한을 검증한다. 예를 들어 IoT 디바이스 소유자는 데이터 조회(Get)뿐만 아니라 저장(Put), 삭제(Remove), 수정(Modify)에 대한 모든 관리 권한이 부여된다. 반면, 데이터 사용자는 오직 데이터 조회(Get) 권한만이 허용된다. Fog

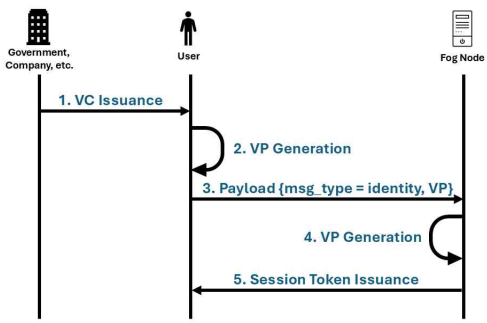

그림 8. 등록 과정 흐름
Fig 8. Registration Flow

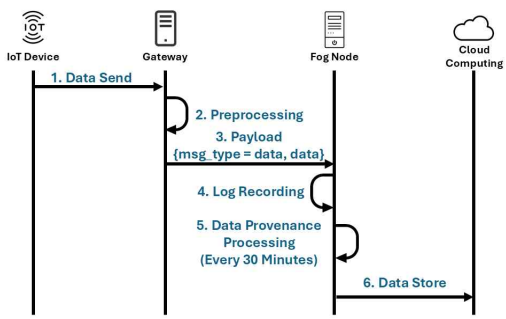

그림 9. 저장 단계 흐름
Fig. 9. Save Flow

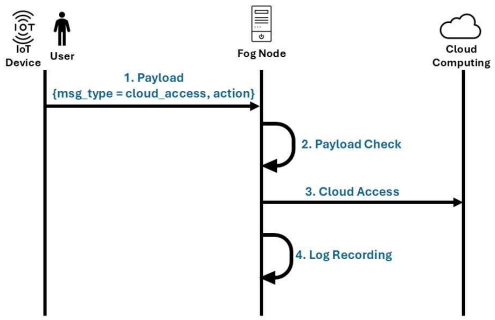

그림 10. 사용 단계 흐름
Fig. 10. Usage Flow

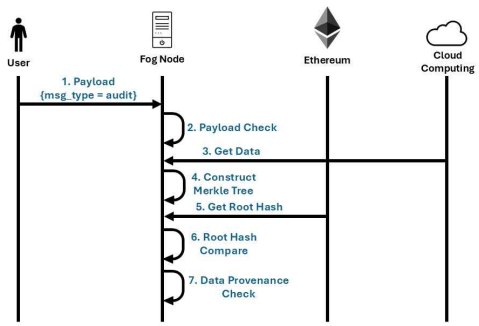

그림 11. 감사 단계 흐름
Fig. 11. Audit Processing

노드는 이러한 권한 검증이 완료된 후 클라우드와 상호작용하여 요청된 작업을 수행하고 그 결과를 반환한다. 마지막으로 이러한 동작에 대한 로그를 생성해서 저장한다.

마지막으로 감사 단계는 IoT 디바이스 소유자가 자신의 디바이스에서 수집된 데이터가 변조되지 않았는지를 확인하는 과정으로, 그림 11과 같은 흐름으로 수행된다. 사용자는 발급받은 세션 토큰과 함께 msg_type을 audit으로 설정하여 의심되는 시간 구간을 포함한 페이로드를 Fog 노드로 전송한다. Fog 노드는 세션 토큰 검증을 진행한 후, 페이로드에 명시된 시간대의 원본 데이터를 클라우드에서 불러온다. 이후 동일 구간의 데이터를 기반으로 머클 트리를 재구성하여 루트 해시를 계산한다. Fog 노드는 이 값을 Ethereum에 기록된 해당 시간대의 루트 해시와 비교한다. 두 값이 일치하는 경우 데이터가 변조되지 않았음을 확인할 수 있다. 반대로 해시 값이 다를 경우, 생성된 데이터 프로비넌스를 조회한다. Fog 노드에 저장된 해당 데이터의 마지막 로그를 찾아서 어떻게 값이 어떻게 변경되었는지 확인한다.

Ⅴ. 실험 및 검증

본 장에서는 제안하는 시스템이 데이터 무결성과 접근 제어를 보장하는지 분석한다. 보안 분석을 실제 운영을 가정한 환경을 기반으로 진행했다. IoT 디바이스는 Raspberry Pi 3를 사용하였다. 게이트웨이는 Desktop 환경으로 구현했으며, Fog 노드는 다른 Desktop 환경에서 구현하였다. 클라우드 환경은 가상 환경을 이용하였다. 본 논문에서 제안하는 시스템의 보안성을 평가하기

표 3. STRIDE 위협 모델별 제안 시스템 대응
Table 3. Countermeasures of the Proposed System based on the STRIDE Threat Model

STRIDE Category	Threat Definition	Proposed Countermeasures
Spoofing	An attacker attempts unauthorized access using a forged VP.	VP signature verification
Tampering	(1) A CSP arbitrarily modifies or deletes cloud data. (2) An attacker modifies or deletes log data recorded on the Fog node.	(1) Blockchain anchoring and auditing (2) Log chain verification
Repudiation	A legitimate user with modification privileges denies having performed an action.	Data provenance backtracking
Information Disclosure	An attacker attempts to steal data by sniffing network packets in the End–Fog segment.	Public key-based encrypted transmission
Denial of Service	An attacker attempts to flood the system with requests containing invalid signatures.	Preemptive blocking mechanism
Elevation of Privilege	A user with only read permissions attempts to modify or delete data.	Session token-based access control

위해 STRIDE 위협 모델 [22]을 적용한다. STRIDE 위협 모델은 6가지 항목을 기반으로 시스템의 보안 위협을 식별하고 분석하는 방법론이다. 항목은 총 6가지로, 스푸핑(Spoofing), 변조(Tampering), 부인(Repudiation), 정보 노출(Information Disclosure), 서비스 거부(Denial of Service), 그리고 권한 상승(Elevation of Privilege)으로 구성된다. 본 모델을 통해 시스템 설계 상의 취약점과 위험 요소를 식별할 수 있다. 본 논문에서 제안하는 시스템은 STRIDE 위협 모델의 6가지 항목을 기반으로 총 6가지 시나리오를 설정한다. 표 3은 각 항목에 대한 공격 시나리오와 이에 대한 대응을 정리하였다.

4.1 스푸핑 시나리오

첫 번째 시나리오는 스푸핑 공격으로, 공격자가 위조된 VP를 이용해 Fog 노드에 비인가 접근을 시도하는 상황을 가정한다. 본 시스템에서는 VP는 사용자의 DID와 VC를 포함하며, 신원 인증을 수행하는 핵심 요소로 사용된다. 공격자는 과거에 발급된 정당한 VP를 조작하거나, 다른 사용자의 VC를 변형하여 새로운 VP를 구성한 뒤 이를 이용해 접근 권한을 얻으려고 시도한다. 실험에서는 기존에 발급된 정상 VP를 일부 변조한 뒤 Fog 노드에 제출하여 인증 절차를 우회하려는 공격을 시도하였다. Fog 노드는 검증 과정에서 위조된 VP의 서명 불일치를 탐지하였으며, 그림 12와 같이 인증 실패 메시지를 반환하였다. 이를 통해 제안하는 시스템이 위조된 VP 기반의 신원 위조 및 비인가 접근을 효과적으로 차단함을 확인할 수 있다.

4.2 변조 시나리오

두 번째 시나리오는 변조 공격으로, 시스템 내 데이터의 무결성을 훼손하는 위협을 다룬다. 본 논문에서는 변조 공격의 대상을 크게 두 가지로 분류하여 시나리오를 구성하였다. 첫째는 클라우드 계층에서 CSP에 의해 원본 데이터가 조작되는 경우이다. 둘째는 공격자가 Fog 노드 내부 시스템에 침투하여, 자신의 악의적인 행위를 숨기기 위해 로그 데이터를 수정하거나 삭제하는 상황이다.

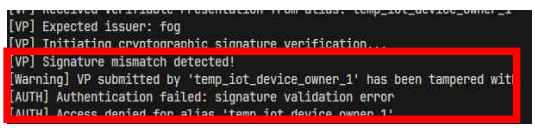

그림 12. VP 검증 결과
Fig. 12. VP Verification Results

4.2.1 CSP의 악의적인 행위

시나리오 2-1은 악의적인 CSP가 Fog 노드의 통제를 벗어나 클라우드에 저장된 데이터를 임의로 삭제하거나 값을 변경하는 상황을 가정한다. 실험은 CSP의 악의적 행위를 묘사하기 위해 Python 스크립트를 사용하여 11월 11일 10:00부터 14:00 사이에 생성된 데이터 중 무작위로 10개를 선정한다. 그 중 5개는 삭제, 5개는 값을 변조한다. 이후 해당 시간 구간에 대한 Audit을 수행한다. 그 결과, 그림 13과 같이 시스템은 삭제된 데이터와 변조된 데이터의 불일치를 모두 탐지하고 경고를 발생시켰다. 따라서 본 논문에서 제시하는 시스템은 CSP의 악의적인 행동에 대해 데이터 무결성을 검증하는 메커니즘을 제공한다.

4.2.2 로그 데이터 위·변조

시나리오 2-2는 공격자가 Fog 노드 내부 시스템에 침투하여, 자신의 악의적인 행위를 숨기기 위해 로그 데이터를 수정하거나 삭제하는 상황이다. 로그 데이터는 데이터 프로비넌스의 핵심이자 변조 탐지의 기준점이므로, 무결성이 훼손되어서는 안 된다. 공격자가 이미 블록체인에 앵커링 된 과거 로그를 수정하거나, 아직 머클 트리가 생성되기 전의 최신 로그를 조작하여 흔적을 지우려는 시도를 할 수 있다. 실험은 두 유형의 변조 시나리오를 직접 구현하였다. 첫 번째 유형은 Ethereum에 루트 해시가 기록된 로그의 내용을 수정한 후, 사용자가 audit 요청을 진행한 상황이다. 두 번째 유형은 머클 트리가 계산되기 이전에 로그를 조작하여, 해당 로그의 curr_hash와 다음 로그의 prev_hash가 불일치가 발생한 상황이다. 두 유형을 재현하기 위해, Python 스크립트로 임의의 로그 데이터를 수정하였다. 그 결과 그림 14와 같이 두 상황에서의 로그 데이터의

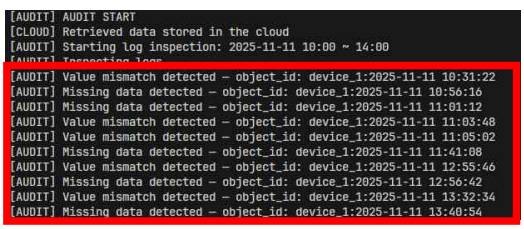

그림 13. 클라우드 데이터 감사
Fig. 13. Cloud Data Audit Results

```
[AUDIT] Experiment started: log tampering scenario simulation
[AUDIT] Detecting tampering in previously finalized logs...
[AUDIT] Loading Merkle root from blockchain...
[AUDIT] Recalculating Merkle root from cloud logs...
[AUDIT] Hash mismatch detected
Warning] Previously committed log has been modified
[AUDIT] Detecting tampering before Merkle tree generation...
[AUDIT] Verifying log chain hash linkage (prev_hash)...
[AUDIT] Broken hash chain detected - prev_hash mismatch
WARNING] Log modified before Merkle tree creation
[AUDIT] Both tampering scenarios were correctly identified
[AUDIT] System integrity verification completed
```

그림 14. 로그 위·변조 탐지 결과
Fig. 14. Log Tampering Detection Results

위·변조를 탐지했음을 보여준다. 주기적인 감사를 통해, 이전의 로그가 변조됐음을 탐지하고 경고하고 있다. 또한, 머클 트리로 계산되기 전에 로그가 수정됐을 때, 해시 값이 달라져 시스템이 위·변조를 탐지한다.

4.3 부인 시나리오

세 번째 시나리오는 부인으로, 데이터의 수정 권한을 가진 사용자가 데이터를 수정했지만, 이를 부인하는 상황을 가정한다. 정당한 권한을 가진 IoT_Device_Owner_2가 특정 데이터를 Fog 노드를 통해 수정한 후, 해당 행위를 수행하지 않았다고 주장하는 상황을 가정한다. 감사 모듈은 데이터 프로비넌스 로그를 역추적하여, 수정 이벤트가 발생한 시점의 로그를 추출하였다. 이후 로그 내에 기록된 이벤트 발생자를 확인하여 IoT_Device_Owner_2 임을 식별하였다. 그림 15는 감사 결과 화면을 나타내며, 로그에 기록된 내역을 통해 부인 방지를 보장한다.

4.4 정보 노출 시나리오

네 번째 시나리오는 정보 노출 시나리오로, 공격자가 End-Fog 구간의 네트워크 패킷을 스니핑하여 민감한 데이터를 탈취하려는 상황을 가정한다.

```
[AUDIT] Audit Request Received for Object ID: device_1:2025-12-10 19:30:45
[AUDIT] Backtracking Provenance Logs for device_1:2025-12-10 19:30:45...
[INFO] Target Event Found! Event Type: MODIFY
[INFO] Target Event Actor: IoT_Device_Owner_2
```

그림 15. 부인 방지를 위한 데이터 프로비넌스 감사 결과
Fig. 15. Audit Result of Data Provenance for Non-Repudiation

```
0000  08 bf b8 13 1a 0a b8 27  eb 9a 75 d3 08 00 45 10   ·····'··u···E·
0010  00 78 a2 c7 40 00 40 06  16 3f c0 a8 00 16 c0 a8   ·x··@·@· ·?····
0020  00 03 00 16 d7 e1 94 84  ba 7d d1 2b 53 bd 50 18   ····· ··}·+S·P·
0030  03 8f 03 6c 00 00 e0 99  a2 a8 9b 97 ea b4 e4 c3   ···l····· ·····
0040  ed 48 99 c2 4b 46 35 6b  cb 09 b0 16 22 28 73 04   ·H··KF5k ····"(s·
0050  ac 53 b0 29 81 75 f3 f7  52 69 c0 f5 0f 6a 42 9e   ·S·)·u·· Ri···jB·
0060  c8 ee 29 74 2c 0a d9 cb  ff de 6b 90 df f0 d5 53   ··)t,··· ··k···S
0070  55 d0 3b 7b 35 55 19 c3  19 b0 03 79 ef 03 93 b9   U·;{5U·· ···y····
0080  27 6f 5e 98 7e 46                                  'o^·~F
```

그림 16. Wireshark로 캡처한 암호화된 패킷 페이로드
Fig. 16. Encrypted Packet Payload Captured by Wireshark

본 시스템은 전송 구간의 기밀성을 보장하기 위해 게이트웨이가 Fog 노드의 DID 문서에서 획득한 공개키를 사용하여 데이터 페이로드를 암호화한 후 전송하는 방식을 적용한다. 실험에서 네트워크 패킷 분석 도구인 Wireshark [23]를 이용한다. Wireshark를 통해 게이트웨이와 Fog 노드 간의 통신을 가로챈 결과, 그림 16과 같이 패킷 데이터가 암호화된 상태로 전송되는 것을 확인하였다. 이를 통해 제안 시스템은 네트워크 도청 시도에 대해서도 데이터의 기밀성을 보장함을 입증하였다.

4.5 서비스 거부 시나리오

다섯 번째 시나리오는 서비스 거부 시나리오로, 공격자가 대량의 무의미한 요청을 Fog 노드에 전송하여 시스템 자원을 고갈시키고 가용성을 저해하는 상황을 가정한다. 본 시스템은 데이터 처리 로직 수행 전 세션 토큰 및 VP 서명 유효성을 검증한다. 이러한 메커니즘으로 유효하지 않은 패킷을 네트워크 수신 단계 직후 즉시 폐기한다. 실험에서는 Python 스크립트를 통해 초당 1,000건 이상의 유효하지 않은 서명이 포함된 요청을 전송하는 플러딩(Flooding) 공격을 수행하였다. 공격이 수행되면서 동시에 Fog 노드의 CPU 및 메모리 사용량을 측정하였다. 그림 17은 Fog 노드가 인증 실패를 식별하고 악성 요청을 즉각적으로 거부하는 과정을 모니터링한 결과이다. 그림 18과 그림 19는 플러딩 공격당하고 있을 때 측정한 CPU와 메모리 사용량 변화를 나타낸다. 두 그래프에서 붉은색 수직선은 공격이 시작된 시점을, 푸른색 수직선은 공격이 중단된 시점을 의미한다. CPU 사용량은 플러딩 공격을 당하는 중에도 13% 이하로 안정적으로 유지되었다. 또한 메모리 사용량은 공격을 시도한 기간 동안 급격한 증가 없이 48%를 유지하였다. 이는 대량의 악성 패킷이 시스템의 처리 대기열(Queue)이나 메모리에 적재되기 이전에 인증 검증 단계에서 즉시 폐기되었음을 나타낸다. 결과적으로 제안 시스템은 자원 고

```
[+] Access Attempt
[WARN] Session Token Verification Failed
[-] Access Denied
[+] Access Attempt
[WARN] Session Token Verification Failed
[-] Access Exit
[-] Access Denied
[WARN] Session Token Verification Failed
[-] Access Denied
```

그림 17. 서비스 거부 공격 시 모니터링 화면
Fig. 17. Monitoring Screen During a DoS Attack

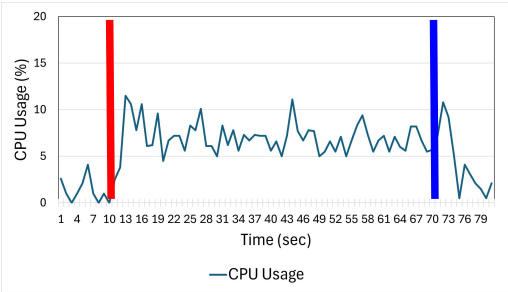

그림 18. 서비스 거부 공격 시 CPU 사용량
Fig. 18. CPU Usage During a DoS Attack

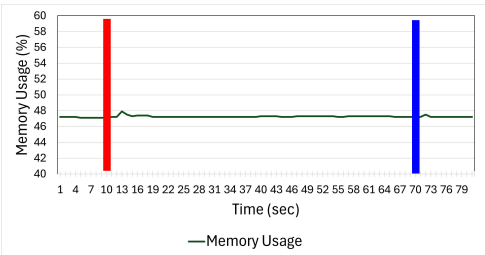

그림 19. 서비스 거부 공격 시 메모리 사용량
Fig. 19. Memory Usage During a DoS Attack

갈을 유도하는 서비스 거부 공격 상황에서도 시스템의 리소스를 효율적으로 보호하여 가용성을 안정적으로 확보함을 확인하였다.

4.6 권한 상승 시나리오

마지막 여섯 번째 시나리오는 권한 상승으로, 조회 권한만 가진 일반 데이터 사용자가 관리자 권한이 필요한 데이터 삭제나 수정을 시도하는 상황을 가정한다. 본 시스템은 이러한 비인가 행위를 방지하기 위해, 사용자 인증 시 발급된 세션 토큰에 기반한 접근 제어를 수행한다. 실험 결과, 그림 20과 같이 일반 사용자의 토큰을 포함한 데이터 삭제 요청이 토큰에 명시된 권한 범위를 초과함에 따라 시스템 정책에 의해 즉시 차단됨을 확인하였다.

```
[+] Access Attempt
[WARN] Permission Mismatch
[-] Access Exit
```

그림 20. 권한 상승 시도에 대한 접근 차단
Fig. 20. Results of Blocking Privilege Escalation Attempts

Ⅵ. 결론 및 향후 연구

본 논문에서는 End-Fog-Cloud 환경에서 발생하는 데이터 무결성 훼손과 비인가 접근 문제를 해결하기 위한 보안 시스템을 제안한다. 제안 시스템은 블록체인, 로그 기반 데이터 프로비넌스 그리고 DID·VC/VP 기반 인증 및 토큰 기반 접근 제어를 결합한 구조이다. IoT 디바이스, IoT 디바이스 소유자, 데이터 사용자를 대상으로 DID 기반 등록과 VC/VP 검증을 수행함으로써 신원 위조를 방지한다. 또한 Fog 노드는 수집된 데이터와 로그를 30분 주기로 머클 트리로 구성하고, 루트 해시를 블록체인에 저장하여 데이터와 로그의 무결성을 보장한다. 데이터 접근 과정에서 세션 토큰 기반 검증을 적용해 안전한 접근 통제를 제공한다.

제안하는 시스템의 보안성을 검증하기 위해 STRIDE 위협 모델을 기반으로 평가 시나리오를 구성하였다. 시나리오는 총 6개로, 스푸핑, 변조, 부인, 정보 유출, 서비스 거부, 권한 상승 상황으로 구성하였다. 실험 결과, 6가지의 보안 위협을 효과적으로 탐지하고 방어함으로써 시스템의 안전성과 데이터 무결성을 보장함을 확인하였다.

향후 연구에서는 로그를 역순으로 탐색하는 현재의 감사 구조가 데이터 축적에 따라 검증 시간이 증가하는 한계를 가진다는 점을 고려하여, 로그 인덱싱 기법 또는 구간 단위 요약 구조를 도입함으로써 감사 효율을 개선할 계획이다. 또한 머클 트리 생성 과정과 DID 기반 인증 절차에서 발생하는 지연 시간을 기존 시스템과 비교·분석할 예정이다. 이를 통해 데이터 무결성과 접근 제어를 강화함으로써 발생하는 오버헤드가 어느 수준인지 정량적으로 평가하고, 제안 시스템이 기존 시스템 대비 갖는 효율성을 객관적으로 제시하고자 한다.

References

[1] C. Li, J. Wang, S. Wang, and Y. Zhang, "A review of IoT applications in healthcare," *Neurocomputing 2024*, Vol.565, pp. 1-12, Jan. 2024.

[2] X. Mu, M. F. Antwi-Afari, "The applications of Internet of Things (IoT) in industrial management: a science mapping review," *International Journal of Production Research*, Vol. 62, No. 5, pp. 1928-1952, Mar. 2024.

[3] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J, Yu, and W. Lv, "Edge computing security: State of the art and challenges," *Proceedings of the IEEE*,

Vol. 107, Issue. 8, pp. 1608 – 1631, Jun. 2019.

[4] N. Sharma, P. Dhiman, "A survey on IoT Security: Challenges and their solutions using machine learning and blockchain technology," *Cluster Computing 2025*, Vol. 28, No. 313, pp. 1 – 40, Apr. 2025.

[5] A. AI-Dulaimy, M. Jansen, B. Johansson, A. Trivedi, A. Losup, M. Ashjaei, and A. V. Papadopoulos, "The computing continuum: From IoT to the cloud," *Internet of Things*, Vol. 27, pp. 1 – 33, Jun. 2024.

[6] M. Almutairi, T. F. Sheldon, "IoT – Cloud Integration Security: A Survey of Challenges, Solutions, and Directions," Electronics 2025, Vol. 14, No. 7, pp. 1 – 28, Mar. 2025.

[7] L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, and S. K. Das, "Edge-computing-driven Internet of Things: A Survey," ACM Computing Surveys, Vol. 55, Issue 8, No. 174, pp. 1 – 41, Dec. 2022.

[8] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in Proc. *Mobidata '15: Proceedings of the 2015 Workshop on Mobile Big Data,* pp. 37 – 42, New York, USA, June 2015.

[9] S. N. Srirama, "A decade of research in fog computing: Relevance, challenges, and future directions," *Practice and Experience,* Vol. 54, Issue 1, pp. 3 – 23, Jan. 2024.

[10] P. Williams, I. K. Dutta, H. Daoud, and M. Bayoumi, "A survey on security in internet of things with a focus on the impact of emerging technologies," *Internet of Things,* Vol. 19, pp. 1 – 24, Jul. 2022.

[11] P. Goswami, N. Faujdar, S. Debnath, A. K. Khan, and G. Singh, "Investigation on storage level data integrity strategies in cloud computing classification, security obstructions, challenges and vulnerability," *Journal of Cloud Computing*, Vol. 13, No. 45, pp. 1 – 23, Feb. 2024.

[12] Y. L. Simmhan, B. Plale and D. Gannon, "A survey of data provenance in e-Science," *ACM Sigmod Record*, Vol. 32, No. 3, pp. 31 – 36, Sep. 2005.

[13] M. Di Pierro, "What is the blockchain," *Computer in Science & Engineering*, Vol. 19, No. 5, pp. 92 – 95, Sep. 2017.

[14] Z. Zheng, S. Xie, H. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems,* Vol. 105, pp. 475 – 491, Apr. 2020.

[15] R. C. Merkle, "A digital signature based on a conventional encryption function," in Proc. *Advances in Cryptology—CRYPTO '87*, California, USA, pp. 369 – 378. Aug. 1987.

[16] W3C DID Working Group, Decentralized Identifiers (DIDs) v1.0(2022), Retrieved Nov. 18, 2025, from https://www.w3.org/TR/did-1.0/.

[17] W3C, Verifiable Credentials Data Model v2.0(2024), Retrieved Nov. 10, 2025, from https://www.w3.org/TR/vc-data-model-2.0/.

[18] H. Zang, H. Kim, and J. Kim, "Blockchain-based decentralized storage design for data confidence over cloud-native edge infrastructure," *IEEE Access*, Vol. 12, pp. 50083 – 50099, Apr. 2024.

[19] P. Nguyen, H. H. Nguyen, P. Phung, H. L. Truong, and T. Cheung, "Advanced context-sensitive access management for edge-driven IoT data sharing as a service," *ACM Transactions on Internet Technology*, Vol. 25, Issue 2, No. 9, pp. 1 – 31, Apr. 2025.

[20] F. T. Jaigirdar, B. Tan, C. Rudolph, and C. Bain, "Security-aware provenance for transparency in IoT data propagation," *IEEE Access*, Vol. 11, pp. 55677 – 55691, May. 2023.

[21] O. Faraj, D. Megías, and J. Garcia-Alfaro, "ZIRCON: Zero-watermarking-based approach for data integrity and secure provenance in IoT networks," *Journal of Information Security and Applications*, Vol. 85, pp. 1 – 25, Sep. 2024.

[22] M. Howard and S. Lipner, The Security Development Lifecycle, Microsoft Press, 2006.
[23] Wireshark Foundation, Wireshark User's Guide (2024), Retrieved Dec., 9, 2025, from https://www.wireshark.org/docs/.

배 태 모 (Taemo Bae)

2020년 3월~현재 : 강원대학교 컴퓨터공학과 학사

<관심분야> 네트워크 관리, 블록체인, 정보보안

방 지 원 (Jiwon Bang)

2018년 2월 : 강원대학교 컴퓨터과학과 졸업
2020년 2월 : 강원대학교 컴퓨터과학과 석사
2020년 3월~현재 : 강원대학교 빅데이터메디컬융합학과 박사

<관심분야> 네트워크 관리, 블록체인, 정보보안

황 향 희 (Hyang-Hee Hwang)

1994년 2월 : 강원대학교 체육학사
1997년 2월 : 강원대학교 교육학석사
2003년 2월 : 한국체육대학교 이학박사
2005년~현재 : 강원대학교 스포츠과학과 교수

<관심분야> 스포츠여가사회학, 스포츠여가복지, 노인체육

최 미 정 (Mi-Jung Choi)

1998년 2월 : 이화여자대학교 공학사
2000년 2월 : 포항공과대학교 공학석사
2004년 2월 : 포항공과대학교 공학박사
2004년~2005년: 프랑스 INRIA 연구소 박사 후 연구원
2005년~2006년 : 캐나다 워털루대학 박사 후 연구
2006년~2008년 : 포항공과대학 컴퓨터공학과 연구조교수
2008년~현재 : 강원대학교 컴퓨터학부 컴퓨터공학전공 교수

<관심분야> 네트워크 관리, 블록체인, 정보보안