논문 2025-4-4 http://dx.doi.org/10.29056/jsav.2025.12.04

탈중앙화 환경에서 Solid 애플리케이션 기반 인증 처리 시간 최적화 방안 연구

장성일*, 홍두표*, 조용준*, 신동명*†

A Study on Optimizing Authentication Processing Time for Solid Applications in Decentralized Environments

Sung-Il Jang*, Du-Pyo Hong*, YongJoon Joe*, Dong-Myung Shin*†

요 약

Solid 프로젝트는 데이터 주권 회복을 목표로 탈중앙화된 웹 생태계를 지향하지만, 데이터 저장소와 ID 공급자가 분리된 구조적 특성으로 인해 인증 과정에서 필연적인 네트워크 지연이 발생한다. 특히 Solid-OIDC 프로토콜의 DPoP 방식은 높은 보안성을 제공하는 반면, 복잡한 핸드셰이크와 서명 검증 과정으로 인해 애플리케이션의 초기 로딩 속도를 저하시키는 주요 원인이 되고 있다. 본 논문에서는 이러한 성능 한계를 극복하기 위해 Solid 사양을 준수하면서도 인증 처리 속도를 획기적으로 개선할 수 있는 서버 기반의 인증 최적화 아키텍처를 제안한다. 제안하는 시스템은 백엔드 서버 내에 안전 영역(Trusted Zone)을 확보하고, 캐싱 기법을 도입하여 액세스 토큰을 재사용함으로써 반복적인 서명 검증 절차를 최소화한다. Community Solid Server 환경에서의 성능 비교 실험 결과, 제안 방식은 표준 DPoP 방식 대비 약 89%, 일반 Bearer 토큰 방식 대비 약 61%의 인증 처리 시간을 단축하였다. 이는 보안성을 유지하면서도 Solid 애플리케이션의 상용화 가능성을 높이는 실질적인 해결책이 될 수 있음을 시사한다.

Abstract

The Solid project aims to restore data sovereignty through a decentralized ecosystem. However, the architectural separation between data stores and identity providers induces network latency during authentication. Specifically, the DPoP mechanism, while secure, degrades application performance due to complex handshakes and verification processes. To address this, this paper proposes a server-based authentication architecture that improves processing speed while adhering to Solid specifications. The proposed system establishes a Trusted Zone within the backend and employs caching to reuse access tokens, minimizing repetitive verifications. Experimental results in a Community Solid Server environment demonstrate that the method reduces authentication time by 89% compared to standard DPoP and 61% compared to standard Bearer tokens. These findings suggest the proposed architecture offers a viable solution for Solid applications without compromising security.

한글키워드 : 솔리드, 탈중앙화 웹, 인증, 액세스 토큰, 성능 최적화

keywords : Solid, Decentralized Web, Authentication, Aceess Token, Performance Optimization

* 엘에스웨어㈜ 소프트웨어연구소 연구개발본부
† 교신저자: 신동명(email: roland@lsware.co.kr)

접수일자: 2025.11.26. 심사완료: 2025.12.09.
게재확정: 2025.12.20.

1. 서 론

현대 웹 생태계는 거대 플랫폼 기업들이 사용자 데이터를 독점하는 중앙집중형 구조에서, 개인이 자신의 데이터에 대한 통제권을 갖는 탈중앙화 구조로 패러다임이 전환되고 있다[1]. 이러한 흐름을 주도하는 Solid(Social Linked Data) 프로젝트는 개인 데이터 저장소인 'Pod(Personal Online Data Store)'를 통해 데이터와 애플리케이션을 구조적으로 분리함으로써 진정한 의미의 데이터 주권을 실현하고자 한다[2].

하지만 Solid 애플리케이션이 실제 서비스 환경에서 상용화되기 위해서는 극복해야 할 기술적 난제들이 존재하며, 그중 가장 시급한 과제는 성능 최적화이다. 단일 서버 내에서 인증과 데이터 처리가 완결되는 기존 중앙집중형 웹 애플리케이션과 달리, Solid 아키텍처는 사용자의 WebID를 관리하는 ID 공급자(Identity Provider, IDP)와 실제 데이터가 저장된 Pod 서버가 물리적으로 분리된 환경을 전제로 한다. 이러한 분산 구조는 애플리케이션 구동 및 데이터 접근 시 노드 간 빈번한 네트워크 통신을 유발하여 필연적인 지연(Latency)을 초래한다.

특히 Solid-OIDC 프로토콜을 따르는 인증 프로세스는 WebID 역참조, OIDC 발급자(Issuer) 탐색, 토큰 교환 및 검증 등 다단계의 핸드셰이크 과정을 포함한다[3]. 모바일 환경이나 네트워크가 불안정한 상황에서 이러한 복잡한 인증 절차는 애플리케이션의 초기 로딩 속도를 크게 저하시켜 사용자 경험을 악화시키는 주요 원인이 된다.

따라서 본 논문에서는 Solid 사양을 준수하면서도 클라이언트 애플리케이션의 인증 처리 속도를 획기적으로 개선할 수 있는 아키텍처 설계 방안을 제안한다. 2장에서는 Solid의 핵심 사양과 구현체, 그리고 인증 메커니즘을 분석하고, 3장에서는 문제 해결을 위한 인증 최적화 아키텍처를 설계한다. 4장에서는 비교 실험을 통해 제안 방식의 성능 우위를 검증하며, 5장에서 결론을 맺는다.

2. 관련 연구

2.1 Solid 사양 (Solid Specification)

Solid는 W3C 표준을 기반으로 하는 웹 탈중앙화 프로젝트로, 데이터의 상호운용성(Interoperability)과 정교한 접근 제어를 핵심 가치로 둔다. Solid 생태계를 구성하는 주요 개념은 다음과 같다.

- Solid Protocol: LDP(Linked Data Platform)를 기반으로 HTTP 메서드를 사용하여 리소스를 생성, 읽기, 수정, 삭제하는 통신 규약이다[4, 5]. 모든 데이터는 기계가 이해할 수 있는 의미론적 구조를 갖춘 RDF(Resource Description Framework) 형식(주로 Turtle)으로 저장된다.
- Pod(Personal Online Data Store): Solid 생태계의 핵심 저장소인 Pod은 사용자의 프로필, 미디어, 문서 등을 저장하는 논리적 공간이다. 단순 파일 서버와 달리, 웹 접근 제어(WAC) 또는 접근 제어 정책(ACP)을 통해 리소스 단위의 정교한 권한 관리를 수행하여 데이터 주권을 보장한다[6, 7].
- WebID: Solid 생태계 내에서 사용자를 식별하는 글로벌 고유 식별자이다. HTTP URI 형식을 따르며, 해당 URI를 역참조하면 사용자의 프로필 문서(WebID Profile Document)에 접근할 수 있다[8].

2.2 Solid 구현 및 인증

Solid 사양을 준수하는 서버 구현체는 다양하

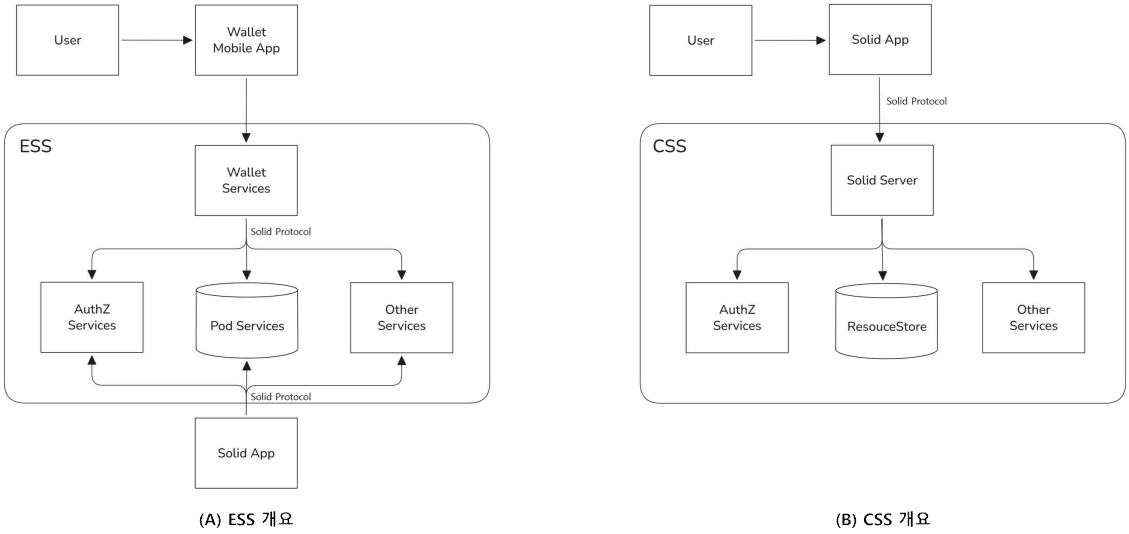


그림 1. Solid 사양 구현체 아키텍처 비교
Fig. 1. Architecture for Implementation of the Solid Specifications

게 존재하나, 본 논문에서는 가장 널리 활용되는 ESS(Enterprise Solid Server)와 CSS(Community Solid Server)를 비교 분석한다[9, 10].

또한, Solid 생태계에서 인증은 서버 구현체와 상관없이 표준을 기반으로 상호호환성이 보장된다.

2.2.1 Solid 구현체 비교

- ESS (Enterprise Solid Server): Inrupt 사가 개발한 Java 기반의 엔터프라이즈급 서버이다. 대규모 트래픽 처리, 보안성, 관리 편의성에 최적화되어 있다. Wallet 기반의 관리형 Pod을 제공하여 접근성을 높인 것이 특징이다.
- CSS (Community Solid Server): 오픈 소스 커뮤니티 주도로 개발된 Node.js 기반 서버이다. 모듈러(Modular) 아키텍처를 채택하여 기능 확장이 용이하고 설정 변경이 유연해 연구 및 개발 목적으로 널리 사용된다. 파일 시스템 기반의 저장소를 기본 지원하며, 사용자가 직접 데이터를 관리하고 제어하는 데 중점을 둔다.

그림 1은 ESS와 CSS의 아키텍처를 추상화한 그림이다. ESS는 DID 서비스와 유사하게 스마트폰 등을 활용한 Wallet을 통해 안전한 영역을 확보하고 관리형 서비스를 제공하는 반면, CSS는 사용자가 직접 인프라를 제어하는 구조를 지향한다.

2.2.2 Solid 인증

Solid의 인증은 OAuth 2.0과 OpenID Connect (OIDC)를 확장한 Solid-OIDC 프로토콜을 따른다[3]. 인증 흐름은 아래와 같다.

1. 식별: 사용자가 자신의 WebID URI를 애플리케이션에 입력한다.
2. 발급자 탐색: 애플리케이션은 WebID 프로필 문서를 역참조(HTTP GET)하여 solid:oidcIssuer 값을 파싱함으로써 사용자가 신뢰하는 IDP를 식별한다.

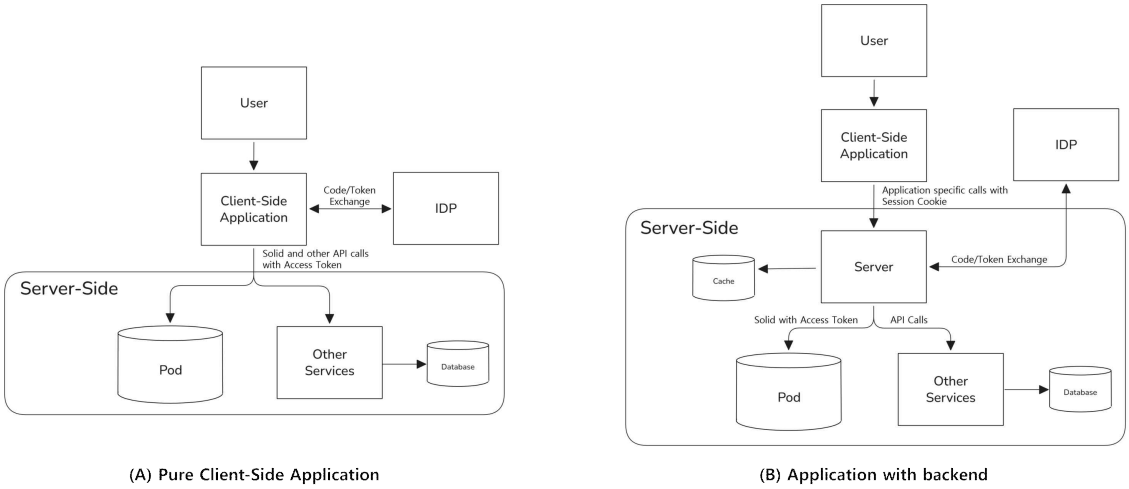


그림 2. Solid 애플리케이션 아키텍처
Fig. 2. Solid Application Architecture

3. 인증 요청: 식별된 IDP로 리다이렉트하여 사용자의 로그인을 수행한다.
4. DPoP (Demonstrating Proof-of-Possession): 보안 강화를 위해, 기존 Bearer 토큰 대신 클라이언트가 생성한 비공개 키로 서명된 DPoP 헤더를 사용하여 액세스 토큰을 요청하고 검증한다. 이는 토큰 탈취 시 재사용을 방지하는 핵심 메커니즘이다.

2.3 Solid 애플리케이션

현재 Solid 생태계에는 데이터 상호운용성을 활용한 다양한 클라이언트 애플리케이션이 존재한다. 그 중 대표적인 애플리케이션은 아래와 같다.

- Media Kraken[11]: 사용자의 Pod에 저장된 영화나 도서 컬렉션을 관리하는 미디어 관리 애플리케이션이다. 데이터가 특정 플랫폼에 종속되지 않고 사용자 Pod에 RDF로 저장되는 장점이 있으나, 대량의 메타데이터 로딩 시 발생하는 인증 오버헤드가 성능의 병목이 될 수 있다.
- dokieli[12]: 탈중앙화된 학술 출판 및 주석 달기를 지원하는 클라이언트 단 에디터다. 다른 사용자의 문서에 주석을 달면 Linked Data Notifications(LDN)으로 알림을 전송한다[13]. 문서 렌더링 시 외부 WebID와 연결된 다양한 데이터 소스를 실시간으로 가져와야 하므로 인증 및 네트워크 지연 최적화가 필수적이다.

3. Solid 애플리케이션 설계

Media Kraken, dokieli를 포함한 대부분의 Solid 애플리케이션은 그림 2의 A와 같이 클라이언트 단에서 직접 Pod 및 기타 API를 호출하는 구조를 가진다. 이는 소규모 서비스에는 적합할 수 있으나, 복합적인 기능을 제공하는 대규모 서비스에서는 성능 저하의 원인이 된다.

특히, 대규모 서비스 환경에서 각 서비스가 Pod에 접근하여 데이터를 사용해야 한다면 그림

2의 B와 같이 서버 단 컴포넌트를 도입하여 캐싱 등의 최적화 기법을 적용함으로써 응답 시간을 단축할 수 있다. 본 논문에서는 CSS 환경을 기반으로 인증 처리 시간을 최소화하는 서버 기반 아키텍처를 제안한다.

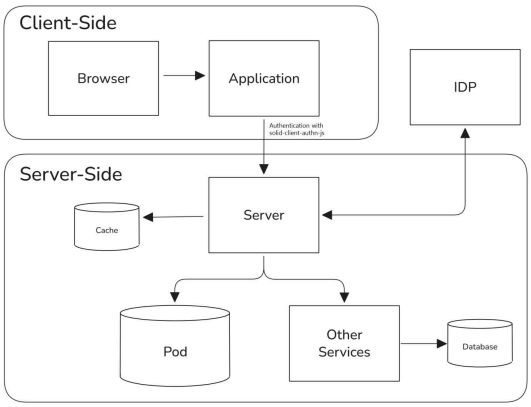


그림 3. 제안 시스템 아키텍처
Flg. 3. Proposed System Architecture

3.1 시스템 아키텍처

제안 시스템은 그림 3과 같이 세 가지 핵심 컴포넌트로 구성된다.

- 클라이언트 단 애플리케이션: 사용자는 브라우저를 통해 애플리케이션에 접근하고, 로그인한다. 애플리케이션은 Inrupt 사가 관리하는 solid-client-authn-js 라이브러리를 통해 인증을 수행한다.
- ID 공급자: CSS에 내장된 표준 ID 공급자를 사용하여 사용자의 인증 정보를 검증하고 액세스 토큰을 발급한다.
- 서버 단 백엔드: 서버는 애플리케이션의 Pod 접근을 중개하고, 다수의 서비스를 통합한다. Cache를 사용하여 액세스 토큰을 캐싱한다.

사용자는 클라이언트 단 애플리케이션을 통해 로그인을 수행하며, 인증 완료 후 백엔드 서버를 거쳐 Pod 데이터에 접근한다. 본 연구에서 정의하는 인증 처리 시간은 (1) IDP 연동, (2) 토큰 발급 및 교환, (3) 실제 리소스 접근 승인에 이르는 전체 소요 시간을 의미한다.

3.2 인증 처리 시간 최적화 전략

본 논문의 핵심 전략은 ESS가 Wallet을 통해 클라이언트 측에 안전 영역을 확보한 개념을 차용하여, 백엔드 서버를 통해 신뢰할 수 있는 안전 영역(Trusted Zone)을 확장하는 것이다.

DPoP-bound 액세스 토큰은 요청마다 클라이언트의 서명 생성 및 서버의 서명 검증 과정을 요구하므로 연산 비용이 높다. 반면, 제안하는 아키텍처의 백엔드 서버 내(안전 영역)에서는 외부 탈취 위험이 통제되므로, 상대적으로 오버헤드가 적은 Bearer 액세스 토큰의 사용이 가능하다.

서버가 IDP로부터 발급받은 액세스 토큰을 인메모리 캐시(In-memory Cache)에 저장하여 재사용함으로써, 요청마다 반복되는 서명 검증 및 토큰 발급 절차를 생략할 수 있다. 이는 적절한 보안 정책(예를 들어 토큰 롤링, 짧은 토큰 수명 주기)과 결합될 때 보안성을 저해하지 않으면서도 비약적인 성능 개선을 가능하게 한다.

4. 실험 및 평가

본 논문에서 제안한 인증 최적화 전략의 유효성을 검증하기 위해, (1) DPoP-bound 액세스 토큰, (2) 일반 Bearer 액세스 토큰, (3) 캐싱된 액세스 토큰의 인증 처리 시간을 비교 측정하였다. 캐싱되지 않은 방식은 리소스 접근 시마다 신규 토큰을 발급받으며, 제안 방식은 인메모리 데이터베이스를 캐시 스토리지로 활용하였다.

4.1 실험 결과

실험환경은 그림 4와 같이 클라우드 인스턴스(Pod 및 IDP) 1대와 서버(백엔드 및 클라이언트) 1대로 구성하였다.

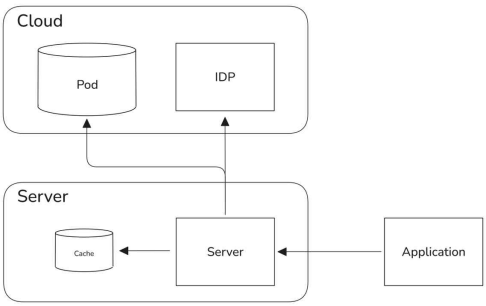


그림 4. 실험환경
Fig. 4. Experimental Environment

- 클라우드 인스턴스: 4 vCPU, 8GB Memory
- 서버: AMD Threadripper PRO 3975WX, 256GB Memory
- 소프트웨어: CSS v7.0.0, Redis v8.4.0, solid-client-authn-node v3.1.1

실험은 각 인증 방식에 대해 100회의 리소스 조회 요청을 수행하였으며, 데이터의 신뢰도를 위해 최댓값과 최솟값을 제외한 평균 처리 시간을 소수점 셋째 자리 반올림하여 기록하였다. 실험 결과는 표 1과 같다. DPoP-bound 액세스 토큰의 경우 193.1ms로 가장 오래 걸렸고, Bearer 액세스 토큰이 54ms, 캐싱된 액세스 토큰이 20.61ms로 측정되었다. 평균 처리 시간 계산에서 제외된 최댓값의 경우, DPoP-bound 액세스 토큰과 Bearer 액세스 토큰 방식에서는 각 실험의 첫 요청이었다. 다른 요청에 비해 2~3배 크게 측정되었는데 이는 WebID 역참조 및 다단계 핸드셰이크 과정으로 인해 발생한 오버헤드다. 실제로 인증과정에서 IDP로 요청을 보내지 않는 캐싱 방식에서는 첫 요청이 다른 요청들과 크게 다르지 않았다.

4.2 평가

DPoP-bound 액세스 토큰은 클라이언트 측의 개인키 서명 생성과 서버 측의 서명 검증 프로세스로 인해 가장 긴 처리 시간을 기록하였다. 이를 통해 DPoP-bound 액세스 토큰은 보안성은 높지만, 성능상 병목이 될 수 있는 것을 확인하였다.

반면, 본 논문에서 제안한 캐싱된 액세스 토큰 방식은 초기 1회 발급 이후 유효 기간 내에는 캐싱된 토큰을 재사용하므로, DPoP 방식 대비 약 89%, 일반 Bearer 방식 대비 약 61%의 성능 향상을 보였다. 이를 통해 서버 단의 안전 영역을 활용한 캐싱 전략이 Solid 애플리케이션의 응답 속도를 획기적으로 개선할 수 있음을 확인하였다.

5. 결론

본 논문에서는 탈중앙화 환경인 Solid 애플리

표 1. 인증 처리 시간 비교
Table 1. Comparison of Authentication Processing Times

(단위: ms)

구분	DPoP-bound			Bearer			Caching		
	평균	최대	최소	평균	최대	최소	평균	최대	최소
1회	193.68	595.34	179.06	56.54	218.33	47.71	21.13	32.43	17.88
2회	194.27	609.11	177.89	56.97	286.88	48.79	21.45	31.13	18.35
3회	191.35	302.09	177.1	48.48	137.33	41.85	19.24	30.79	16.8
평균	193.1	508.18	178.02	54	214.18	46.12	20.61	31.45	17.68

케이션의 성능 저하 문제를 해결하기 위해, 서버 단 캐싱을 활용한 인증 최적화 아키텍처를 제안하고 검증하였다. 실험 결과, 서버 내 안전 영역에서 토큰을 캐싱하고 재사용하는 전략이 표준 DPoP 방식 대비 인증 처리 시간을 획기적으로 개선되는 것을 확인하였다.

이는 Solid 생태계가 성능적 한계를 극복하고 대규모 서비스로 확장될 수 있는 가능성을 보여준다. 향후 연구에서는 캐싱된 토큰의 탈취 위협에 대비하여, 토큰 갱신 주기를 동적으로 조절하거나 서버 간 보안 채널을 강화하는 등 더욱 안전한 캐싱 운영 정책에 대한 심화 연구를 진행할 계획이다.

본 연구는 문화체육관광부 및 한국콘텐츠진흥원의 2025년도 신기술 융합 저작권 기술개발 사업으로 수행되었음(과제명 : Web3.0 탈중앙화 환경에서 창작자간의 저작권 이용허락 거래 자동화 기술 개발, 과제번호 : RS-2024-00441360, 기여율 : 100%)

참 고 문 헌

[1] Y. Chen, J. I. Richter, P. C. Patel, “Decentralized Governance of Digital Platforms”, Journal of Management, 47(5), pp1305-1337, 2020.02, DOI : 10.1177/0149206320916755

[2] H. J. Pandit, “Making Sense of Solid for Data Governance and GDPR”. Information, 14(2), 114, 2023.02, DOI : 10.3390/info14020114

[3] A. Coburn, E. Pavlik, D. Zagidulin, “Solid-OIDC”, 2022, Solid Project, https://solidproject.org/TR/oidc

[4] S. Capadisli, T. Berners-Lee, K. Kjernsmo, “Solid Protocol”, 2024, Solid Project, https://solidproject.org/TR/protocol

[5] A. Sambra, E. Mansour, S. Hawke, M. Zereba, N. Greco, A. Ghanem, D. Zagidulin, A. Aboulnaga, T. Berners-Lee, “Solid : A Platform for Decentralized Social Applications Based on Linked Data”, 2016, MIT CSAIL & Qatar Computing Research Institute, http://emansour.com/research/lusail/solid_protocols.pdf

[6] T. Berners-Lee, H. Story, S. Capadisli, “Web Access Control”, 2024,Solid Project, https://solidproject.org/TR/wac

[7] M. Bosquet, “Access Control Policy (ACP)”, 2022, Solid Project, https://solidproject.org/TR/acp

[8] A. Sambra, H. Story, T. Berners-Lee, “WebID 1.0”, 2014, Solid Project, https://www.w3.org/2005/Incubator/webid/spec/identity/

[9] Inrupt, “Wallet Architecture and Application Technical Flows”, 2025, Inrupt, https://docs.inrupt.com/wallet/architecture-and-flows

[10] Community Solid Server, “Architecture overview”, 2022, Community Solid Server, https://communitysolidserver.github.io/CommunitySolidServer/latest/architecture/overview/

[11] NoelDeMartin, “Media Kraken”, 2025, https://noeldemartin.github.io/media-kraken

[12] dokieli, “dokieli”, 2025, https://dokie.li/

[13] JP. Calbimonte, D. Calvaresi, M. Schumacher, “Multi-Agent Interactions on the Web through Linked Data Notifications”, Springer, 2018, DOI : 10.1007/978-3-030-01713-2_4

저 자 소 개

장성일(Sung-Il Jang)

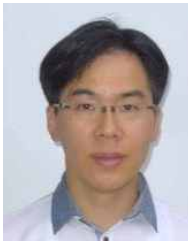
2019.08 : 숭실대학교 컴퓨터학과 석사
2021.08 : 숭실대학교 소프트웨어학과 박사수료
2021.09-현재 : 엘에스웨어㈜ 수석연구원
<주관심분야> 시스템 프로그래밍, 분산 컴퓨팅, 블록체인

홍두표(Du-Pyo Hong)

2024.02 : 숭실대학교 컴퓨터학과 석사
2024.01-현재 : 엘에스웨어㈜ 주임연구원
<주관심분야> 클라우드, 빅데이터, 블록체인

조용준(YongJoon Joe)

2011.03 : 큐슈대학교 전기정보공학과 학사
2013.03 : 큐슈대학교 정보학부 석사
2016.03 : 큐슈대학교 정보학부 박사 수료
2013.04-2016.03 : 일본 학술진흥원 특별연구원
2016.04-현재 : 엘에스웨어㈜ 소프트웨어연구소
연구개발본부 기술이사
<주관심분야> 오픈소스, 저작권, 병렬·분산 컴퓨팅, 게임이론, 분산 제약 최적화 문제



신동명(Dong-Myung Shin)

2003.08 : 대전대학교 컴퓨터공학과 박사
2001-2006 : 한국정보보호진흥원(KISA)
응용기술팀 선임연구원
2006-2014 : 한국저작권위원회
저작권기술팀 팀장
2014-2016 : 한국스마트그리드사업단
보안인증팀 팀장
2016-현재 : 엘에스웨어㈜ 소프트웨어연구소
연구개발본부 연구소장/상무이사
<주관심분야> 오픈소스 라이선스, 저작권 기술, 시스템/네트워크 보안, SW 취약점 분석·감정, 블록체인 기술