KIPS Trans. Comp. and Comm. Sys.
Vol.11, No.3 pp.91~98
ISSN: 2287-5891 (Print), ISSN 2734-049X (Online)

https://doi.org/10.3745/KTCCS.2022.11.3.91

The Reliable Communication Method for Self-Sovereign Identity Ecosystems

Gyu Hyun Chio† · Geun-Hyung Kim††

ABSTRACT

With the recent increase in interest in metaverse in which virtual and physical spaces are digitally fused, many activities in physical spaces are expected to take place in web-based virtual spaces. Therefore, there is a need for research on a self-sovereign identity system that can secure privacy and mutual trust in a DID(decentralized identifier)-based virtual space environment. We, in this paper, developed and validated a reliable communication method consisting of DIDComm messages, a procedure for generating distributed identifiers, asymmetric keys, and DID documents based on Hyperledger Indy and DIDComm open sources. The developed communication method can be applied to verify each other by exchanging additional information and verifiable credentials for trust among communication participants.

Keywords : Self-Sovereign Identity, Decentralized Identifier, Public-Key Cryptography, DID Auth, DIDComm

자기주권 신원 생태계를 위한 신뢰할 수 있는 통신 방법

최 규 현† · 김 근 형††

요 약

최근 가상공간과 물리공간이 디지털 융합된 메타버스에 대한 관심이 높아지면서 물리공간에서의 많은 활동이 웹 기반의 가상공간에서 이루어질 전망이다. 따라서 분산 식별자 기반 가상공간 환경에서 사생활과 상호 신뢰를 확보할 수 있는 자기주권 신원시스템에 대한 연구가 요구된다. 본 논문에서는 분산 식별자 기반 자기주권 신원시스템을 위해 Hyperledger Indy와 DIDComm 오픈 소스를 기반으로 분산 식별자, 비대칭 키, DID 문서를 생성하여 블록체인과 지갑에 등록하는 절차, DIDComm 메시지로 구성된 신뢰할 수 있는 통신 방법을 개발하고 검증하였다. 이는 통신 참여자 간의 신뢰를 위한 추가적인 정보 교환 및 검증 가능한 자격증명을 교환하여 서로를 검증하는데 활용이 가능하다

키워드 : 자기주권 신원, 분산 식별자, 공개키 암호, 분산 식별자 인증, 분산 신원 통신

1. 서 론

최근 블록체인 기술을 활용한 여러 애플리케이션이 개발되고 있으며, 그중 탈중앙화 신원인증 시스템의 기반이 될 DID (Decentralized Identifier) 기술에 관한 관심이 높아지고 있다. DID 기술에 대한 관심이 높아지는 이유는 기존 중앙 집중형 신원인증 시스템의 한계와 해킹 문제 때문이다. 중앙 집중형 신원인증 시스템의 경우 특정한 중앙 서버에 자신의 정보를 제공하여 자신의 신원을 증명하는데, 이는 특정한 중앙 서버에 개인 정보가 몰리는 현상을 발생시킨다, 또한 자신의 정보를 보유한 중앙 서버가 해킹을 당하면 본인의 의사와 상관없이 개인 정보가 유출되거나 중앙 서버 관리자가 사용자 정보를 임의로 사용할 수 있는 등 여러 가지 문제점들이 발생하기 시작했다. 인터넷 사용이 보편화되고 사용자가 증가하면서 인터넷에서 사용되는 개인 정보량도 증가하였고, 그에 따라 해킹으로 인한 피해 규모 또한 늘어나고 있다[1].

DID 기술은 앞에서 언급한 문제를 해결하기 위해 개인 정보를 특정한 기관이 통제하는 것이 아닌, 본인이 통제하는 것으로 본인의 개인 정보를 스스로 관리하며 인증을 가능하게 만드는 자기 주권 신원(Self-Sovereign Identity)을 가능하게 만들어 주었다[2, 3]. 또한 DID기반 DIDComm은 자기주권 신원 생태계에서 신뢰를 제공할 수 있는 통신 방법으로 고려할 수 있다. DIDComm은 사용자 간 또는 사물 간 통신에서 중계 서버가 필요 없는 통신 방식으로 중앙 서버의 관여를 배제할 수 있다. DID와 DIDComm 기술은 앞으로 더 발전할 인터넷 환경에 강력한 인증 및 통신 메커니즘을 제공해

※ This work was supported by the National Research Foundation of Korea(NRF) grant funded by Korea government(MSIT) (No. NRF-2021 R1F1A1047573).
※ 이 논문은 2021년 한국정보처리학회 ACK 2021의 우수논문으로 "자기주권 신원 기반 피어-투-피어 통신 메커니즘 분석"의 제목으로 발표된 논문을 확장한 것임.
† 준 회 원 : 동의대학교 게임애니메이션공학전공 학부생
†† 종신회원 : 동의대학교 게임공학전공 교수
Manuscript Received : December 28, 2021
Accepted : January 24, 2022
* Corresponding Author : Geun-Hyung Kim(geunkim@deu.ac.kr)

줄 것이고, 더 나아가 가상공간과 물리공간의 융합이 활성화 될 메타버스 환경에서 신뢰할 수 있는 디지털 자산 및 가치의 교환에 필수적인 강력한 자기 인증 및 통신 수단으로 활용될 것이다.

본 논문에서는 탈중앙화 웹 기반 자기주권 신원 생태계의 기반이 되는 분산 식별자와 분산 식별자 인증 기술, 및 분산 신원 간 안전하고 비공개적이고 전송계층에 독립적인 DIDComm에 대해서 살펴보고 DIDComm 오픈소스를 분석하고 DIDComm과 Hyperledger Indy와 같은 블록체인 기반 DID 네트워크와 상호작용하는 방법을 제시한다. 논문의 구성은 다음과 같다. 2장에서 관련 기술을 살펴보고 3장에서 DIDComm 코드 분석 및 검증 결과를 기술하였으며 4장에 기존 방식과의 차이점을 설명하고 5장에서 결론을 맺는다.

2. 관련 기술

2.1 분산 식별자(Decentralized Identifier)

분산 식별자는 자기주권 신원인증 시스템에서 검증가능한 자격증명의 핵심요소로 W3C에서 표준이 이루어진 새로운 유형의 식별자[4]이다. 분산 식별자는 중앙 기관에 무관하게 소유자가 생성하는 특별한 종류의 식별자이다. 분산 식별자는 다음의 요구사항을 모두 만족하는 식별자이다[5].

- 탈중앙화(decentralized): 식별자의 중앙 발행기관이 없어야 한다.
- 영속성(persistent): 본질적으로 영구적이어야 하며 하부 기관의 지속적인 운영을 요구하지 않아야 한다.
- 암호로 검증 가능(cryptographically verifiable): 식별자의 제어권을 암호로 증명할 수 있어야 한다.
- 해석 가능(resolvable): 식별자에 대한 메타데이터를 검색할 수 있어야 한다.

분산 식별자로 식별되는 개체를 인증하기 위해서 분산 식별자 하나에 관련된 공개키(public key)와 개인키(private key) 쌍이 이용된다. 분산 식별자는 블록체인 또는 다른 탈중앙 네트워크에 저장된 공개키의 주소 역할을 한다. 모든 분산 식별자는 관련된 하나의 DID 문서(DID document)가 있다. DID 문서에는 DID 주체(subject)에 대한 메타데이터가 기술되며 이 메타데이터는 분산 식별자에 의해 식별되는 DID 문서를 통해 설명하려는 자원에 대한 정보이다. 예로 개인에 대한 분산 식별자와 관련한 DID 문서는 일반적으로 암호화 키, 인증방법 및 통신할 대상자와 신뢰할 수 있는 상호작용에 참여하는 방법을 설명하는 메타 데이터를 포함한다. 분산 식별자와 DID 문서를 제어하는 엔티티를 DID 컨트롤러라 한다. Fig. 1은 DID 컨트롤러가 DID 주체와 동일한 경우의 분산 식별자와 DID 문서와의 관계를 보인다.

2.2 분산 식별자 인증(DID Auth: DID Authentication)

분산 식별자 인증(DID Auth)은 상대방에게 분산 식별자

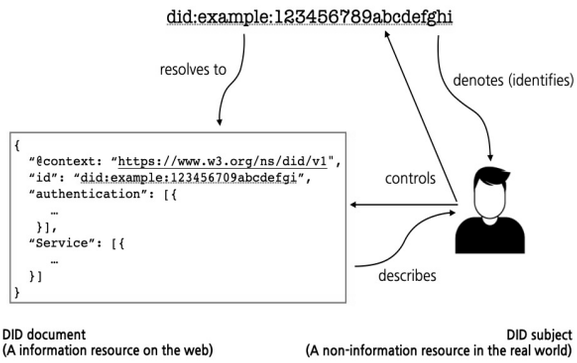


Fig. 1. The Relationship Between DID, DID Document, and DID Subject

에 대한 소유권을 증명하는 과정으로 분산 식별자의 소유자가 분산 식별자와 관련한 개인키를 제어할 수 있음을 증명함으로써 자신을 인증하기 위한 표준이다. 분산 식별자 인증을 위해서 DID 문서에 포함된 공개키와 인증(authentication) 방식 항목을 사용한다[6].

분산 식별자 인증은 시도-응답(Challenge-Response) 인증 방식을 활용한다. 시도-응답 인증 방식의 경우 인증이 필요한 사용자에게 일회성 challenge를 상대방의 공개키로 암호화하여 전송한 뒤, 사용자가 이를 개인키로 확인하고 response을 만들어 서명을 추가해 전송하는 것으로 이루어지는 인증 방식이다. 이때 challenge를 보낼 경우 예측할 수 없는 일회성 값을 사용한다. 이는 공개키와 개인키를 사용하는 분산 식별자 환경에 적합한 인증 방식이며 일회성이기 때문에 누군가의 해킹이 이루어져 추가적인 response이 전송되더라도 이미 사용된 값이기 때문에 재사용이 불가능하여 안전하다.

분산 식별자 인증은 다음 과정을 통해 이루어진다. 먼저 사용자는 인증이 필요한 기관에 자신의 분산 식별자를 전달하고 기관은 전달받은 분산 식별자를 통해 블록체인에 저장된 DID 문서를 얻은 후 DID 문서 내의 인증 항목과 인증에 사용하는 공개키를 확인한다. 기관은 DID문서를 통해 확인한 사용자의 공개키를 활용하여 미리 생성해둔 challenge를 암호화해 사용자의 서비스 종단점을 통해 사용자에게 전달한다. challenge를 전달받은 사용자는 기관에서 전송한 특정 메시지를 자신의 개인키로 복호화한 후 확인하고 확인한 내용을 전달하기 위해 response를 만든다. 이때 메시지 무결성을 확인하기 위해 자신의 개인키로 암호화하며 기관에 전달한다. 기관에서는 해당 문서를 다시 사용자의 공개키로 복호화하여 response를 확인하는 것으로 분산 식별자 인증이 완료된다. 해당 과정을 통해 분산 식별자 소유자가 개인키를 제어할 수 있음을 증명할 수 있다. Fig. 2는 사용자의 분산 식별자 인증 과정을 나타낸다.

2.3 통합 해석기(Universal Resolver)

통합 해석기는 W3C에서 표준화 중인 분산 식별자[4]와 분산 식별자 해석[7] 표준에 따라 다양한 분산 식별자 메소드를

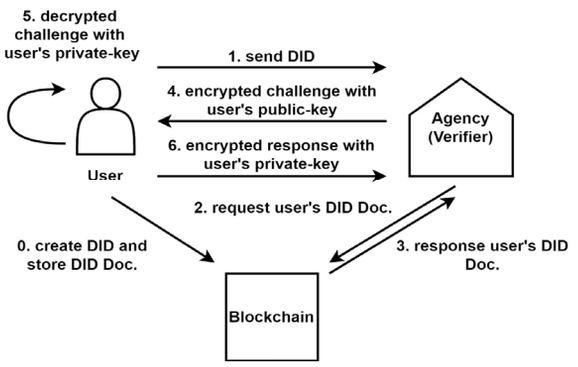


Fig. 2. DID Auth Procedure

갖는 다양한 분산 식별자의 메타데이터를 일관된 방식으로 얻을 수 있도록 한다[8]. 앞으로도 자기주권 신원 생태계가 커질수록 분산 식별자를 위한 다양한 블록체인 네트워크가 만들어 질 것이다. 다양한 블록체인 네트워크 환경에서 개발된 분산 식별자에 대한 메타데이터를 가져오기 위해서는 DID 네트워크에 특화된 별도의 API를 사용하여야 하는 문제가 존재한다. 이를 해결하기 위해 개발된 기술이 통합 해석기로 분산 식별자만으로 해당 분산 식별자의 메타데이터 정보를 검색해 표준화된 DID 문서를 생성한다. 이는 자기주권 신원 생태계에서 블록체인 기반 DID 네트워크 수의 증가와 관계없이 작동할 수 있으며, 자기주권 신원 생태계에서 상대방의 분산 식별자의 메소드에 관계없이 누구나 상대방의 분산 식별자로 DID 문서를 얻을 수 있도록 한다.

2.4 DIDComm

DIDComm은 분산 식별자의 분산 설계를 기반으로 구축된 안전한 개인적인 통신 방법을 제공하는 것이 목적이다[9]. DIDComm은 사람, 사물, 또는 기관에 의해 제어되는 소프트웨어 에이전트 간의 안전한 통신 채널(secure communication channel)을 생성한다. 안전한 통신을 위해 DIDComm과 다른 강력한 방법이 이미 존재한다. 그러나 대부분의 안전한 통신 방법들은 키 저장소, 신원 제공자, 신원 인증 기관, 브라우저, 앱 공급업체와 유사한 중앙 집중화된 요소에 종속적이다. 이들은 특정한 하나의 전송 프로토콜에 결합되어 있어 광범위한 형식에서 온라인 및 오프라인, 단방향/양방향 송수신으로 사람과 사물 간 대화에 동일한 방식을 적용시키는 것은 어렵다. 결과적으로 기관과 사용자 간의 비대칭 관계를 영구화한다.

DIDComm은 상호 인증 채널을 구성하며 주어진 분산 식별자의 개인키에 대한 제어권이 해당 분산 식별자로 식별되는 사람, 사물 또는 기관의 진위를 증명하는 것이다. 즉 DIDComm 구조는 통신에 참여하는 모든 주체가 신뢰적인 통신을 위해 상호 인증하는 방법을 제공하며, 기본 패러다임은 메시지 기반, 비동기식, 단방향 통신으로 세션 개념을 가지지 않는다.

DIDComm은 분산 식별자를 사용한 메시지 타입을 정의하고 있으며 DIDComm이 제공하는 API를 사용하는 것으로 메시지를 만들고 암호화, 복호화, 메시지를 읽는 것을 가능하게 한다.

1) 구동 방식

DIDComm을 사용한 통신은 다음과 같은 방식으로 진행된다. 통신에 참여하는 피어는 분산 식별자로 식별되고 해당 DID 문서가 블록체인에 저장되며 서로 상대방의 분산 식별자를 알고 있어야 한다.

통신을 원하는 사용자 A가 사용자 B에게 메시지를 전달하기 위해선 사용자 B의 분산 식별자가 필요하며, 이를 사용해 블록체인에서 DID 문서를 확인해 사용자 B의 공개키와 서비스 종단점을 확인한다. 전송할 내용을 작성하여 메시지를 만들고 이를 DID 문서에서 확인한 상대방의 공개키를 사용하여 암호화한 뒤, 이를 사용자 B의 서비스 종단점으로 전송한다. 이때 메시지 무결성을 위해 본인의 서명을 추가하여 전송할 수 있다. 메시지를 수신한 사용자 B는 해당 메시지를 본인의 개인키를 사용해 복호화하여 메시지 내용을 확인한다. 사용자 B는 사용자 A의 분산 식별자를 사용하여 DID 문서를 가져오고, 여기서 공개키 정보와 서비스 종단점을 확인한다. 이때 메시지에 서명이 있을 경우 가져온 공개키를 사용하여 메시지 무결성을 확인한다. 다음 확인한 공개키와 서비스 종단점을 사용해 위와 같은 행동을 반복하여 결과적으로 사용자 A에게 메시지를 전송하는 것으로 연결을 종료한다.

서로의 공개키와 통신을 위한 서비스 종단점을 가지고 있으므로 안전한 통신 채널 생성이 가능하다. DIDComm 기반 통신은 통신 참여자의 분산 식별자와 서비스 종단점으로 메시지를 전송할 수 있어 중앙의 서버를 통하지 않는 통신을 가능하게 만들어 주며, 이는 분산 식별자의 탈중앙화 특징을 유지한다. 그러나 앞의 과정만으로는 분산 식별자의 유효성만을 확인할 수 있을 뿐 분산 식별자의 사용자가 신뢰할 만한 인물인지는 알 수 없다. 이러한 문제점의 해결이 필요할 경우 서로를 인증하기 위한 추가적인 정보 교환이 요구된다.

2) 메시지 유형

DIDComm은 세 가지 메시지 유형을 정의한다. 첫 번째는 암호화되지 않은 일반적인 텍스트 메시지를 지원한다. 일반적인 텍스트 메시지의 경우 암호화되지 않은 메시지로 안정성과 기밀성이 떨어져 실제로 사용하기 힘든 메시지 유형이다. 대부분 테스트를 위해 사용된다. 두 번째 메시지 유형은 서명된 메시지를 지원한다. 서명된 메시지의 경우 본인의 메시지를 자신의 개인키로 암호화하는 것을 의미하며 해당 메시지를 작성한 사람이 개인키를 소유하고 있음을 증명하는 데도 사용한다. 개인키를 사용하여 서명을 하고 이를 서명자의 공개키로 복호화하는 것으로 메시지의 무결성을 확인한다. 마지막 메시지 유형은 암호화된 메시지이다. 공개키를 통해 암호화되며 DIDComm의 경우 상대방과의 안전한 통신

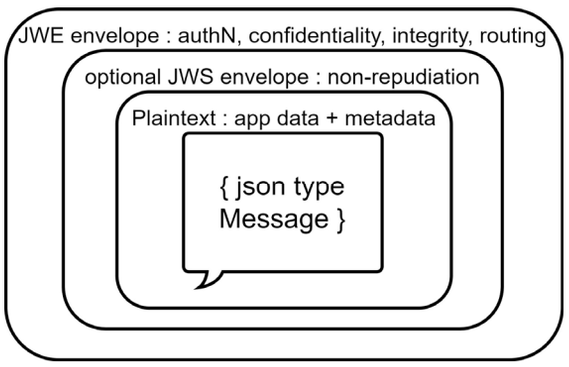


Fig. 3. DIDComm Message Type

을 위해 상대방의 공개키를 사용하여 메시지를 암호화할 때 사용된다. 세 가지 메시지 유형외에 메시지의 무결성 확인을 위해 암호화와 서명을 함께 사용하는 것 또한 가능하며 이때 메시지에 서명을 한 뒤, 메시지의 암호화를 하게 된다. Fig. 3은 DIDComm 메시지의 유형을 나타낸다.

3) 메시지 구조

DIDComm의 일반 텍스트 메시지 구조는 Fig. 4와 같은 여러 속성으로 구성된다.

typ은 메시지 유형을 나타내는 것으로 암호화 메시지인지 서명 메시지인지 암호화 없는 텍스트 메시지인지를 표현한다. id는 메시지의 고유한 값으로 발신자에 대해 고유한 값을 가진다. type은 body에 작성될 실제 메시지가 어떠한 유형으로 작성되어 있는지를 명시한다. from과 to는 송신자와 수신자의 분산 식별자 정보를 가진다. created_time은 메시지 생성 시간, expires_time은 메시지의 만료 시간을 명시한다. body영역은 실제 메시지가 작성되는 곳으로 type에서 작성된 메시지 유형에 따라 값을 가지며 이는 JSON 객체 형식이어야 한다. id, type, body 속성은 필수적으로 사용되어야 하며, 나머지 속성은 필요에 따라 선택적으로 사용이 가능하다.

```
message :
{
        "id":"1234567890",
        "typ":"application/didcomm-plain+json",
        "type":"<message-type-uri>",
        "from":"6763PStb3HGL4HF6Axtsgj",
        "to":["9DnpiHG5oatGEG2PnyYiSh"],
        "craeted_time":1640517414,
        "expires_time":1640624581,
        "body":
                {
                        "title":"body"
                }
}
```

Fig. 4. Plain Text Message Structure

3. DIDComm 코드 분석 및 테스트 결과

3.1 DIDComm 코드 분석

DIDComm은 JSON 형식의 메시지 유형, 공개키와 개인키를 사용한 암호화 및 복호화, 마지막으로 블록체인을 통한 분산 식별자를 사용하여 구현하였다. DIDComm 기능의 구현은 Hyperledger 재단에서 관리 중인 indy-sdk 오픈 소스(이하 libindy)[10]와 메시지 생성 기능을 위해 Java로 구현한 오픈 소스인 didcomm-jvm[11]을 사용하였다. Fig. 5는 libindy를 사용하여 DIDComm을 구현하기 위해 만든 클래스 다이어그램으로 DIDComm 사용자인 DIDuser 클래스, 메시지 암호화 및 복호화를 담당하는 LibindyDIDComm 클래스, 암호화와 결과물을 관리하기 위한 PackEncryptMessageResult 클래스, 복호화의 결과물을 관리하기 위한 UnpackMessageResult 클래스, DIDuser에 종속적인 Pool Resolver클래스로 구성된다. Pool Resolver 클래스는 블록체인에서 정보를 가져오기 위해 기본적으로 사용자의 분산 식별자 서명에 필요하기 때문에 DIDuser 클래스에 종속적인 관계이다. LibindyDIDComm 클래스는 DIDuser클래스와 독립적으로 동작하도록 분리하여 다른 사용자가 사용하더라도 문제없이 동작하도록 하였다.

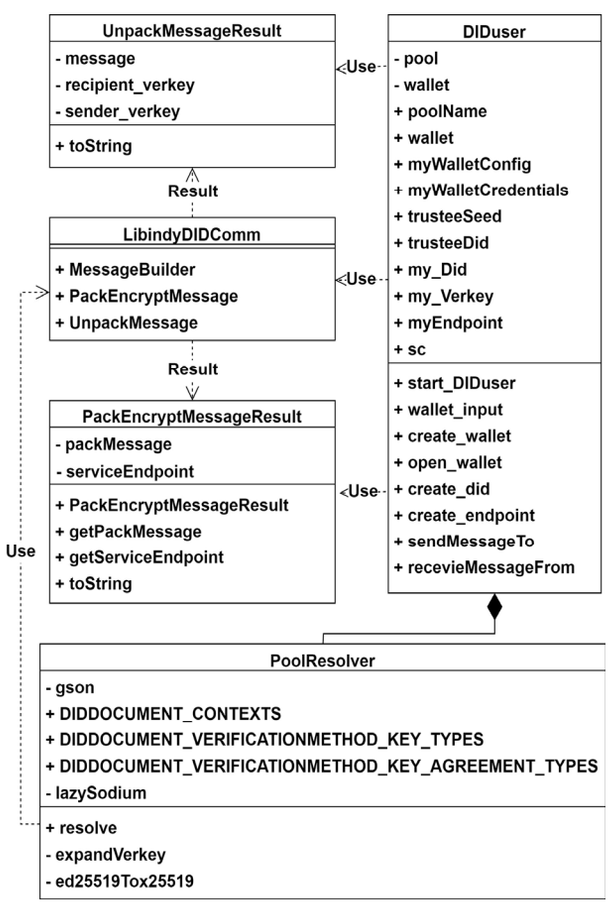


Fig. 5. Class Diagram Related to DIDComm

DIDComm 기능 구현은 DIDComm 메시지 관련 오픈 소스인 didcomm-jvm과 분산 식별자 관련 오픈 소스인 libindy 라이브러리(이하 libindy)를 연동하도록 설계하였으나 libindy와 DIDComm 메시지 기능 간 연동에 문제가 있어 libindy의 메시지 암호화와 메시지 복호화 기능을 사용하였다. 연동 구조로 동작하지 못한 이유는 DIDComm 기능에서 분산 식별자 관련 개인키 접근을 직접 할 수 없기 때문이다. 오픈 소스 didcomm-jvm 구현은 공개키와 개인키를 생성하여 따로 저장한 후 개인키를 직접 접근하여 JSON 메시지의 복호화 및 서명에 사용한다. 그러나 libindy는 공개키와 개인키를 생성하는 경우 개인키를 안전하게 관리하기 위해서 개인키 생성과 동시에 지갑에 저장하고 애플리케이션이 개인키를 직접적으로 접근할 수 있는 인터페이스를 제공하지 않아 개인키에 대한 직접적인 접근을 필요로 하는 didcomm-jvm 오픈 소소와 연동할 수 없다. 이는 개인키의 보안성을 높이기 위해서 libindy에서 개인키의 직접 접근 기능을 제공하지 않았다. 개인키의 직접 접근 기능을 제공하기 위해 지갑 기능을 수정할 수 있으나 보안상의 문제가 있어 채택하지 않았다. 공개키와 개인키를 필요로 하지 않는 메시지의 생성 부분을 didcomm-jvm 오픈 소스를 사용해 구현하였으며, 메시지의 암호화 및 복호화 부분의 경우 libindy의 암호화 복호화 기능을 사용하여 구현하였다.

오픈 소스 indy-sdk는 분산 식별자를 사용하기 위해 여러 API(Application Programming Interface)를 제공하고 있으며 wrapper를 통해 Java, C++, JavaScript언어에서도 사용할 수 있도록 구현되었다. 오픈 소스 indy-sdk를 사용하기 위해서는 API의 정보가 담겨있는 libindy를 참조하여 사용해야 한다. 제공하는 API 기능은 분산 식별자의 관련한 지갑 생성, 블록체인과의 연결을 위한 풀 연결과 분산 식별자 생성 및 비 대칭키 쌍 생성 등이 있다. DIDComm을 사용한 통신을 위해서는 기본적으로 분산 식별자가 필요하므로 libindy를 사용하였다.

Fig. 6은 DIDuser가 DIDComm 메시지 생성, 송신, 수신과 관련한 시퀀스를 보인다. DIDComm 메시지를 생성하기 전에 이루어지는 분산 식별자 및 DID 문서 생성 등과 관련한 기능은 sendMessageTo 함수 호출 전에 이루어진다. 먼저 블록체인과의 연결을 위해 블록체인에 대한 정보가 들어간 풀(Pool) 구성을 만들어 연결한다. 다음 분산 식별자와 비 대칭키 저장을 위한 지갑을 생성하고 분산 식별자 생성과 비 대칭키를 생성하여 지갑에 저장하고 해당 정보를 풀을 사용해 블록체인에 저장한다. 이 과정이 완료되면 DIDComm메시지를 생성하여 통신할 수 있는 기반이 생성된다. DIDuser는 LibindyDIDComm 클래스의 sendMessageTo 메소드를 호출하여 메시지 생성과 메시지 암호화 함수를 사용한다. 메시지 생성은 앞에서 살펴본 메시지 유형에 맞추어 JSON 형식으로 만들어지며 메시지 암호화의 경우 먼저 암호화를 위해 상대방의 공개키를 가져오는 과정이 필요하기 때문에 Pool Resolver 클래스의 객체가 필요하다. Pool Resolver

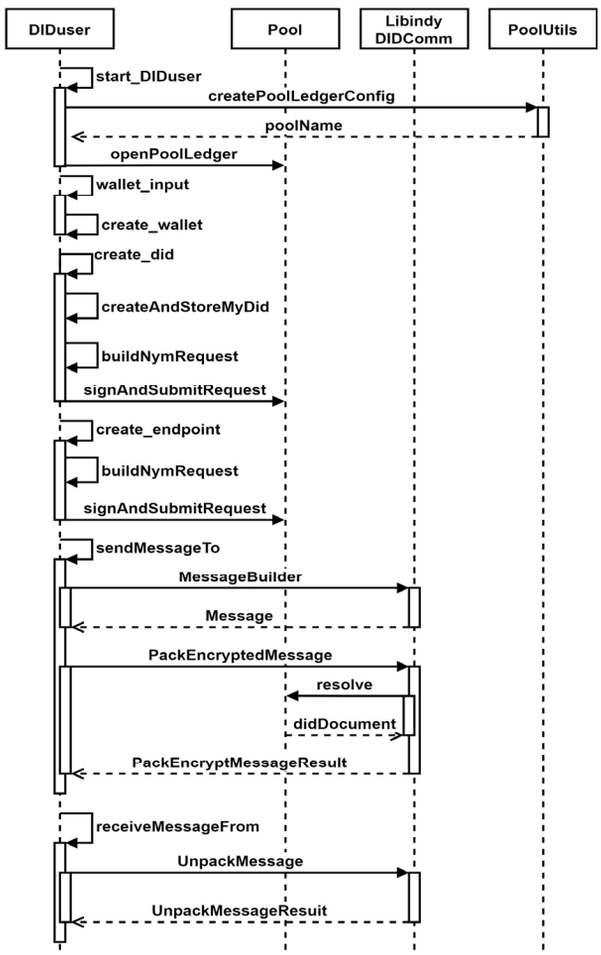


Fig. 6. Sequence Diagram for DIDComm Message Exchange

클래스의 resolve 메소드를 통해 상대방의 분산 식별자에 해당하는 DID 문서를 취득한다. DID 문서 또한 JSON 형식으로 만들어져 있기 때문에 공개키에 해당하는 속성을 찾아 값을 가져올 수 있다. 또한 상대방과의 통신을 위한 서비스 종단점 정보 또한 필요하므로 DID 문서에서 해당 속성을 통해 값을 가져온다. 공개키와 서비스 종단점 정보를 취득하면 통신을 위한 상대방 정보를 모두 얻은 것이다. 블록체인을 통해 해당 분산 식별자에 대한 메타 데이터를 가져올 수 없는 경우 예외를 발생시키며, 해당 상황은 입력한 분산 식별자가 올바르지 못하거나 분산 식별자가 폐기된 경우 발생한다.

다음으로 메시지 암호화를 진행한다. 여기서 사용한 메시지 암호화의 경우 libindy에서 자체적으로 제공 중인 메시지 암호화 API를 사용하여 메시지를 암호화한다. 해당 API는 packMessage라는 API로 사용자의 지갑과 상대방의 공개키, 사용자의 공개키, 메시지의 byte 값을 입력하는 것으로 메시지가 만들어진다. 여기서 사용자의 공개키 값 유무에 따라 서명의 유무가 결정된다. 서명에 사용자의 공개키가 사용되는 이유는 libindy에서 개인키의 직접적인 접근을 불가능하도록 하였기 때문이다. libindy의 경우 개인키를 사용해야 할 때, 개인키를 만들 때 같이 만들어진 공개키와 개인키가

저장되어 있는 지갑의 정보를 통해 개인키를 사용하기 때문에 사용자의 공개키 정보가 필요하다. 다음과 같은 과정을 통해 암호화 메시지가 만들어지게 되며 해당 메시지를 서비스 종단점을 통해 상대방에 전송하는 것으로 메시지 전송이 이루어진다.

메시지 복호화의 경우 앞에서 만들어진 메시지를 본인의 개인키로 복호화하면 메시지를 확인할 수 있다. 해당 부분 또한 libindy에서 제공해주는 API를 사용하여 복호화 하였으며, 해당 API는 unpackMessage이다. 위에서 언급한 내용과 같이 개인키의 직접적인 접근이 불가능하기 때문에 지갑 정보와 본인의 공개키 정보를 통해 접근할 수 있으며, 해당 정보와 메시지를 입력 값으로 함수를 호출할 경우 메시지의 복호화가 이루어진다. 복호화를 진행할 때 다른 분산 식별자의 공개키로 메시지가 암호화 되어 있거나 받은 메시지 형식이 올바른 구문이 아닐 경우 예외를 발생시킨다.

3.2 테스트 결과

구현한 DIDComm 기능의 테스트는 구현한 클래스들을 사용하여 진행하였다. 앞의 시퀀스 다이어그램은 암호화 메시지를 사용할 때 진행되는 시퀀스이며, 해당 테스트에서는 암호화 메시지, 서명이 들어간 암호화 메시지, 일반 텍스트 메시지를 만들고 암호화된 메시지를 복호화하는 과정을 테스트하였다. 소요 시간의 경우 인터넷 상황과 컴퓨터 성능에 따라 달라질 수 있다.

Fig. 7은 메시지 전송을 위해 처음 생성하는 일반 텍스트 메시지이다. 메시지는 앞에서 다룬 메시지 유형의 모습을 가지고 있으며, 메시지 생성의 경우 0.115초의 시간이 소요되었다.

해당 메시지를 그대로 전송할 경우 암호화하지 않은 일반 텍스트 메시지로 전송이 된다. 만들어진 메시지를 상대방의 공개키로 암호화해야 하므로 통합 해석기를 통해 상대방의 분산 식별자로 DID 문서를 가져온다. 해당 과정의 경우 0.001초가 소요되었다. Fig. 8은 상대방의 분산 식별자를 통해 통합 해석기를 통해 얻은 DID 문서이다.

통합 해석기를 통해 얻은 DID 문서로부터 공개키 정보를 취득한 후 해당 공개키를 사용해 메시지를 암호화한다. Fig.


```
message :
{
        "id":"89cbe7c6-1432-4407-abd4-c86523e45f98",
        "typ":"application₩/didcomm-plain+json",
        "type":"json",
        "from":"BcSaCDpJS7yhNf833hiRZQ",
        "to":["BcSaCDpJS7yhNf833hiRZQ"],
        "body":
                {
                        "title":"body"
                }
}
```


Fig. 7. An Example of Plain Text Message


```
didDocument :
{
        "@context":["https://www.w3.org/ns/did/v1","https://w3id.org/security/suites/ed25519-2018/v1","https://w3id.org/security/suites/x25519-2019/v1"],
        "id":"https://localhost:8080/7GSXaruXL8f3wgE8tqMmUD",
        "verificationMethod":
                [{
                        "type":"Ed25519VerificationKey2018",
                        "id":"7GSXaruXL8f3wgE8tqMmUD#key-1",
                        "publicKeyBase58":"4R64R1vmv7RetyyT4rZ86yTq95bFt7c4ryaRuZAUzzDm"
                },{
                        "type":"X25519KeyAgreementKey2019",
                        "id":"7GSXaruXL8f3wgE8tqMmUD#key-agreement-1",
                        "publicKeyBase58":"AVYdMDkLPHV9dY9wKWmqo1Xba12ejotEr15CHFBDTVSJ"
                }],
        "authentication":["7GSXaruXL8f3wgE8tqMmUD#key-1"],
        "assertionMethod":["7GSXaruXL8f3wgE8tqMmUD#key-1"],
        "keyAgreement":["7GSXaruXL8f3wgE8tqMmUD#key-agreement-1"],
        "service":
                [{
                        "type":"test",
                        "serviceEndpoint":"http://192.168.10.102:8383/api/request/"
                }]
}
```


Fig. 8. DID Document Generated by Universal Resolver


```
packedMsg1 :
{
        "ciphertext":"jqgy45tnmxa_aRt0FlOLX1SRti2eyo4q2Am7CrMY216sAeoKxPZMPO-NK0Q4CkuxfdZD4LTFHEW3i6oo9O_wsnbYrBl9S85kL3lD8Aftp_99k5xotqJjAXdyfoBx56YN28_2fhotFYFhdif8DETRhxn4x7vCmn3d8-hUXToYYYjv6wFwoUxCBkY9DljgcKHONLMqgKjvRrpJ4nVjtq5wePUGMP7q6uP0e795g2utmPpbC4rYxbG4pl2SOw==",
        "protected":"eyJlbmMiOiJ4Y2hhY2hhMjBwb2x5MTMwNV9pZXRmIiwidHlwIjoiSldNLzEuMCIsImFsZyI6IkFub25jcnlwdCIsInJlY2lwaWVudHMiOlt7ImVuY3J5cHRlZF9rZXkiOiJ2ZjZKTGZZaS04Q1c0MFV0eTk5enM5Y2hYLU52OU43U3dBZW91Q0JqYWlvM0NyYU9MT25aS1ZRZ0k1eEhUdHBlZ3EtWVZWQmZwWlBEUm5iU2pPR1RJMk1hS2hJU09VNmQwNThBQTlpM0VVOD0iLCJoZWFkZXIiOnsia2lkIjoiOE5CR0NBRGJKVTY2UXRZVVJ1MVJwQlZURzRaVkJQa2FuSnlIZndpcU1lYTUifX1dfQ==",
        "tag":"ZkVkNrd9RRcO1DJk-pTVog==",
        "iv":"dLsiOekvg1WEC_4s"
}
```


Fig. 9. The Encrypted Message by Using Public-Key

9는 공개키를 사용하여 암호화된 메시지이다. 해당 과정은 0.325초의 시간이 소요되었다.

메시지 전송을 위해 상대방의 DID 문서에서 얻은 상대방의 서비스 종단점으로 메시지를 전송한다. 암호화된 메시지를 수신한 상대방은 본인의 개인키를 사용하여 메시지를 복호화한다. Fig. 10은 복호화한 메시지의 모습이다. 해당 과정의 경우 0.342초의 시간이 소요되었다.


```
unpackMsgJson1 :
{
        "recipient_verkey":"8NBGCADbJU66QtYURu1RpBVTG4ZVBPkanJyHfwiqMHa5",
        "message":"
                {
                        ₩"id₩":₩"89cbe7c6-1432-4407-abd4-c86523e45f98₩",
                        ₩"typ₩":₩"application₩₩/didcomm-plain+json₩",
                        ₩"type₩":₩"json₩",
                        ₩"from₩":₩"BcSaCDpJS7yhNf833hiRZQ₩",
                        ₩"to₩":[₩"BcSaCDpJS7yhNf833hiRZQ₩"],
                        ₩"body₩":
                                {
                                        ₩"title₩":₩"body₩"
                                }
                }
"}
```


Fig. 10. The Decrypted Message by Using Private-Key

```
packedMsg :
{
        "ciphertext":"S4FysdZX__UXCEzQR0-YtcPre_Rodw7b1AAMRpru-41MqKzUDfixNT-5DbW6--W7GOLb2CFho_Sn3egpsabOPG6aIuoQVNVfXRG9WimOtI4vY3dB0_mFuC7KmaV-uGenxA2ltxx-TIZ3pSLItItIo68XujhPiM9WLdHTFX7a2h79xgL1eU7Jgc1sY9JsDrapXV_KIZa7gXnyakKXdiHmrAUQ0CrXVKv6jnGP8R1zi6RB4Xr--rTxbBv9Og==",
        "protected":"eyJlbmMiOiJ4Y2hhY2hhMjBwb2x5MTMwNV9pZXRmIiwidHlwIjoiSIdNLzEuMCIsImFsZyI6IkF1dGhjcnlwdCIsInJlY2lwaWVudHMiOlt7ImVuY3J5cHRlZF9rZXkiOiJNUDh2MUdxR0J4eGxyRnhZN0s1dzlQMmVqNy1hYk4zY1VMaTI2NDN1TjJxWkRyWEdBdmViTXN0MmdBeWlsUmNFIiwiaGVhZGVyIjp7ImtpZCI6IjhOQkdDQURiSlU2NlF0WVVSdTFScEJWVEc0WlZCUGthbkp5SGZ3aXFNSGE1IiwiaXYiOiJOa1k3akNWR3VZcFhnR1h3YzQ1VFBHUUM5SEYzVkJPbiIsInNlbmRlciI6IndiUWFyMk9acE1CNFlUMzdmNldndkdhTmwtaUw4NzIJaTdHSGVWa0tuM0FiRDZWNGlNTGpoS1BfcjV2UHlPOXFtNEZqczdqRktyTU9iV1lUUXJXbVpaeFRhQm94RXl1R2JJUGhCVlhjaUllaVJxcVpyQ0ZBeEl4VGVCYz0ifX1dfQ==",
        "tag":"iLup4F4KVPC1HZDqv_xwRQ==",
        "iv":"VcoVMm5cT3JwGTTe"
}
```


Fig. 11. The Encrypted Message with Signature


```
unpackMsgJson :
{
        "recipient_verkey":"8NBGCADbJU66QtYURu1RpBVTG4ZVBPkanJyHfwiqMHa5",
        "sender_verkey":"6nSJQnx28KAVUvSCE3BDQmk5UUA92mSAiM6tyvFxzsbh",
        "message":"
                {
                        ₩"id₩":₩"89cbe7c6-1432-4407-abd4-c86523e45f98₩",
                        ₩"typ₩":₩"application₩₩/didcomm-plain+json₩",
                        ₩"type₩":₩"json₩",
                        ₩"from₩":₩"BcSaCDpJS7yhNf833hiRZQ₩",
                        ₩"to₩":[₩"BcSaCDpJS7yhNf833hiRZQ₩"],
                        ₩"body₩":
                                {
                                        ₩"title₩":₩"body₩"
                                }
                }
"}
```


Fig. 12. Integrity-verified Message

Fig. 11은 메시지의 무결성을 확인하기 위해 본인의 서명을 추가한 메시지로 메시지에 본인의 개인키로 서명을 한 후, 상대방의 공개키로 암호화한 메시지이다.

Fig. 12는 위에서 만들어진 메시지를 수신하여 본인의 개인키로 복호화한 후 상대방의 공개키를 사용해 메시지 무결성까지 확인을 마친 메시지이다.

위의 일련의 과정 중, 서명을 추가한 암호화 메시지를 처리하는데 약 0.368초가 소모되었다. 서명이 없는 암호화 메시지의 경우와 시간 소요의 차이가 크게 차이 나지 않는다. 또한 해당 과정의 경우 복호화 부분을 포함해서 진행한 부분이며 실제로 사용할 경우 통신 환경에 따라 블록체인에서 DID 문서를 가져오는 부분과 메시지를 전송하고 받는 부분에서 추가적인 시간 소모가 일어날 수 있다.

4. 기존 방식과의 차이점

기존 방식은 중앙 서버를 통해 서로 연결에 필요한 정보를 주고받기 때문에 보안적인 통신을 위해 중앙 서버와 인증 서버가 요구된다. 이는 앞에서 언급한 바와 같이 중앙 서버의 해킹으로 인한 개인 정보 유출과 개인 정보 무단 사용 등의 문제점을 가지고 있다.

DIDComm을 이용한 통신은 중앙 서버를 필요로 하지 않는다. 이는 중앙 서버의 유무가 통신에 영향을 미치지 않는다. 통신 참여자의 소프트웨어 에이전트가 메시지 암호화 및 복호화 기능을 가지며 분산 식별자를 기반으로 비대칭 키 기반의 보안이 확보된 통신이 가능한 환경이라면 어디서든 상대방과 신뢰할 수 있는 통신이 가능하다. 이렇듯 DIDComm을 사용한 통신은 중앙 서버의 개입이 없는 탈중앙화 통신 구조로 기존의 중앙 서버를 통하지 않고 통신의 신뢰 문제를 해결하였다.

그러나 이러한 통신 방법은 개인에 의해 연결되기 때문에 그만큼 개인의 책임이 커진다. 중앙 서버를 통한 통신의 경우 중앙 서버에서 사용자를 관리하는 것으로 사기 등의 문제를 어느 정도 제어하는 것이 가능하며, 대화 기록 등을 저장해 두는 것으로 대화 내용을 복구하는 등의 업무를 수행할 수 있다. 그러나 중앙 서버의 개입이 없는 사용자간의 피어-투-피어 통신의 경우 통신 연결에서 문제가 생길 경우 문제를 파악하고 해결하는 것에 어려움이 있을 것이며, 사기 등의 피해를 미연에 방지하는 것이 어렵다.

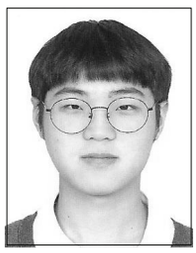
이러한 문제는 개인 간의 통신에서 신뢰를 위해 정보 교환을 추가하거나, 중간에 중개자를 두는 것으로 보완이 가능하다. 예를 들어 누군가 분산 식별자를 통한 연결을 요청했을 때 요청을 받은 사람은 요청자에게 추가적인 데이터를 요구하는 메시지를 보내어 상대방이 신뢰할 인물인지를 검증할 수 있다. 또는 검증 가능한 자격 증명이나 프레젠테이션을 요구하는 것으로 상대방이 신뢰 가능한 인물인지 검증이 가능하다. 특히 분산 식별자를 사용한 자격 증명을 사용할 경우 중개자의 개입 없이 본인을 증명할 수 있고 DIDComm 자체가 분산 식별자를 사용하기 때문에 오히려 활용도가 높다.

5. 결 론

본 글은 DIDComm 기술을 분석하여 동작 과정과 메시지 유형을 확인하고 기존 방식과의 차이점 및 테스트 결과를 설명하였다. 분산 식별자를 사용한 자기주권 신원 시스템은 개인의 정보를 본인이 관리할 수 있게 해주는 기술로 향후 등장할 가상-물리 생태계에서 본인의 정보를 보다 안전하게 사용할 수 있도록 할 것이며 해킹으로 인한 피해를 줄일 것이다. 그리고 이러한 분산 식별자 기술을 활용한 DIDComm은 앞으로 발전할 인터넷 세계에 강력한 보안을 가진 통신을 제공해 줄 것이다. 그리고 이러한 기술은 떠오르고 있는 메타버스 시대에서 주목할 만한 통신 방법이다. 중앙의 서버를 통하지 않기 때문에 서로 DIDComm을 사용한 메신저만 있다면 통신이 가능해지며, 이를 사용하여 개인적인 대화나 거래에도 사용할 수 있다. 하지만 DIDComm을 사용한 통신은 그만큼 중앙 서버의 개입이 적어지므로 개인의 책임이 커지고 만약에 거래를 통해 피해를 볼 경우, 그에 대한 본인의 책임 또한 매우 커질 것이다. 이를 보완하기 위해 개인 간 신용을 위한 정보 교환을 투명하게 하는 블록체인과 같은 중개자를 추가할 수도 있다.

References

[1] Y. H. Kim, “A study on the application of digital identification using blockchain,” Master dissertation. Kookmin University, Seoul, 2020.
[2] D. G. Yoon, “Self-Sovereign Identity Structure Analysis,” 1st ed., Seoul: Jpub, 2020.
[3] A. Preukschat and D. Reed, “Self-Sovereign Identity,” 1st ed., New York: Manning Publications Co., 2021.
[4] M. Sporny, D. Longley, M. Sabadello, D. Reed, O. Steele, and C. Allen, “Decentralized Identifiers (DIDs) v1.0 Core architecture, data model, and representations,” W3C PR. Aug. 2021.
[5] K. H.-Duffy, R. Grant, and A. Gropper, “Use Cases and Requirements for Decentralized identifiers,” W3C WG Note, Mar. 2021.
[6] M. Sabadello, K. D. Hartog, C. Lundkvist, C. Franz, A. Elias, A. Hughes, J. Jordan, and D. Zagidulin, Introduction to DID Auth [Internet], https://url.kir/xar7qu.
[7] M. Sabadello and D. Zagidulin, “Decentralized Identifier Resolution (DID Resolution) v0.2,” W3C Draft CG Report Aug. 2021.
[8] DIF, Universal Resolver [Internet], https://url.kr/xa3oqj.
[9] S. Curren, T. Looker, O. Terbu, “DIDComm Messaging,” DIF Editor’s Draft, 2022. 1. [Internet], https://url.kr/i7b1t9.
[10] Hyperledger Indy-sdk Repository [Internet], https://url.kr/l4y726.
[11] SICPA DLab Didcomm-jvm Repository [Internet], https://url.kr/vhg3i9.



최 규 현

https://orcid.org/0000-0002-5862-222X
e-mail : giry8647@gmail.com
2016년 ~ 현 재 동의대학교 게임애니메이션공학전공 학부생
관심분야 : 블록체인, 게임 프로그래밍

김 근 형

https://orcid.org/0000-0002-7691-5608
e-mail : geunkim@deu.ac.kr
1986년 서강대학교 전자공학과(공학사)
1988년 서강대학교 전자공학과(공학석사)
2005년 포항공과대학교 컴퓨터공학과 (공학박사)
2007년 ~ 현 재 동의대학교 게임공학전공 교수
관심분야 : 탈 중앙 웹, Web 3.0, 인공지능, 설명가능 인공지능, 블록체인, 자기주권 데이터