



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

공학석사 학위논문

하이퍼레저 패브릭을 활용한  
선원승하선관리시스템

Embarkation and Disembarkation Management System for Sailors  
Using Hyperledger Fabric Platform



지도교수 이 장 세

2020년 8월

한국해양대학교 대학원

컴퓨터공학과

이정욱

본 논문을 이정욱의 공학석사 학위논문으로 인준함

위원장 : 박 휴 찬



위 원 : 김 재 훈



위 원 : 이 장 세



2020년 7월 16일

한국해양대학교 대학원

# 목 차

<b>List of Tables</b> .....	iii
<b>List of Figures</b> .....	iv
<b>Abstract</b> .....	vi
<b>제 1 장 서론</b> .....	<b>1</b>
<b>제 2 장 관련 연구</b> .....	<b>4</b>
2.1 블록체인 기술 .....	4
2.1.1 블록체인 개요 .....	4
2.1.2 블록체인 프레임워크 종류 .....	8
2.1.3 스마트계약의 개요 및 한계 .....	11
2.2 하이퍼레저 기술 .....	11
2.2.1 하이퍼레저 프레임워크 개요 .....	12
2.2.2 하이퍼레저 패브릭 아키텍처 .....	14
2.2.3 하이퍼레저 패브릭 체인코드 .....	15
2.2.4 하이퍼레저 패브릭의 트랜잭션 흐름 및 작동원리 .....	17
2.3 승하선공인 업무개요 .....	21
<b>제 3 장 승하선관리시스템 설계</b> .....	<b>24</b>
3.1 승하선관리시스템 정의 .....	24
3.2 네트워크 구성요소 및 설계 .....	26
3.3 체인코드 설계 .....	28
3.4 미들웨어 설계 .....	31

제 4 장 승하선관리시스템 개발 및 검증 .....	32
4.1 네트워크 구현 및 실행 .....	33
4.2 체인코드 및 미들웨어 구현 .....	36
4.3 승하선관리시스템 검증 .....	41
제 5 장 결 론 .....	45
참고문헌 .....	46



## List of Tables

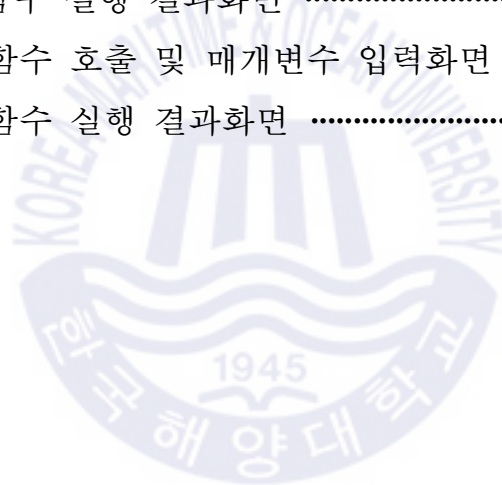
Table 2.1 퍼블릭 블록체인과 허가형 블록체인의 특징 .....	10
Table 2.2 기존의 블록체인과 하이퍼레저 패브릭의 특징 .....	12
Table 2.3 하이퍼레저 프레임워크 및 도구 .....	13
Table 3.1 승하선관리시스템의 데이터 .....	24
Table 3.2 승하선관리시스템 기능을 구현한 체인코드 함수 .....	30
Table 4.1 시스템 구현환경 .....	32



## List of Figures

Fig 2.1	탈중앙화를 위한 블록체인 개념 .....	5
Fig 2.2	블록 구조와 블록 헤더 .....	6
Fig 2.3	머클 트리의 구조와 머클 루트 .....	7
Fig 2.4	블록의 내부 구조 .....	8
Fig 2.5	하이퍼레저 패브릭 아키텍처의 구성 .....	15
Fig 2.6	단일 프로세스에서 체인코드 처리과정 .....	16
Fig 2.7	하이퍼레저 패브릭 트랜잭션 흐름 .....	17
Fig 2.8	하이퍼레저 패브릭의 Gossip프로토콜 동작의 예 .....	19
Fig 2.9	Fabric-Certificate Authority 구조 .....	20
Fig 2.10	승하선공인신청서 양식 .....	21
Fig 2.11	선박 및 육상사무소에 비치되는 서류에 관한 선원법 .....	22
Fig 2.12	선원수첩에 관한 선원법 .....	23
Fig 3.1	승하선관리시스템의 구성도 .....	25
Fig 3.2	하이퍼레저 패브릭 네트워크 기본 구성도 .....	27
Fig 3.3	하이퍼레저 패브릭 기반 승하선관리시스템 네트워크 구성 .....	28
Fig 3.4	승하선관리시스템에서 요구되는 기능 .....	29
Fig 3.5	API 체인코드 호출 시 내부 동작 .....	31
Fig 4.1	하이퍼레저 패브릭 네트워크 조직 구현 .....	33
Fig 4.2	하이퍼레저 패브릭 네트워크 피어 환경변수 설정 .....	34
Fig 4.3	우분투 환경에서 네트워크 실행화면 .....	35
Fig 4.4	네트워크 생성 시 조직 및 정렬자의 하부 디렉터리 .....	36
Fig 4.5	승하선관리시스템의 데이터 구조체 .....	36

Fig 4.6 승하선관리시스템의 함수목록 .....	37
Fig 4.7 InitLedger 함수 구현 .....	37
Fig 4.8 관리자 등록 기능을 위한 미들웨어 구현 .....	38
Fig 4.9 사용자 등록 기능을 위한 미들웨어 구현 .....	39
Fig 4.10 Query기능을 위한 미들웨어 구현 .....	39
Fig 4.11 Invoke기능을 위한 미들웨어 구현 .....	40
Fig 4.12 우분투 환경에서 관리자 및 사용자 등록 실행화면 .....	41
Fig 4.13 체인코드 설치 및 인스턴스화 실행화면 .....	42
Fig 4.14 QueryAll함수 실행 결과화면 .....	42
Fig 4.15 QuerySailor함수 실행 결과화면 .....	43
Fig 4.16 CreateInfo함수 실행 결과화면 .....	43
Fig 4.17 DisEmDate함수 호출 및 매개변수 입력화면 .....	44
Fig 4.18 DisEmDate함수 실행 결과화면 .....	44





# 하이퍼레저 패브릭을 활용한 선원승하선관리시스템

이 정 욱

한국해양대학교 대학원  
컴퓨터공학과

## 초록

현재 선원법 시행규칙에 따르면 선장 또는 선박소유자는 선원의 승하선 시 선박과 육상사무소에 비치해 둔 선원명부를 해양항만관청에 공인을 받아야 한다. 하지만 선원명부를 직접 해양항만관청에 가져가서 공인해야하기 때문에 그 과정이 비효율적이고 분실의 위험이 있다. 또한, 선박과 육상사무소에 문서를 비치함에 따라 개인정보노출의 위험이 있으며 관리의 불편함이 존재하고 해양항만청이라는 중앙 집중적 데이터베이스에 의존함에 따라 단일 지점으로서의 해킹 위험성이 있다. 본 논문에서는 승하선공인과정에서 신뢰성을 제공하고 자동화할 수 있는 시스템 개발을 목표로 한다. 기존의 중앙 집중적인 시스템의 문제점인 개인정보노출과 신뢰성 침해 문제를 해결하기 위해 허가형 블록체인(Permissioned blockchain)인 하이퍼레저를 사용하여 데이터의 투명성과 무결성을 제공한다. 특히 하이퍼레저는 기존의 블록체인보다 발전된 형태로 처리속도와 데이터 저장의 한계를 개선하였으며 퍼블릭 블록체인과는 달리 비즈니스관계자들만

참여해 보안성과 비즈니스 효율성을 강화하여 비즈니스 프로젝트에 적합하다. 승하선관리시스템을 통해 선원명부 및 선원수첩과 같은 승하선공인 시 필요한 서류를 디지털화하여 분실의 위험을 제거한다. 또한, 블록체인 기술을 통하여 신뢰성을 제공하고, 효율적이고 안전한 관리가 가능하며 기존의 승하선처리과정을 보다 효율적으로 처리할 수 있다.

키워드 : 블록체인, 승하선공인, 하이퍼레저 패브릭



# Embarkation and Disembarkation Management System for Sailors Using Hyperledger Fabric Platform

Lee, Jeongwook

Department of Computer Engineering,  
Graduate School of  
Korea Maritime and Ocean University

## Abstract

According to the Seamen Law, a captain or ship owner is required to obtain official approval from the Korea Maritime Port Authority for the list of crew members kept in ships and land offices when embarking and disembarking ships. However, the process is inefficient and there is a risk of loss because the list of sailors must be taken directly to the Maritime Port Authority for approval. In addition, there is a risk of personal information exposure and inconvenience in management as documents are kept in ships and land offices, and there is a risk of hacking as a single point because the Maritime Port Authority is a centralized database.

The objectives of this paper are to develop a system for automating process of official approval for embarkation and disembarkation and providing reliability through it. To do this, we apply Hyperledger fabric.

Hyperledger, a permissioned blockchain, was used to provide transparency and integrity of data to solve the problem of privacy exposure and reliability infringement, which are problems of the existing centralized system. Hyperledger is a more advanced than the existing blockchain, improving the limits of processing speed and data storage, and unlike public blockchains, it is suitable for business projects by enhancing security and business efficiency by involving only business stakeholders.

We expect that the proposed system will eliminate the risk of loss by digitalizing the documents necessary for the authorization of the ship. In addition, the blockchain technology provides reliability, enables efficient and safe management, and enables more efficient handling of existing embarking and disembarking processes.

**Key Words:** Embarking and Disembarking official approval, Blockchain, Hyperledger Fabric,

## 제 1 장 서 론

해양수산부에서는 선원법에 따라 “승선공인을 받지 않은 선원에 대해서는 선원의 직무를 수행해서는 안된다”고 규정하고 있다. 따라서 유효한 면허가 있다고 하더라도 승하선공인을 받지 않고 직무를 수행하는 경우 무자격 선원에 해당되어 문제가 발생한다. 과거 선원법에 따르면 승하선 공인과정에서 선원의 승하선이 이루어진 지역을 관할하는 지방해양수산청에서만 신청이 가능해 선박 소유자의 소재지와 선원의 승하선 지역이 떨어져 있을 경우 불편함을 겪었으나 2019년 5월 전국 지방해양수산청 어디에서나 승하선공인을 신청할 수 있도록 선원법 시행규칙을 개정하게 되었다[1].

하지만 여전히 그 과정에 있어 선원수첩등의 분실의 위험이 있고 비효율적이며 선박 및 육상사무소에 서류로 보관해야하는 불편함과 보관 중에 서류의 분실의 가능성이 존재한다. 또한 승하선 공인 시 필요한 선원수첩에는 중요한 개인정보가 포함되어 있어 개인정보노출의 위험이 존재하며 공인시스템의 중앙집중적 구조 때문에 단일지점으로써의 해킹 위험이 존재한다. 국제해사기구인 IMO(International Maritime Organization) 제 6차 인적요소 결과보고서에 따르면 선원의 전자 증명서와 서류의 용이한 사용을 위하여 전자 증명서 사용지침을 승인하였다[2]. 중국 및 러시아 등 일부 회원국에서는 전자문서의 활용에 대한 필요성은 인식하였으나 데이터 위변조, 전자문서복제 등의 정보 유출 위험을 언급하였다.

데이터 위변조를 해소하기 위한 연구의 하나로, 4차 산업혁명의 핵심기술인 블록체인이 제안되었다[3]. 블록체인이란, 분산되어 있는 사용자들 간의 합의에 의해 블록에 데이터를 생성하고 동기화하는 분산 원장기술이

다. 블록체인을 통해 중앙화된 시스템에서 탈중앙화의 목표를 달성하고 네트워크에 참여하고자 하는 사용자는 모두 참여하여 분산원장을 소유하게 된다. 블록체인은 생성한 데이터를 네트워크 내에서 공유하여 투명성을 보장하고 정보를 보호한다[4]. 기존의 블록체인은 무결성을 제공하지만 익명의 다수가 네트워크에 참여하기 때문에 개인정보 노출과 비즈니스 거버넌스에 적합하지 않아 한계점을 가진다. 하이퍼레저는 허가형 블록체인으로서 허가된 참여자만 네트워크에 참여하여 제 3자에 정보노출위험이 없으며 이를 통해 비즈니스 거버넌스에 맞게 자동화된 시스템을 가동할 수 있다.

본 논문에서는 기존의 선원 승하선공인신청과정의 문제점을 극복하기 위해 하이퍼레저 기반 승하선관리시스템을 제안한다. 승하선관리시스템은 기존의 승하선공인신청과정을 대신하여 하이퍼레저 기반 자동화된 프로그램을 통해 선박과 육상사무소에서 승하선하는 선원들의 문서를 지방해양청에 직접 방문하지 않고 제출할 수 있으며 공인신청서를 따로 작성할 필요가 없다. 또한, 승하선공인신청을 위한 문서를 선박과 육상사무소에 비치할 필요가 없어 서류 보관이 용이하고 개인정보가 포함된 선원수첩의 분실 및 개인정보 노출의 위험이 제거된다. 승하선관리시스템은 블록체인의 탈중앙화형태로 중앙 서버의 해킹으로 인한 피해가 없으며 블록체인 네트워크 내에 허가된 조직만 참여하여 승하선 공인 신청하는 선박과 육상사무소에서 선원의 정보 및 공인일을 작성하여 기존의 공인신청을 대신하고 지방해양청과 해양수산부에서 이를 처리한다.

본 논문의 구성은 다음과 같다. 2장에서는 하이퍼레저 기반 승하선관리 시스템 개발과 관련된 기술을 소개하고 현재 선원법에 따른 승하선공인과정을 기술한다. 3장에서는 승하선관리시스템의 네트워크를 구성하여 하이퍼레저 패브릭의 네트워크로 구현하고 자동화 프로그램인 체인코드 및 사용자가 어플리케이션에서 체인코드를 사용할 수 있도록 미들웨어를 설계한다. 4장에서는 설계된 체인코드를 구현하고 네트워크에 배포하여 자동화된 시스템을 구축하고 설계된 미들웨어를 구현함으로써 사용자가 작성

된 체인코드를 통해 시스템을 동작할 수 있도록 한다. 승하선과정을 작성된 체인코드로 수행하여 시스템을 검증하고 이후 5장에서 결론으로 마무리한다.



## 제 2 장 관련 연구

### 2.1 블록체인 기술

#### 2.1.1 블록체인 개요

지속적으로 증가하는 정보산업에서의 데이터 위변조문제와 비용문제를 해결하고 투명성과 신뢰성을 제공하기 위해 블록체인(Blockchain)기술이 제안되었다. 블록체인 시스템은 데이터와 시스템의 무결성을 확보하고 유지하기 위해 네트워크 내에서 일어난 트랜잭션을 블록에 기록하여 연결하고 암호화한 분산 P2P(Peer to Peer) 시스템을 의미한다[3,4]. 블록체인 네트워크 내에서 블록은 정해진 시간마다 발생되고 트랜잭션(Transaction)이 발생하면 블록에 이를 순서대로 나열한다. 발생한 트랜잭션은 한 번만 기록되며 어떤 사용자도 트랜잭션을 변경하거나 위조할 수 없다. 네트워크에 참여한 모든 사용자들은 각각 원장을 분산 소유하고 새로운 트랜잭션이 발생하면 블록에 기록하고 블록을 생성하는 작업은 공동으로 수행한다. 네트워크 참여자들에 의해 검증된 트랜잭션은 일관성 있게 네트워크 인프라에 공유됨으로써 신뢰성이 제공되고 분산 원장은 공유되지만 참여자의 개인정보는 암호화를 통해 데이터노출에 대한 적절한 보호가 이루어진다.

그림 2.1은 블록체인의 개념을 나타낸다. 그림 2.1(a)는 기존의 중앙 집중형 시스템으로서 개인은 신뢰할 수 있는 제 3자 기관의 신뢰에 크게 의존하여 단일지점으로서의 해킹 및 보안 위험이 존재한다. 따라서 중앙 시스템의 보안 위험 및 오류를 예방하기 위해 인프라와 보안부분에 대규모 투자로 인해 시스템 관리비용이 높다는 단점이 존재한다. 그림 2.1(b)는



탈중앙화 형태로 대표적으로 블록체인이 있다. 블록체인은 제 3자 기관이 만들어 놓은 비즈니스 신뢰를 시스템 구조적 합의 알고리즘을 통해 다양하게 구현할 수 있다. 모든 정보가 중앙에 있는 중앙 집중형 시스템과 달리 분산된 네트워크 내에서 모든 참여자가 트랜잭션을 검증하고 유지 관리 하며 각 주체들은 제 3자 기관의 개입 없이 합의 알고리즘을 통해 트랜잭션을 검증한다. 검증된 트랜잭션을 대조해 무결성을 확보하여 트랜잭션 정보의 투명성 확보와 시스템 구축 및 유지보수 비용 감소의 장점이 있다.

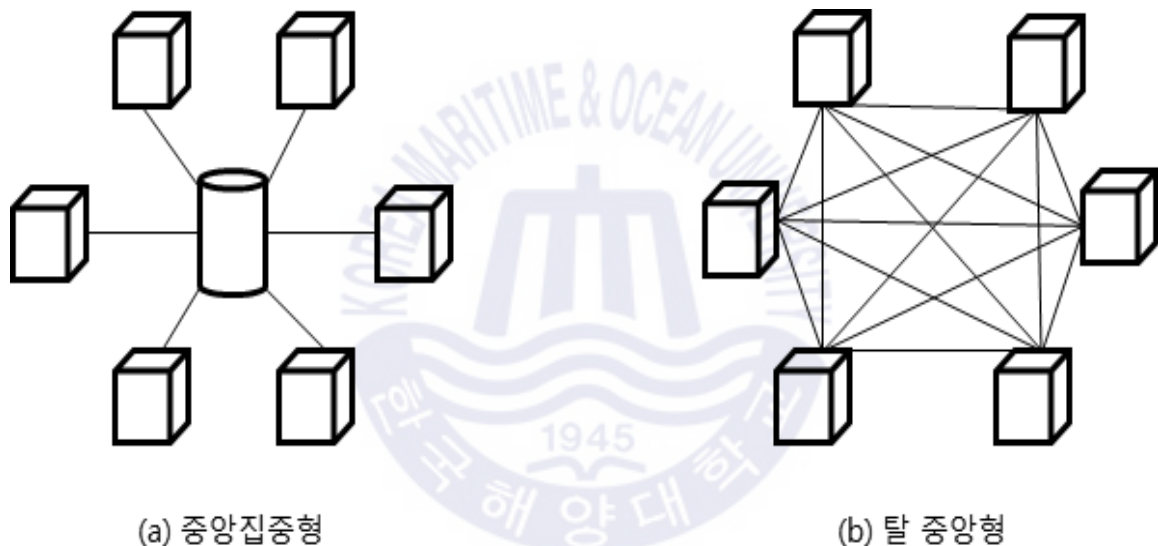


Fig. 2.1 탈중앙화를 위한 블록체인 개념

그림 2.2(a)는 블록체인의 블록 구조를 보여주고 있다. 블록은 블록의 크기를 나타내는 블록 크기 필드, 블록 헤더 필드, 트랜잭션 개수 필드, 트랜잭션을 저장하는 트랜잭션 필드로 구성되어 있다[3]. 블록의 전체의 크기는 가변적이지만 트랜잭션 크기가 1 Mbyte를 넘길 수 없어 1 Mbyte가 넘을 경우 다음 블록에 트랜잭션을 담아야 한다. 블록의 바디인 트랜잭션 필드에는 해당하는 트랜잭션 내용이 저장되고 트랜잭션 내용은 네트워크 참여자 누구나 생성할 수 있고 블록체인이 연결되기 전까지 주변들



으로 데이터 무결성과 트랜잭션을 인증 및 검증하고 방대한 양의 정보를 요약해서 작은 단위로 저장하기 위해 필수적이다. 구해진 해시 값은 블록의 헤더에 있는 머클 루트 필드에 저장되고 머클 루트를 통해 트랜잭션에 대한 무결성을 검사하고, 블록 헤더에 저장되어 해당 블록의 트랜잭션에 대한 유효성을 검증한다.

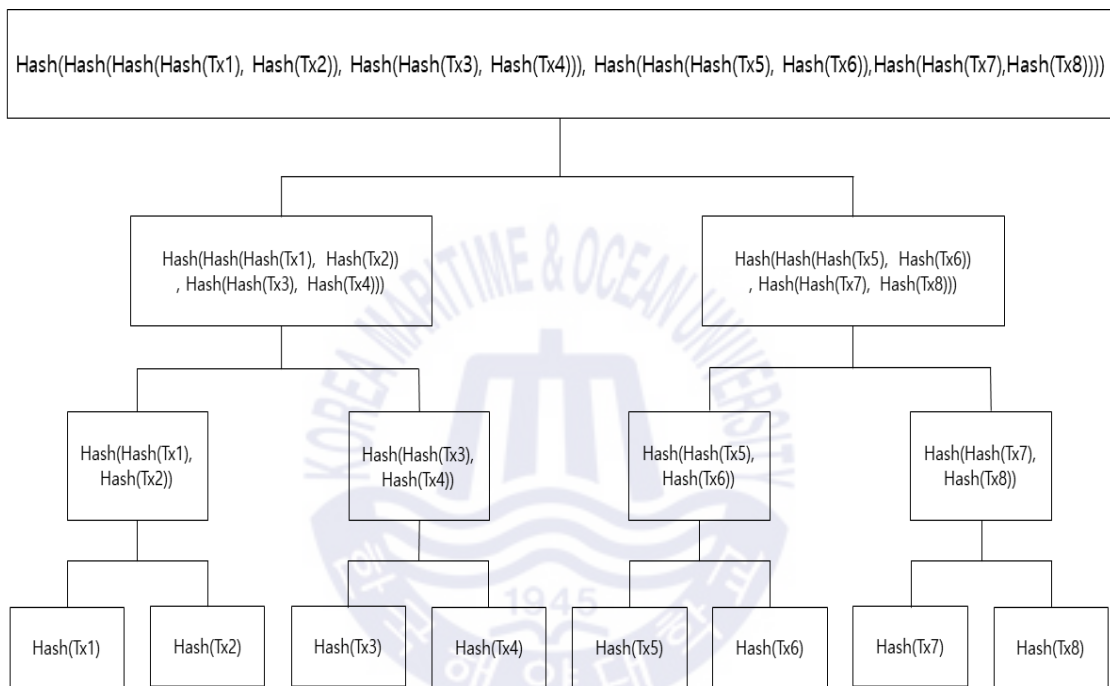


Fig. 2.3 머클 트리의 구조와 머클 루트

그림 2.4는 블록체인의 내부 연결 구조를 나타낸다. 블록체인에서 블록은 블록의 고유한 값인 해시값을 갖게 된다. 현재 블록의 해시값은 다음 블록의 헤더에 있는 이전의 블록의 해시 필드에 저장되고 그 값을 참조하여 블록은 연결되고 트랜잭션이 조작되는 경우 블록체인 내의 모든 트랜잭션과 헤더 간의 연결은 끊어진다. 블록의 해시값은 블록의 데이터를 연속해서 두 번 해시한 후 사용되는데 사용되는 해시 함수는 SHA-256로 입력에 관계없이 항상 256 bit의 출력을 생성한다.

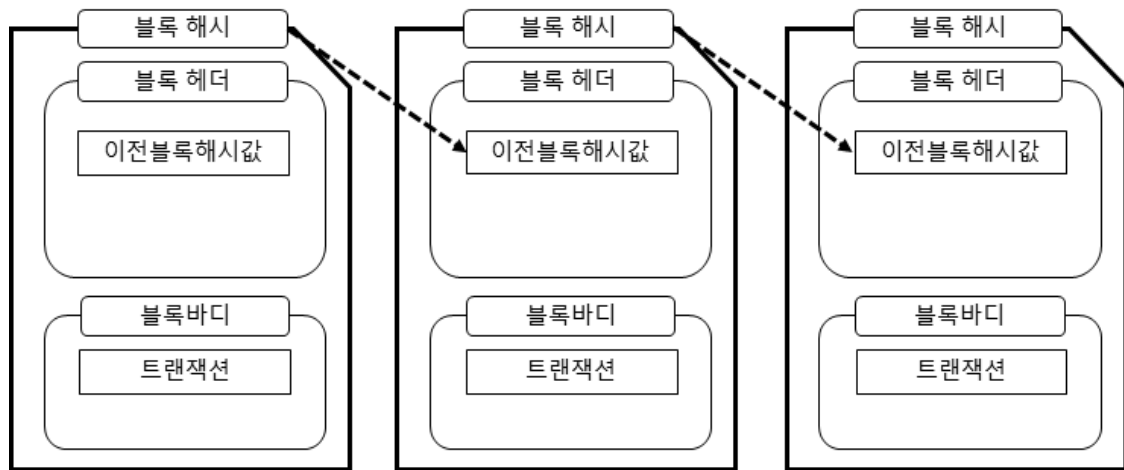


Fig. 2.4 블록의 내부 구조

### 2.1.2 블록체인 프레임워크 종류

블록체인은 비즈니스 목적에 따라 크게 퍼블릭 블록체인(Public blockchain)과 허가형 블록체인(Permissioned blockchain)으로 크게 나뉜다 [10,19]. 퍼블릭 블록체인과 허가형 블록체인의 특징을 비교하여 정리하면 표 2.1과 같다.

#### (1) 퍼블릭 블록체인

퍼블릭 블록체인은 완전히 분산된 형태로 참여자의 제한이 없고 권한 없이 누구나 네트워크에 참여하여 사용할 수 있는 블록체인으로 현재 가장 일반적으로 사용되는 블록체인이다[20].

퍼블릭 블록체인을 사용하기 위해서는 권한 없는 사용자 누구나 네트워크에 참여하여 컴퓨터를 통해 채굴해야하며 불특정 다수의 참여자에 의해 운용되기 때문에 채굴자들에게 보상이 필요하고 보상인 암호화폐에 가치가 있어야 한다. 또, 참여자들은 작업증명 POW(Proof-Of-Work)이나 지분 증명 POS(Proof-Of-Stake)을 통하여 트랜잭션의 정당성을 입증하여야 한다[9]. 비즈니스 목적으로 퍼블릭 블록체인을 활용 시 낮은 트랜잭션 처리량과 긴 지연시간으로 비효율적이며 가용성이 저하된다. 실제 이더리움에

서는 초당 20개의 트랜잭션이 수행되며 10~20초의 지연이 발생한다[5,18].

제 3자가 원장을 볼 수 있기 때문에 개인정보 및 기밀성 보호의 부재가 발생한다. 허가되지 않은 참여자들도 블록체인 네트워크에 참여할 수 있기 때문에 권위 있는 조직의 일관된 정책과 전략으로 비즈니스를 수행할 수 없고 이러한 환경은 블록체인 거버넌스의 부재로 비즈니스의 효율성과 제약성을 저하시킨다. 또한 퍼블릭 블록체인은 파워와 전력 등 대규모의 에너지가 소모되기 때문에 비즈니스 요구사항을 모두 만족할 수 있는 비즈니스를 수행할 수 없다.

## (2) 허가형 블록체인

프라이빗 블록체인 중 허가된 참여자들만 네트워크에 참여하는 허가형 블록체인은 퍼블릭 블록체인의 단점인 기밀성, 효율성, 가용성의 한계를 해결하고 네트워크 참여자들 간의 신뢰할 수 있는 환경을 제공한다.

허가형 블록체인은 네트워크 비즈니스 목적에 부합하는 서로 신뢰할 수 있는 참여자들 간의 합의를 통해 디지털 상호작용을 하고 주체가 분명한 구성원들로 이루어져 있기 때문에 새로운 네트워크를 넓혀 나갈 필요가 없으며 블록체인 영역에서의 보상이 필요 없다. 비즈니스 요구사항에 기밀성, 효율성, 가용성을 준수하여 조직은 상황에 맞는 합의 알고리즘을 사용하여 트랜잭션의 처리속도를 극대화하고 이해관계자들에 한해서만 데이터가 공개되어 데이터에 대한 프라이버시도 지켜진다. 또 지나친 에너지를 사용하지 않아도 되기 때문에 시스템의 무결성을 유지하면서 효율적인 블록검증이 가능하다. 블록체인의 비즈니스 목적과 정책이 조직의 목표와 연계되어 거버넌스를 수행할 수 있게 하고 비즈니스 요구사항을 만족할 수 있는 네트워크가 운영되어지도록 지시하고 감독할 수 있어 조직의 상황에 따라 유연하게 적용할 수 있다[11,12].

허가형 블록체인은 기존의 퍼블릭 블록체인과는 다른 트랜잭션 처리 흐름을 가진다. 허가형 블록체인은 코드를 먼저 실행하고 주문을 처리하기 때문에 퍼블릭 블록체인에 비해 계약을 빠르게 실행하며 모듈식 아키텍처

를 지원하여 합의 알고리즘과 신원관리를 증명할 수 있다. 퍼블릭 체인의 경우 모든 참여자가 블록을 공유하므로 기밀성이 침해되지만 허가형 블록 체인은 허가된 참여자들만 데이터에 접근할 수 있어 기밀성이 제공된다 [9,16].

Table 2.1 퍼블릭 블록체인과 허가형 블록체인의 특징

구분	퍼블릭 블록체인	허가형 블록체인
주체	· 모든 참여자	· 허가된 참여자
권한	· 모든 참여자가 모든 권한 보유	· 권한 지정 가능
트랜잭션속도	· 느리다	· 빠르다
확장성	· 어렵다	· 쉽다
데이터 접근	· 모든 참여자가 가능	· 허가 된 참여자만 가능
증명 및 검증방법	· POW나 POS에 의해 결정	· 합의 알고리즘으로 결정
트랜잭션 처리순서	· 주문 - 실행	· 실행 - 주문
컨트랙트 실행순서	· 순차적으로 실행	· 병렬로 실행하여 처리속도가 빠름
모듈성	· 제공하지 않음	· 합의 알고리즘 · 신원증명 관리
기밀성	· 제공하지 않음	· 제공
활용 예	· 비트코인 · 이더리움	· 하이퍼레저

### 2.1.3 스마트계약의 개요 및 한계

블록체인에서 비트코인 시스템은 금융거래에 접목되어 비트코인이라는 가상통화의 가치이전을 목적으로 사용하였다면 이더리움은 이를 업그레이드 한 것으로 금융거래에 한정적이던 기존의 블록체인 시스템을 모든 분야로 확장시켰고 블록체인 상에 프로그램을 넣어 자동으로 실행시키는 구조로 발전하였는데 이를 스마트계약이라 한다[8]. 스마트계약 자체는 블록체인 이전부터 보급되었으나 주목 받지 못했고 블록체인 기술이 등장함에 따라 기술이 정비되고 활용할 수 있는 분야가 넓어졌다. 스마트계약이라는 단어는 계약인 것처럼 보이기 때문에 혼란을 줄 수 있지만 반드시 계약이 아니라 업무를 자동적으로 수행하기 위한 프로그램 또는 코드이다. 스마트계약의 목적은 솔리디티 또는 Go언어로 계약 조항을 구현하여 노드에 배포함으로써 제 3자 기관에 의존하지 않고 계약이 자동적으로 실행되어 비용을 절감하고 블록체인기술과 결합되어 안정성과 신뢰성을 제공하는 것이다. 하지만 스마트계약은 프로그램 또는 코드가 모든 실제 사회의 계약을 해석할 수 없다는 한계를 가지고 있다. 프로그래밍 언어로 작성할 수 있는 경우에만 사용할 수 있고 프로그램이 자체적으로 판단하는 인공지능 분야에서는 실현할 수 없다. 자동적으로 실행하는 코드를 구현했다 하더라도 코드에 있는 에러에 의해 법적문제가 발생할 수 있기 때문에 코드와 계약 사이에 차이가 있는 경우에 어떻게 해결할 것인가에 대한 규정을 계약서에 설정해 두는 것이 필요하다.

## 2.2 하이퍼레저 기술

블록체인 기술이 무결성을 유지할 수 있는 이유는 익명의 네트워크 참여자 모두에 의한 검증이 기반이 되어있기 때문이다. 하지만 익명의 참여자들에게 모든 정보가 노출되기 때문에 개인정보 등 기밀성에 위배된다. 그러므로 기존의 블록체인 기술인 이더리움을 통한 스마트계약은 한계를 가지게 된다[11]. 이 때문에 특정 비즈니스에서의 효용성이 떨어지고 확장

성을 유지하기 힘들다. 이러한 한계를 극복하기 위해 인증 기반의 블록체인인 하이퍼레저를 통해 신원확인을 가능하게 하고 정보 보호의 문제를 해결할 수 있다. 하이퍼레저는 인증 서버를 갖추어 참여자들의 신분을 확인하고 허가받은 참여자만 스마트계약이 사용 가능하도록 한다. 표 2.2는 기존의 블록체인과 하이퍼레저 패브릭의 특징이다. 기존의 블록체인은 네트워크를 신뢰하지 못하기 때문에 익명의 참여자 모두에게 검증을 받아야 한다. 하지만 하이퍼레저 패브릭은 MSP(Membership Service Provider)에 의해 모든 네트워크 참여자를 인증하고 통제하고 있으며 네트워크에 참여한 모두를 신뢰하고 있다.

Table 2.2 기존의 블록체인과 하이퍼레저 패브릭의 특징

블록체인	하이퍼레저 패브릭
익명기반	인증기반
익명의 모두에게 검증	인증된 피어들에 반복된 검증
리더 랜덤 선출	정렬자가 최종기록 허가
비동기 시스템	정렬자에 의해 동기화

### 2.2.1 하이퍼레저 프레임워크 개요

하이퍼레저는 개방형 기술 개발을 지원하는 리눅스 재단 중심의 오픈소스 프로젝트로 여러 분야의 블록체인 기술을 발전시키기 위해 제안되었다 [6,7]. 참여하는 개발자들은 오픈소스로 배포되는 코드에 대한 권리를 갖고 있으며 이를 사용하거나 발전시키고 재배포할 수 있다. 오픈소스 프로젝트의 이점 중 하나는 유연성이며 프로젝트에 참여한 조직들은 개방형 표준 기술을 사용하여 서로 다른 태스크를 통합하고 추가 기능을 사용자 정의에 따라 쉽게 추가할 수 있다. 따라서 오픈소스 프로젝트인 하이퍼레



저는 저렴한 비용으로 기능을 제공하고 서비스를 발전시키고 비즈니스 요구사항을 쉽게 충족할 수 있다. 표 2.3과 같이 하이퍼레저 프레임워크는 합의 알고리즘과 구현 형태에 따라 5가지 프레임워크와 리눅스 재단에서 제공하는 5가지 도구로 나누어진다. 모바일 개발 프로젝트용인 이로하(Iroha)는 하이퍼레저 패브릭을 기반으로 많은 기업들이 참여하여 BFT 합의 알고리즘을 사용한다[14]. 소투스(Sawtooth)는 인텔에서 참여하였으며 새로운 합의 알고리즘인 PoET(Proof of Elapsed Time)을 포함하여 효율적인 분산 합의 달성을 목표로 한다. 하이퍼레저 버로(Burrow)는 인텔이 개발에 참여해 이더리움 가상머신(Ethereum Virtual Machine)을 구현한 모듈형 블록체인이다. 소브린(Sovrin)재단에서 참여한 하이퍼레저 인디(Indy)는 원장에 대한 신원 인증 시스템을 지원하기 위해 만들어진 프로젝트로 분산 ID를 제공하기 위한 라이브러리 및 도구를 제공한다. 마지막으로 하이퍼레저 패브릭(Fabric)은 초기에 IBM 주도 하에 개발되던 코드였는데 현재 여러 다른 기업이 함께 참여하여 매우 활성화 되었다. 하이퍼레저 패브릭 네트워크는 개방된 네트워크 형태가 아니라 MSP라 불리는 인증 관리 시스템에 등록된 사용자만 참여할 수 있어 프라이빗 또는 컨소시엄 블록체인이라 불린다. 이와 함께 개발을 위한 5가지 툴이 제공되는데 컴포저(Composer), 칼리퍼(Caliper), 익스플로러(Explorer), 퀼트(Quilt), 첼로(Cello)등이 개발의 편의성을 위해 제공되고 있다.

**Table 2.3** 하이퍼레저 프레임워크 및 도구

하이퍼레저 프레임워크	하이퍼레저 도구
이로하	컴포저
소투스	칼리퍼
버로	익스플로러
인디	퀼트
패브릭	첼로

## 2.2.2 하이퍼레저 패브릭 아키텍처

하이퍼레저 패브릭은 하이퍼레저 프로젝트 중 가장 활발하게 사용 중인 오픈소스 프로젝트로서 모듈러 아키텍처를 통한 어플리케이션 개발이 가능한 프레임워크이다. 하이퍼레저 패브릭은 비즈니스 목적에 맞게 설계된 허가형 블록체인의 형태를 가지며 누구나 참여가 가능한 기존의 블록체인 네트워크와는 달리 인증 관리 시스템에 허가된 참여자만이 네트워크에 참여할 수 있는 허가형 네트워크를 전제로 크게 신원, 블록체인, 트랜잭션, 체인코드로 구성된다[13].

그림 2.5는 하이퍼레저 패브릭 아키텍처를 나타낸다. 사용자는 하이퍼레저 패브릭 네트워크에 참여하기 위해 신원확인이 되어야하고 사용자 정보를 관리하는 MSP를 통해 확인된다. MSP는 본질적으로 권한을 부여하는 모듈로 구성원의 신원을 관리하기 위해 사용자 등록, 참여자 인증서를 발행하는 역할을 한다. 네트워크에 참여한 신뢰가 있는 노드들은 이미 허가되었기 때문에 악의적인 해킹 또는 사용을 검증하기 위한 복잡한 알고리즘을 필요로 하지 않고 원장에 접근하려는 사용자의 권한만 검증함으로써 네트워크에 참여할 수 있다. 블록체인의 원장은 블록에 트랜잭션 정보가 실제로 저장되는 공간으로서 분산원장 데이터베이스 시스템을 제공하며 이를 처리 및 관리하기 위해 일정한 프로토콜이 사용된다. 블록체인 서비스에서 블록체인 내에서 생성되는 트랜잭션을 관리하고 참여자들의 의견 일치를 통해 트랜잭션 정렬 및 트랜잭션 승인, 검증하여 원장에 갱신된다. 하이퍼레저 패브릭에서 체인코드는 스마트계약과 동일한 기능으로 허가형 블록체인에서 비즈니스 및 컨소시엄 서비스에 맞게 블록체인을 활용할 수 있는 로직으로 개발가능하게 한다. 체인코드는 이해 관계자에 의해 합의된 비즈니스 로직에 따라 특정 트랜잭션을 블록에 삽입 및 조회하기 위해 사용된다. 또한, 이더리움과 달리 코인을 주고받는 처리가 없는 하이퍼레저 패브릭에서의 개발자는 네트워크 참여자들 간의 합의된 로직을 통해 프로그램을 구현하고 요구사항을 만족시킨다.

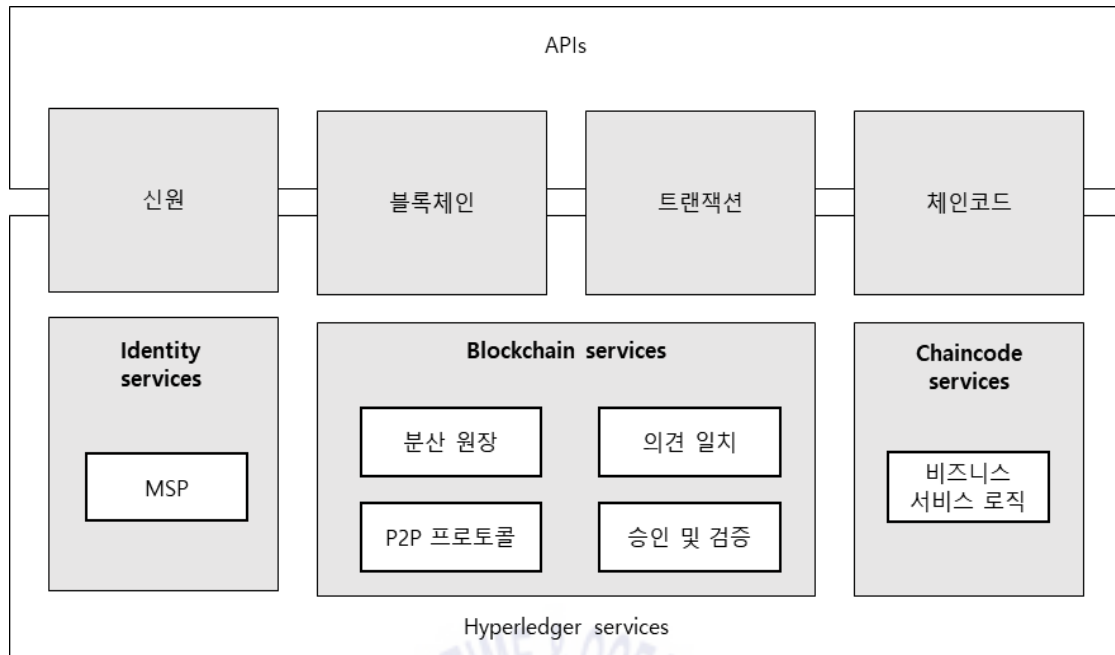


Fig. 2.5 하이퍼레저 패브릭 아키텍처의 구성

### 2.2.3 하이퍼레저 패브릭 체인코드

체인 코드는 생성(Creation), 배포(Deploy), 트랜잭션 작성(Invoke), 조회(Query)의 4가지 과정으로 처리하게 된다. 다음은 체인코드 처리과정의 순서이다.

- 1) 생성과정 : 체인코드를 작성하고 컴파일을 하여 실행파일을 만든다.
- 2) 배포과정 : 정렬자에게 채널 생성을 요청하고 피어를 채널에 참여 시킨 후 생성된 체인코드를 피어에게 배포한다.
- 3) 트랜잭션 작성과정 : 체인코드를 실행하고 피어로부터 응답이 오면 정렬자에게 트랜잭션을 제출한다. 정렬자서비스를 통해 생성된 블록은 채널의 모든 피어에게 전달되고 블록을 전달 받은 피어는 체인에 블록을 등록한다.
- 4) 조회과정 : 트랜잭션을 제안하여 체인코드를 실행하고 실행된 결과를 반환한다.

배포과정에서 체인코드는 소스코드에 따라 네트워크에 체인코드를 등록하고 Invoke함수를 통해 트랜잭션을 작성하고 블록을 생성하여 원장(Ledger)을 갱신한다. 갱신된 원장의 트랜잭션을 Query함수를 통해 조회는 할 수 있지만 갱신하지 않으므로 블록을 생성하지 않는다. 체인코드가 사용되기 위해서는 먼저 피어 노드에 체인코드가 설치되어야한다. 체인코드를 배포하기 전 개발모드 단계에서는 체인코드는 단일 피어에 연결되어 동작하게 되는데 실제 코드가 동작하는지 여부를 확인만 하는 단계이기 때문에 단일 프로세스로서 동작한다. 그림 2.6은 단일프로세스에서 동작하는 체인코드를 보여준다. 피어와 연결된 응용프로그램에서 단순한 Query 트랜잭션이 제안되면 피어는 원장으로부터 받은 결과를 응용프로그램에게 보내게 된다. 응용프로그램이 제안한 트랜잭션이 Invoke인 경우 연관된 피어들에게 제안을 하고 체인코드를 실행하고 합의알고리즘을 통하여 합의된 트랜잭션은 갱신된다.

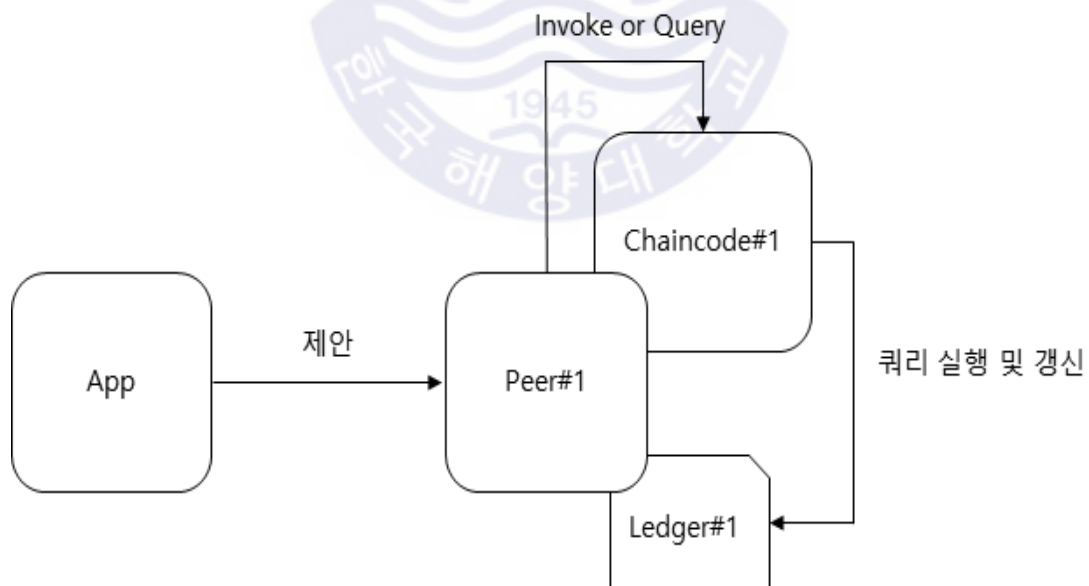


Fig. 2.6 단일 프로세스에서 체인코드 처리과정

## 2.2.4 하이퍼레저 패브릭의 트랜잭션 흐름 및 작동원리

### (1) 하이퍼레저 패브릭의 트랜잭션 흐름

그림 2.7은 하이퍼레저 패브릭의 트랜잭션 흐름을 보여준다. 하이퍼레저 패브릭의 작동 과정은 크게 4가지의 작동 흐름으로 운용되는데 API나 서비스는 피어들로부터 전자서명을 전달받아 무결성을 확보하고, 정렬자에게 전달함으로써 최종적으로 확인을 거친 후 피어들이 분산원장을 갱신하는 과정으로 볼 수 있다. 사용자, 피어, 정렬자 그룹은 모두 MSP에서 신원을 확인 받으며 관리되고 외부 참여는 없다[10].

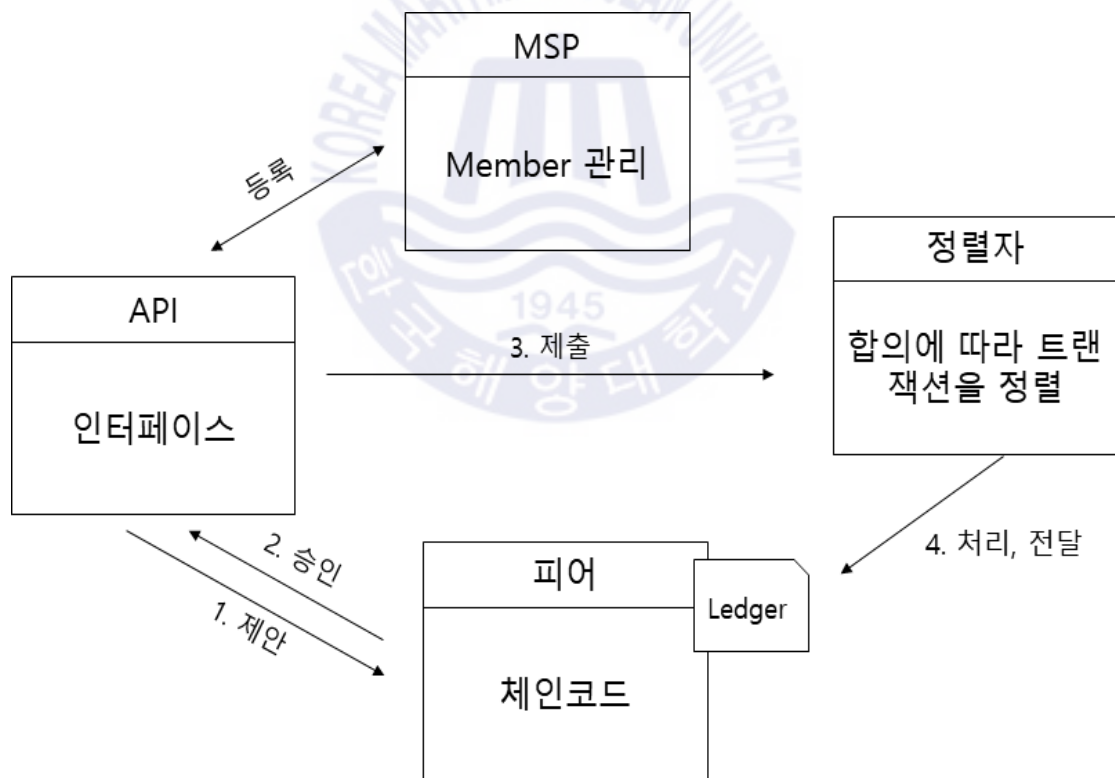


Fig. 2.7 하이퍼레저 패브릭 트랜잭션 흐름

구체적인 작동 과정은 다음과 같다.

1) API나 서비스는 트랜잭션을 실행하기 위해 제안하고 피어 그룹에 요청 하여 모든 피어에게 전달된다.

2) 각 피어는 자신의 로컬에서 체인코드 함수를 동시 실행하여 트랜잭션을 수행하고 그 결과에 전자서명하여 완성하고 모든 피어에게 무결성이 합의된 트랜잭션을 API에게 되돌려 준다. 이 과정은 스마트계약에서 모든 노드가 코드를 자신의 로컬에서 수행하면서 검증하는 절차와 동일하다. 하지만 이더리움에서는 검증을 마친 즉시 동의 절차를 수행하면서 원장에 저장했지만 하이퍼레저 패브릭에서는 API에게 되돌려 준 후 정렬자에게 보내는 과정을 거친다.

3) API는 정렬자에게 트랜잭션을 전달하여 처리를 완료해 줄 것을 요청한다.

4) 기존의 퍼블릭 블록체인에는 없었던 정렬자는 기존의 승인된 트랜잭션을 합의 알고리즘에 따라 순서대로 정렬하고 일괄 처리하여 조직에 리더 역할을 하는 피어에게 블록 생성 제안을 한다. 최종적으로 전달 받은 피어는 원장에 기록하기 전에 유효성 검사를 하고 원장에 기록한다.

## (2) 하이퍼레저 패브릭의 작동원리

### 1) 하이퍼레저 패브릭 프로토콜

하이퍼레저 패브릭에서 노드끼리 네트워크를 구성하고 네트워킹할 때 제약사항이 따르는데 이때 사용하는 프로토콜이 Gossip 프로토콜이다[21]. Gossip 프로토콜은 한 노드에서 전체 노드에 원장 업데이트기록을 알리지 않아도 전체가 알게 되는 것을 말한다. 그림 2.8은 Gossip 프로토콜의 동작을 나타낸다. Gossip 프로토콜은 전체에 말하지 않아도 되는 편리함이 존재하고 한 노드에 업데이트 및 변경기록이 도착하는 순서가 정해져있지 않다. 한 경로가 분리되어도 다른 경로를 통해서 전달받을 수 있어 피어

간의 동기를 맞추는데 용이하다. 하지만 거짓정보가 전달될 수도 있어 하이퍼레저 패브릭은 이러한 단점을 인증을 통해서 원천봉쇄한다.

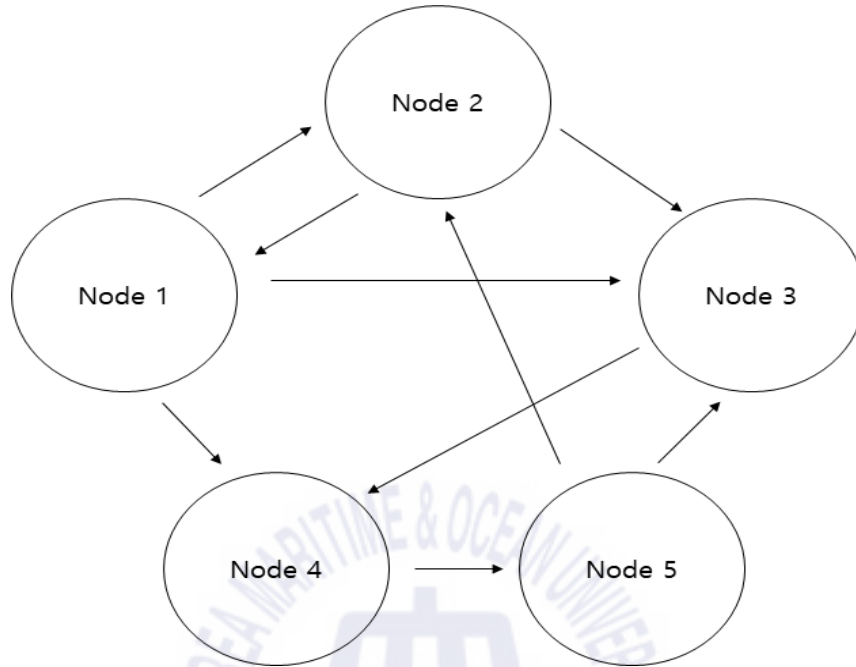


Fig. 2.8 하이퍼레저 패브릭의 Gossip 프로토콜 동작의 예

## 2) 하이퍼레저 패브릭 인증방법

하이퍼레저 패브릭은 허가된 네트워크로 네트워크 참여자들을 증명할 방법이 필요하다. 하이퍼레저 패브릭CA(Certificate Authority)는 PKI(Public Key Infrastructure)를 제공한다[16,17]. PKI는 서명 검증을 위해 공개키를 서버에 등록하고 등록된 공개키를 통해 사용자 인증 및 트랜잭션 인증을 위한 서명 검증에 활용한다. 기존의 블록체인은 P2P방식으로 트랜잭션을 수행하기 때문에 PKI가 필요하지 않지만 하이퍼레저 패브릭은 자체적으로 CA서버를 제공하고 관리자 및 운영자는 네트워크에 참가한 사용자 및 클라이언트를 사전에 CA에 등록하여 사용자를 인증한다[6].

그림 2.9는 패브릭의 CA 구조를 나타낸다. 하이퍼레저 패브릭에서 CA는 사용자등록, 호출된 트랜잭션 및 사용자 또는 구성요소 간의 보안 연

결을 제공한다. 사용자를 증명하기 위해 CA에서 개인키와 공개키 쌍을 생성하여 ID를 발급하고 발급된 ID의 목록을 MSP가 관리한다. Enrollment CA는 새로운 사용자를 네트워크에 등록하는데 사용되고 이를 통해 각각의 멤버, 사용자에게 등록에 사용되는 발행 인증서 ecert가 생성된다. 하이퍼레저 패브릭은 네트워크에 참여한 사용자들을 위해 등록인증서인 ecert를 발행하는데 ecert는 사용자의 신원을 증명하기 위해 사용되며, 사용자들이 트랜잭션을 제출할 때 서명 역할을 한다. 따라서 네트워크 사용자들은 트랜잭션을 작성하기 전, Fabric-CA에 등록하여 ecert를 얻어야 한다. Transaction CA를 통해 등록된 사용자가 트랜잭션을 요청할 때 필요한 인증서 tcert가 최종 생성된다. TLS CA는 사용자들이 원격피어와 통신할 때 보안 채널로서 활용된다.

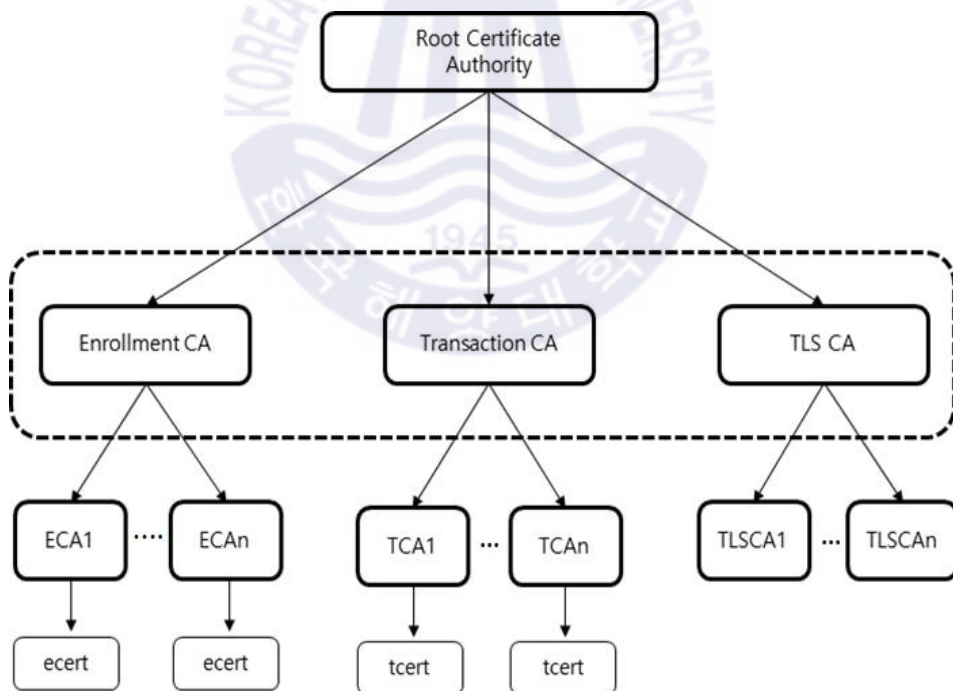


Fig. 2.9 Fabric-Certificate Authority 구조



### 2.3 승하선공인 업무개요

승하선 공인이란 선원의 승선, 하선에 대한 해양수산관청이 행하는 업무를 말한다[1]. 그림 2.10은 승하선 시 필요한 공인 신청서양식이다. 선원의 승, 하선이 있을 때마다 선박소유자 및 선장은 해양수산관청에 선원 명부와 선원수첩의 공인신청을 하도록 규정하고 있다. 공인신청서에 필요한 정보는 선원이 현재 승선신청을 하는지 하선신청을 하는지가 표기되어야 하고 현재 타고 있는 선원이 근무 중인 업체명, 선주의 국적, 선박의 명, 선박의 국적이 기입되어야 한다. 또한, 선박의 운항형태 등 필요한 선박의 용도 및 총 톤수가 기입되어야 하고 최종적으로 공인일의 기입이 요구된다.

[별지 제7호서식] <개정 2017. 7. 19.>

<input type="checkbox"/> 승선 <input type="checkbox"/> 하선 <input type="checkbox"/> 승선취소		공인 신청서 (계 □명)		처리기간
				즉시
①업체명		②선주국적	<input type="checkbox"/> <input type="checkbox"/>	
③선명		④선박국적	<input type="checkbox"/> <input type="checkbox"/>	
⑤운항형태	<input type="checkbox"/> 1. 외항상선 2. 내항상선 3. 연근해어선 4. 원양어선 5. 해외취업상선 6. 해외취업어선			
⑥선박용도	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	⑦선박번호 또는 호출부호		
⑧총톤수	G/T	⑨기관출력		kW
⑩항행구역	<input type="checkbox"/> 1. 평수 2. 연해 3. 근해 4. 원양			
⑪승선일 또는 종료(취소)일				⑫공인일

Fig. 2.10 승하선공인신청서 양식

그림 2.11은 현재 선원법에 규정된 선박 및 육상사무소에 비치해야 할 서류의 항목을 보여준다. 선원법 제 20조에 따라 선박 및 육상사무소에서 선박국적증서, 선원명부, 항해일지, 화물의 관한서류를 갖추고 있어야 한다. 선원법 제 44조에 따라 선원의 승하선 교대가 있을 때마다 선박 및 육상사무소에 비치하고 있는 선원명부를 작성하고 공인받아야 한다.

제20조(서류의 비치)

① 선장은 다음 각 호의 서류를 선내에 갖추어 두어야 한다. <개정 2013.3.23.>

1. 선박국적증서
2. 선원명부
3. 항해일지
4. 화물에 관한 서류
5. 그 밖에 해양수산부령으로 정하는 서류

제44조(선원명부의 공인)

① 선박소유자는 해양수산부령으로 정하는 바에 따라 선박별로 선원명부를 작성하여 선박과 육상사무소에 갖추어 두어야 한다. <개정 2013.3.23.>

② 선박소유자는 선원의 근로조건 또는 선박의 운항 형태에 따라서 해양수산부령으로 정하는 바에 따라 선원의 승선·하선 교대가 있을 때마다 선박에 갖추어 둔 선원명부에 그 사실과 승선 선원의 성명을 적어야 한다. 다만, 선박소유자가 선원명부에 교대 관련 사항을 적을 수 없을 때에는 선장이 선박소유자를 갈음하여 적어야 한다.

③ 선박소유자는 제2항에 따른 승선·하선 교대가 있을 때에는 선원 중 항해구역이 「선박안전법」 제8조제3항에 따라 정하여진 근해구역 안인 선박의 선원으로서 대통령령으로 정하는 사람을 제외한 선원의 선원명부에 대하여 해양항만관청의 공인(인터넷을 통한 공인을 포함한다. 이하 같다)을 받아야 한다. 이 경우 선박소유자는 선장에게 자신을 갈음하여 공인을 신청하게 할 수 있다. <개정 2015. 1. 6.>

Fig. 2.11 선박 및 육상사무소에 비치되는 서류에 관한 선원법

그림 2.12는 승하선 공인 시 제출해야하는 선원수첩에 관한 선원법이다. 선원법에 따르면 선원수첩이란 “선원이 되려는 사람은 대통령령으로 정하는 바에 따라 해양항만관청으로부터 발급” 받아야 하는 선원의 신분증명서로 승선하고 있는 선원은 제 2항에 따라 선장에게 제출하여 보관하게 하여야 하며 제 3항에 따라 선원명부의 공인 시 해양항만관청에 선원수첩을 제출해야 한다. 공인 신청 시 부득이한 사유로 선원수첩을 제출할 수 없을 경우 별도의 서류를 통하여 지방해양항만관청에 직접 제출해야하고 선원수첩을 분실하거나 파손되었을 경우 지방해양관청에 승하선공인의 증명을 신청해야 한다.

제45조(선원수첩)

① 선원이 되려는 사람은 대통령령으로 정하는 바에 따라 해양항만관청으로부터 선원수첩을 발급받아야 한다. 다만, 대통령령으로 정하는 선원의 경우에는 해양수산부령으로 정하는 바에 따라 선박소유자로부터 신원보증서를 받음으로써 선원수첩의 발급을 갈음할 수 있다. <개정 2013.3.23.>

② 선원은 승선하고 있는 동안에는 제1항에 따른 선원수첩이나 신원보증서를 선장에게 제출하여 선장이 보관하게 하여야 하고, 승선을 위하여 여행하거나 선박을 떠날 때에는 선원 자신이 지녀야 한다.

③ 선박소유자나 선장은 제44조제3항에 따라 선원명부의 공인을 받을 때에는 해양수산부령으로 정하는 바에 따라 승선하거나 하선하는 선원의 선원수첩이나 신원보증서를 선원명부와 함께 해양항만관청에 제출하여 선원수첩이나 신원보증서에 승선·하선 공인을 받아야 한다.

④ 제3항에도 불구하고 인터넷을 통하여 승선·하선 공인을 받은 경우 해양항만관청은 선원수첩이나 신원보증서에 대한 공인을 면제할 수 있다. <신설 2015. 1. 6.>

⑤ 해양수산부장관은 선원의 취업실태나 선원수첩 소지 여부를 파악하거나 그 밖에 필요하다고 인정하는 경우에는 선원수첩을 검사할 수 있다. <개정 2013. 3. 23., 2015. 1. 6.>

⑥ 선원수첩의 발급 절차 등에 필요한 사항은 대통령령으로 정한다. <개정 2015. 1. 6.>

Fig 2.12 선원수첩에 관한 선원법

## 제 3 장 승하선관리시스템 설계

### 3.1 승하선관리시스템 정의

현재 선원의 승하선 공인과정은 직접 해양항만관청에 선원명부와 선원수첩을 제출해야하므로 비효율적이며 급하게 신청해야할 경우 별 다른 방법이 없다. 또한 선원수첩에는 개인정보가 포함되어 서류로 보관 시 노출의 위험이 존재함과 동시에 선원명부 및 선원수첩의 분실 위험이 존재한다. 그러므로 본 논문에서는 하이퍼레저 패브릭 플랫폼을 사용하여 데이터 무결성을 제공하며 분실의 위험을 없애고 허가된 참여자들에게만 데이터를 제공해 개인정보를 제 3자에게 노출하지 않는 승하선관리 시스템을 개발한다. 승하선관리시스템에서는 선원의 이름과 공인 신청서의 선박명, 선박의 번호, 선박의 용도, 선박의 총 톤수 등 중복된 정보는 하나의 정보로 간주하고 선박 번호를 사용하고 선원의 신분을 증명하는 선원수첩 번호와 승선일, 하선일을 기록한다. 표 3.1과 같은 내용을 포함하여 트랜잭션으로 생성하고 블록에 기록함으로써 승하선공인과정 시 필요한 데이터를 효율적으로 관리한다.

Table 3.1 승하선관리시스템의 데이터

· 승하선 하는 선원의 이름
· 선원이 승선 또는 하선하는 배의 정보를 위한 선박 번호
· 선원의 신분 증명을 위한 발급된 선원수첩 번호
· 승선일
· 하선일

그림 3.1은 승하선공인시스템의 구성을 나타낸다. 조직1은 선원명부와 선원수첩을 보관하고 승하선공인을 신청하는 선박 및 육상사무소로 구성되고 조직 2는 승하선공인신청을 처리하는 기관인 해양수산부, 지방해양관청으로 구성된다. 조직 1과 조직 2에 속한 각 기관은 각 조직의 CA로부터 개인키, 공개키를 발급받아 허가된 사용자를 증명한다. 각 조직은 선원의 정보를 생성, 조회, 모든 선원의 정보를 조회할 수 있으며, 선원 정보 생성한 날을 선원의 승선한 날로 기록하며 선원의 하선 시 하선한 날짜를 기록할 수 있도록 하였다. 기존의 공인신청 과정의 불편함을 선원의 승, 하선일을 기록하고 블록에 등록함으로써 공인신청을 대신할 수 있다.

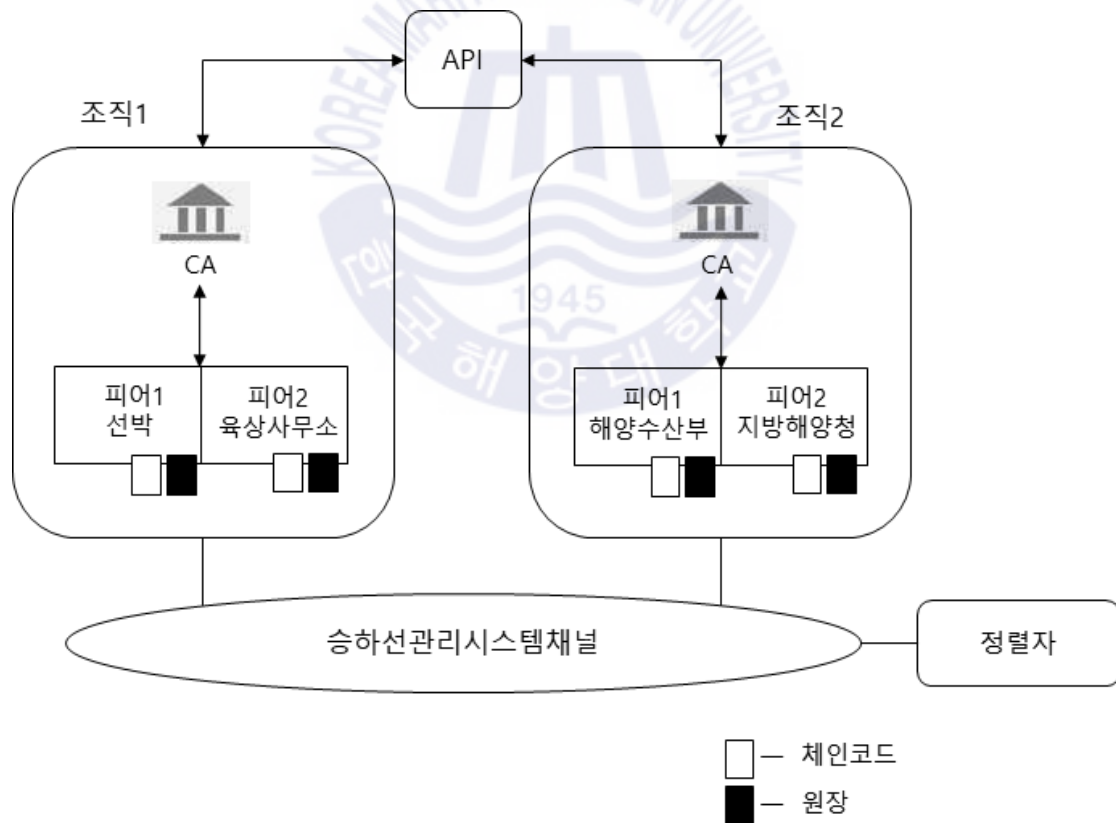


Fig. 3.1 승하선관리시스템의 구성도

## 3.2 네트워크 구성요소 및 설계

그림 3.2는 하이퍼레저 패브릭 네트워크 구성을 나타낸 것이다. 하이퍼레저 패브릭 네트워크는 어플리케이션과 채널로 구분하여 한 채널에는 하나의 장부만 존재하며 내부는 피어, 정렬자, MSP로 분류되고 클라이언트는 API나 SDK로 네트워크에 참여한다. 하이퍼레저 네트워크를 구성하는 각 네트워크의 기능 및 특징은 다음과 같다[15,19].

### (1) 채널

이해관계가 있는 피어들 사이에서 원장을 공유하기 위한 요소로서 이해관계에 따라 조직을 구분하여 채널을 생성할 수 있고 한 네트워크 내에는 하나 이상의 채널이 존재해야 한다. 채널에 참여한 조직의 구성원만 해당 채널의 분산원장에 접근할 수 있어 데이터 기밀성을 유지할 수 있다.

### (2) 피어(Peer)

피어는 하이퍼레저 네트워크에서 원장을 가진 노드로서 체인코드를 포함하고 있고 관리자를 통해 Committing peer, Endorsing peer, Leader peer, Anchor peer로 역할이 분배된다. 클라이언트는 블록체인에 접근하려는 사용자로서 트랜잭션을 생성하여 피어 노드의 Endorser에게 전송하고 피어노드 내의 Anchor를 통해 조직에 속한 다른 피어들을 검색하고 트랜잭션을 조직 내의 다른 피어에게 전송하는 역할을 한다. Endorser에서 트랜잭션이 타당한지를 검증 한 뒤 다시 클라이언트에게 트랜잭션 제안 응답을 보내게 된다. 검증받은 트랜잭션은 클라이언트에서 정렬자로 전송되고 정렬자에서 생성된 블록은 Leader peer에게 전달받게 되는데 Leader peer는 전달받은 블록을 조직 내의 다른 Peer에게 전달하는 역할을 한다. Committing peer는 최신 블록에 대한 검증작업을 하는 Peer로서 모든 피어는 Committing 역할을 수행해야한다.

### (3) 정렬자(Orderer)

정렬자는 트랜잭션을 시간 순서로 정렬하여 블록을 생성하는 요소로서 Endorsing peer로부터 검증받은 트랜잭션을 순서대로 정렬한 후 최신 블록을 생성하여 Leader peer로 보내게 된다.

### (4) MSP

네트워크에 참여하는 각 조직은 하나 이상의 피어를 관리하며 체인코드에 권한을 얻기 위해 피어 및 클라이언트의 인증서를 발급하는 MSP에 의존한다. MSP로 신원을 증명하고, 조직 내에서 관리자, 조직의 멤버 등 역할을 부여한다.

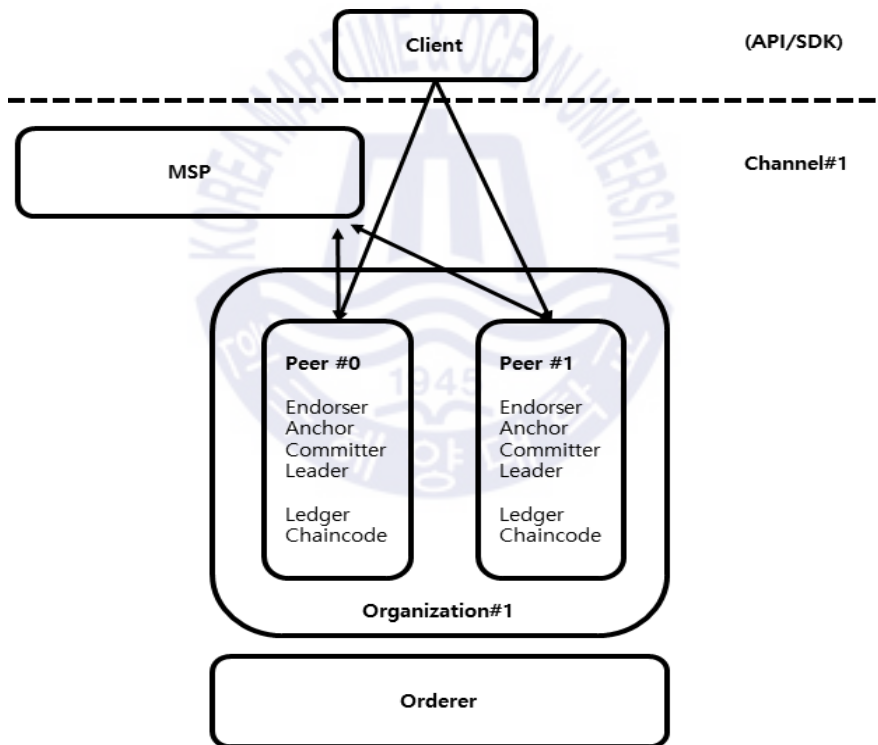


Fig. 3.2 하이퍼레저 패브릭 네트워크 기본 구성도

그림 3.3은 하이퍼레저 패브릭 기반 승하선관리시스템의 네트워크 구조이다. 신뢰관계를 가진 여러 노드를 한 조직으로 조직화하는 것은 보안 및 비용 측면에서 효율적이므로 승하선 공인시스템의 네트워크는 선박과

육상사무소를 한 조직(Org1)으로 구성하고 지방해양관청 및 해양수산부를 또 다른 한 조직(Org2)으로 구성한다. 또한 피어를 가동하는 것은 비용이 많이 들기 때문에 조직의 거버넌스에 맞게 최소화하여 조직을 구성하는 것이 효율적이다. 따라서 조직1에서는 선박 및 육상사무소가 피어0, 피어1을 담당하고 조직2에서는 해양수산부가 피어0, 지방해양관청이 피어1을 담당하고 각 조직에 대해 MSP와 CA가 따로 존재하여 조직의 사용자 및 피어를 관리하게 된다.

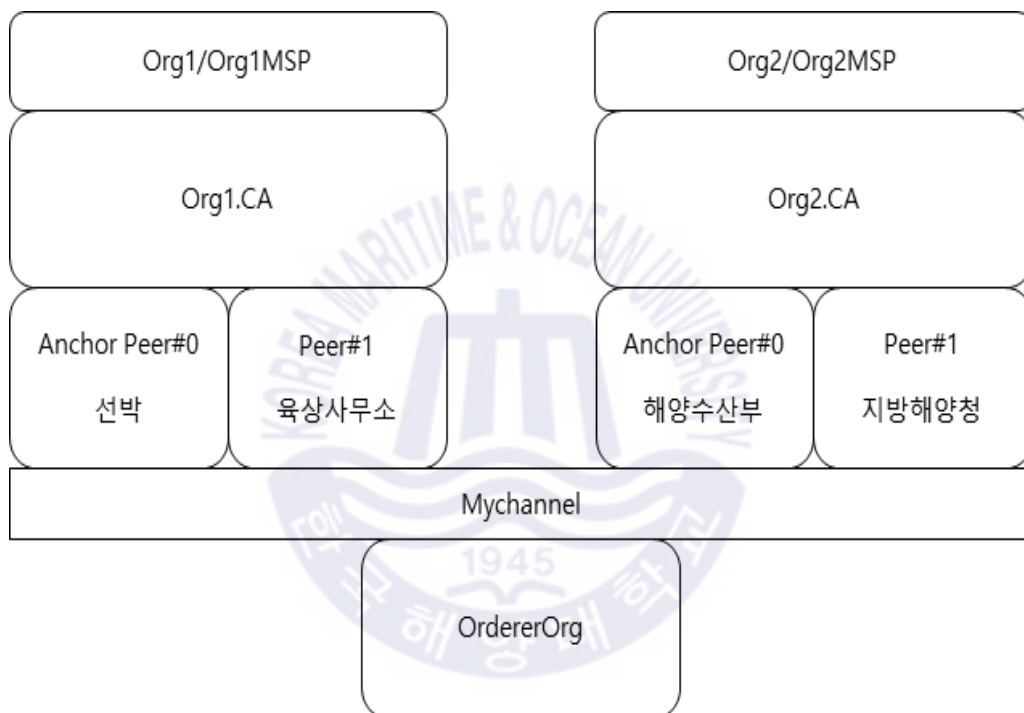


Fig. 3.3 하이퍼레저 패브릭 기반 승하선관리시스템 네트워크 구성도

### 3.3 체인코드 설계

그림 3.4는 승하선관리시스템에서 요구되는 기능을 나타낸다. 하이퍼레저 패브릭 네트워크에서는 네트워크 관리자가 등록되어야 체인코드를 사용할 사용자를 등록 및 관리할 수 있기 때문에 관리할 관리자를 생성하고 등록한다. 관리자의 개인키 및 공개키는 로컬에서 생성된 후 공개키가 CA로 전송되어 API에서 사용할 수 있도록 인증서를 반환한다. 사용자는



각 조직의 CA를 통해 개인키 및 공개키가 발급되며 체인코드를 통해 블록에 쓰인 트랜잭션을 호출할 수 있다. 등록된 사용자는 요구되는 선원의 정보를 작성하여 트랜잭션을 생성하고 원장을 갱신할 수 있고, 원장으로부터 특정 선원의 정보, 모든 선원의 정보를 확인할 수 있다. 승하선관리 시스템에서는 생성된 선원의 객체에서 하선일을 입력하여 등록할 수 있으며 이를 통해 선원의 승선일 및 하선일을 저장하고 확인할 수 있다.

1. 관리자 및 사용자 등록
2. 블록 상태 초기화
  - 단, 관리자 및 사용자가 등록되어 있지 않으면 함수 실행 불가능
3. 블록에 접근하여 데이터 등록 및 호출기능
  - 3-1. 선원 객체 생성을 위한 선원정보인 이름, 선박번호, 선원수첩번호, 승선일이 작성된 트랜잭션 생성 후 체인코드에 전달 후 원장을 갱신
  - 3-2. 블록에 저장되어 있는 특정 선원 정보를 체인코드를 통하여 원장으로부터 쿼리
  - 3-3. 블록에 저장되어 있는 모든 선원 정보를 체인코드를 통하여 원장으로부터 쿼리
  - 3-4. 생성된 선원정보에서 하선일 입력으로 선원의 승하선공인신청 과정을 대신하여 선원의 승선일, 하선일 확인 및 관리
    - 단, 선원 객체 생성 시 승선일 입력, 선원의 하선 시 하선일 기록하는 함수를 통해 이미 등록된 선원 객체 정보를 수정, 하선일 외 다른 정보 변경 불가

Fig. 3.4 승하선관리시스템에서 요구되는 기능

시스템에서 요구되는 기능은 체인코드를 사용하여 함수로 구현할 수 있다. 표 3.2는 승하선관리시스템 기능을 구현할 함수의 이름과 설명을 보여준다. 각 조직은 체인코드에 접근하여 선원의 객체 생성 및 확인을 할 수 있다. 원장의 초기화를 위해 InitLedger 함수를 사용하여 기존의 선원의 정보를 원장에 기록한다. 기록된 트랜잭션을 호출하는 Query 함수가 포함된 모든 선원 정보 호출 함수인 QueryAll과 특정 선원 정보 호출 함수인 QuerySailor는 원장으로부터 트랜잭션을 호출하여 출력한다. 원장에 트랜잭션을 등록하는 Invoke 함수가 포함된 선원 객체 생성 함수 CreateInfo와 하선일 등록 함수 DisEmDate는 출력값 없이 원장에 트랜잭션을 갱신한다.

**Table 3.2** 승하선관리시스템 기능을 구현한 체인코드 함수

함수 이름	설 명
InitLedger	<ul style="list-style-type: none"> <li>원장의 초기화</li> </ul>
QueryAll	<ul style="list-style-type: none"> <li>특정 입력값 없이 블록 내의 모든 선원의 정보를 체인코드로부터 쿼리</li> </ul>
QuerySailor	<ul style="list-style-type: none"> <li>특정 선원의 Key을 입력 시 체인코드로부터 선원의 정보 쿼리</li> </ul>
CreateInfo	<ul style="list-style-type: none"> <li>승하선관리시스템에 필요한 선원정보 이름, 선박번호, 선원수첩번호, 승선일 트랜잭션 생성 후 체인코드에 전달</li> </ul>
DisEmdate	<ul style="list-style-type: none"> <li>특정 선원의 하선일 입력 함수, 특정 선원의 Key값 입력 시 선원의 정보를 쿼리한 후 선원의 하선일을 매개변수로 받아 체인코드에 전달</li> </ul>

### 3.4 미들웨어 설계

하이퍼레저 패브릭은 미들웨어에서 개발자도구를 제공하여 프로그래밍 언어인 Go, NODE.JS, JavaScript를 지원하며 IDE를 통해 개발할 수 있다. 그림 3.5는 API에서 체인코드 호출 시 내부 동작을 보여준다. Application에서는 체인코드를 사용하기 위해 미들웨어를 통해 관리자 및 사용자의 등록이 가능하고 하이퍼레저 패브릭 네트워크 내에 체인코드 배포, 체인코드에서 원장으로 트랜잭션 제출기능이 가능하게 된다. 또한, 체인코드에 Invoke, Query 기능을 실행하여 원장으로부터 트랜잭션 결과가 쿼리된다.

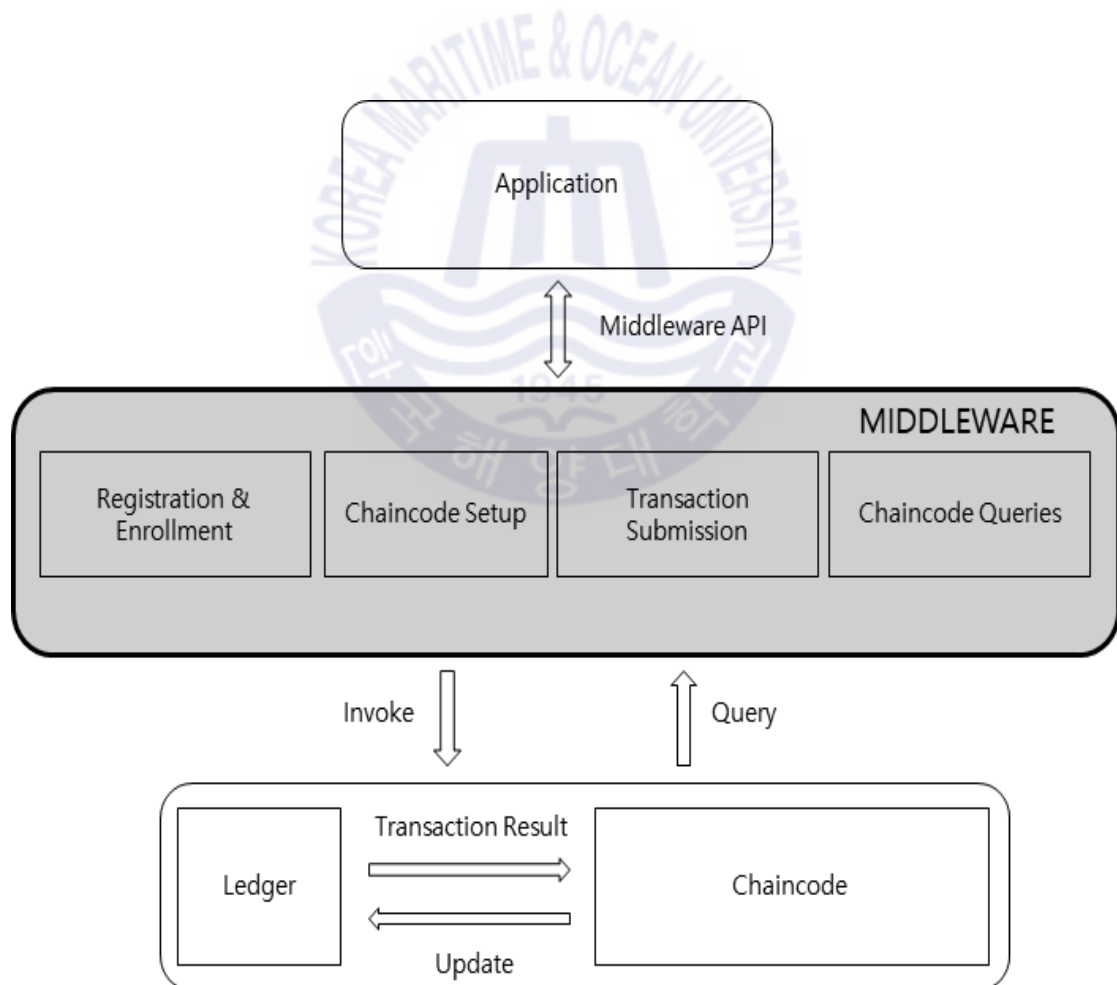


Fig. 3.5 API 체인코드 호출 시 내부 동작

## 제 4 장 승하선관리시스템 개발 및 검증

본 장에서는 설계한 내용을 기반으로 승하선관리시스템을 구현하였다. 구현한 기능인 선원의 정보 생성, 선원의 정보 조회, 모든 선원의 정보 조회, 하선일 입력을 통해 선원의 정보와 공인일의 이력을 관리할 수 있다. 승하선관리시스템의 구현 환경은 표 4.1과 같다.

Table 4.1 시스템 구현환경

항 목	내 용
운영체제	<ul style="list-style-type: none"> <li>• Ubuntu 18.04 LTS</li> </ul>
언어	<ul style="list-style-type: none"> <li>• java Script</li> <li>• Node.js</li> </ul>
플랫폼	<ul style="list-style-type: none"> <li>• Hyperledger Fabric 1.4</li> <li>• Docker 18.09</li> </ul>

제안된 시스템은 Ubuntu 18.04 LTS 운영체제에 Hyperledger Fabric을 설치하여 구현하였으며 개발의 편의성을 위해 Docker를 설치하였고 API와 상호작용하는 미들웨어 언어는 Node.js와 체인코드는 Javascript로 개발하였다.

## 4.1 네트워크 구현 및 실행

하이퍼레저 패브릭에서 네트워크를 구성 시 config.yaml파일을 작성하여 네트워크를 구동시켜야 하고 네트워크 구성에서 가장 먼저 피어들의 조직 및 정렬자의 MSP에 인증서 및 서명키를 생성한다. 조직의 피어 수 및 인증서를 만들어야하는 사용자 수가 포함된다. 조직의 구조에 따라 네트워크를 만들기 위해서 그림 4.1과 같이 Org1, Org2 각각의 Name과 Id를 지정하고 Anchor peer를 설정한다. 네트워크에서 조직을 먼저 생성하고 조직의 인증기관인 CA를 생성해야한다.

```
- &Org1
Name: Org1MSP
ID: Org1MSP
AnchorPeers:
- Host: peer0.org1.example.com
- &Org2
Name: Org2MSP
ID: Org2MSP
AnchorPeers:
- Host: peer0.org2.example.com
# ORG1 CA
ca.org1.example.com:
environment:
- FABRIC_CA_SERVER_CA_NAME=ca.org1.example.com
# ORG2 CA
ca.org2.example.com:
environment:
- FABRIC_CA_SERVER_CA_NAME=ca.org2.example.com
```

Fig. 4.1 하이퍼레저 패브릭 네트워크 조직 구현

그림 4.2는 조직1, 조직2의 피어의 생성을 위한 환경변수 설정 보여준다. 피어를 구동하기 위해서 피어의 Endorsing 피어 역할 여부와 피어의 주소값을 결정한다. 조직1의 피어0과 피어1의 식별ID를 각각 생성하여 발급하고 조직1의 MSP에 등록한다. 조직2의 피어0, 피어1의 식별ID는 조직2의 MSP에 등록한다. 각 피어에 체인코드 관련 정보를 받기 위한 주소값을 결정하고 피어들의 개인키 및 인증서 파일이 저장된 경로를 지정해주면 조직 내에서 피어가 구동가능하게 된다.

```
#peer0.org1
peer0.org1.example.com:
environment:
- CORE_PEER_ENDORSER_ENABLED
- CORE_PEER_ID=peer0.org1.example.com
- CORE_PEER_LOCALMSPID=Org1MSP
- CORE_PEER_CHAINCODEADDRESS

#peer1.org1
peer1.org1.example.com:
environment:
- CORE_PEER_ENDORSER_ENABLED
- CORE_PEER_ID=peer1.org1.example.com
- CORE_PEER_LOCALMSPID=Org1MSP
- CORE_PEER_CHAINCODEADDRESS

#peer0.org2
peer0.org2.example.com:
environment:
- CORE_PEER_ENDORSER_ENABLED
- CORE_PEER_ID=peer0.org2.example.com
- CORE_PEER_LOCALMSPID=Org2MSP
- CORE_PEER_CHAINCODEADDRESS
```

Fig. 4.2 하이퍼레저 패브릭 네트워크 피어 환경변수 설정

```
#peer1.org2
peer1.org2.example.com:
environment:
- CORE_PEER_ENDORSER_ENABLED
- CORE_PEER_ID=peer1.org2.example.com
- CORE_PEER_LOCALMSPID=Org2MSP
- CORE_PEER_CHAINCODEADDRESS
```

Fig. 4.2 하이퍼레저 패브릭 네트워크 피어 환경변수 설정(계속)

그림 4.3은 Ubuntu환경에서 구현한 네트워크 실행화면이다. 네트워크 실행 시 조직1의 피어0, 피어1, 조직2의 피어0, 피어1 및 정렬자가 생성되었음을 확인할 수 있다.

CONTAINER ID	IMAGE	PORTS	NAMES
021066cc67b2	hyperledger/fabric-tools		cli_org2
5f556ac348dc	hyperledger/fabric-tools		cli_org1
6732405dcec0	hyperledger/fabric-peer	0.0.0.0:18051->18051/tcp, 0.0.0.0:18053->18053/tcp	peer1.org1.example.com
0a02e2c8e9d6	hyperledger/fabric-peer	0.0.0.0:17051->17051/tcp, 0.0.0.0:17053->17053/tcp	peer0.org1.example.com
184b35ce1d9d	hyperledger/fabric-peer	0.0.0.0:27051->27051/tcp, 0.0.0.0:27053->27053/tcp	peer0.org2.example.com
b9a0b4e0dda3	hyperledger/fabric-peer	0.0.0.0:28051->28051/tcp, 0.0.0.0:28053->28053/tcp	peer1.org2.example.com
db79d4f6fe25	hyperledger/fabric-orderer	0.0.0.0:7050->7050/tcp	orderer.example.com
2bcf8ae57032	hyperledger/fabric-couchdb		

Fig. 4.3 우분투 환경에서 네트워크 실행화면

그림 4.4는 네트워크 생성 시 생성되는 하부 디렉터리를 나타낸다. 조직1 하부디렉터리에 조직1을 인증하기 위해 조직1의 CA가 생성되었고 조직1의 사용자를 관리하기 위한 조직1의 MSP, 조직1의 피어 디렉터리가 생성되었다. 조직2 하부디렉터리에는 조직2의 CA, 조직2의 MSP, 조직2의 피어가 생성되었고 정렬자 조직 디렉터리에 CA 및 MSP가 생성되었음을 확인할 수 있다.

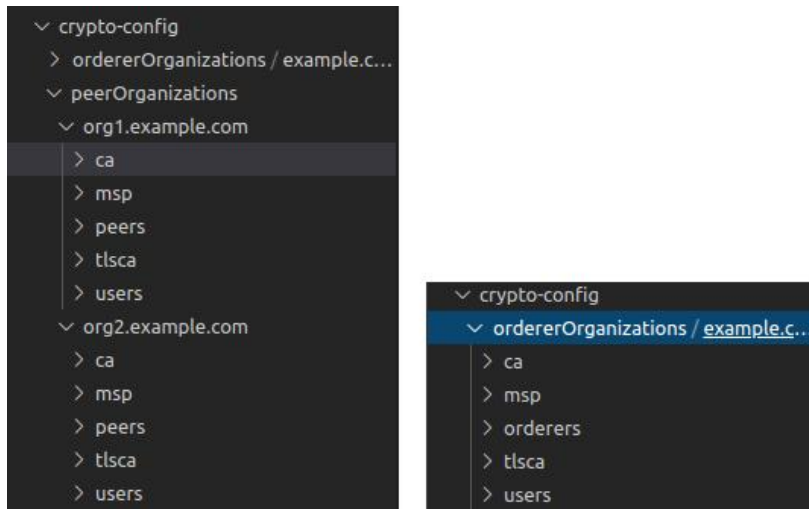


Fig. 4.4 조직 및 정렬자의 하부 디렉터리

## 4.2 체인코드 및 미들웨어 구현

### (1) 체인코드 구현

그림 4.5는 승하선시스템에서 트랜잭션 생성 시 필요한 데이터를 구조체로 구현한 것으로서 선원의 이름(Name), 선박의 번호(ShipNum), 선원수첩의번호(LedgerNum), 승선일(EDate), 하선일(DDate)를 나타낸다.

```

type Man struct {
    Name      string `json:"name"`
    ShipNum   string `json:"shipNum"`
    LedgerNum string `json:"ledgerNum"`
    EDate     string `json:"승선일"`
    DDate     string `json:"하선일"`
}

```

Fig. 4.5 승하선관리시스템의 데이터 구조체

그림 4.6은 승하선시스템에서 필요한 기능호출을 함수로 구현한 것을 나타낸다. 앞서 설계한 표 3.2 함수의 기능을 구현한 것으로서 각 기능을 실행할 수 있다.



```

func (s *SmartContract) Invoke(APIStub shim.ChaincodeStubInterface) sc.Response {

    function, args := APIStub.GetFunctionAndParameters()

    if function == "querySailor" {
        return s.querySailor(APIStub, args)
    } else if function == "initLedger" {
        return s.initLedger(APIStub)
    } else if function == "createInfo" {
        return s.createInfo(APIStub, args)
    } else if function == "queryAll" {
        return s.queryAll(APIStub)
    } else if function == "DisEmDate" {
        return s.DisEmDate(APIStub, args)
    }

    return shim.Error("Invalid Smart Contract function name.")
}

```

Fig. 4.6 승하선관리시스템의 함수목록

그림 4.7은 블록의 초기화를 위한 함수이다. InitLedger는 트랜잭션을 호출하기 위하여 작성한 코드이다. 네트워크를 구성한 후 8명의 선원 객체의 정보인 이름, 선박번호, 선원수첩번호, 승선일이 기록되어있고 하선일은 ‘ - ‘로 입력되어 Query 또는 Invoke할 수 있다. 함수를 구현하고 체인코드를 실행하는 즉시 블록이 초기화되지는 않으며 함수 실행 시 트랜잭션이 블록에 등록된다.

```

func (s *SmartContract) initLedger(APIStub shim.ChaincodeStubInterface) sc.Response {
    mans := []Man{
        Man{Name: "이정옥", ShipNum: "A-15421", LedgerNum: "A124563", EDate: "2020-05-09", DDate: "-"},
        Man{Name: "김종엽", ShipNum: "A-15213", LedgerNum: "A545121", EDate: "2020-06-09", DDate: "-"},
        Man{Name: "여동규", ShipNum: "B-41244", LedgerNum: "B545421", EDate: "2020-05-09", DDate: "-"},
        Man{Name: "이혜주", ShipNum: "B-45214", LedgerNum: "B858851", EDate: "2015-05-15", DDate: "-"},
        Man{Name: "김정래", ShipNum: "S-54874", LedgerNum: "B557451", EDate: "2018-07-28", DDate: "-"},
        Man{Name: "박진화", ShipNum: "Z-15154", LedgerNum: "B888441", EDate: "2019-08-15", DDate: "-"},
        Man{Name: "김진상", ShipNum: "W-48842", LedgerNum: "C548421", EDate: "2020-03-02", DDate: "-"},
        Man{Name: "유성화", ShipNum: "W-48815", LedgerNum: "C888842", EDate: "2020-07-05", DDate: "-"},
    }
}

```

Fig. 4.7 InitLedger 함수 구현

## 2) 미들웨어 구현

그림 4.8은 Node.js기반 관리자 등록 기능을 나타낸다. 관리자가 등록된 후 사용자등록을 위해 관리자를 먼저 등록한다. 관리자가 이미 등록되어 있을 경우 존재한다는 메시지를 출력하고 미등록 시 관리자의 ID 및 비밀번호를 발급하고 X509기반 공개키, 개인키를 발급하여 wallet폴더에 저장하여 관리한다.

```
// 관리자 ID를 저장하기 위한 wallet폴더 생성
const walletPath = path.join(process.cwd(), 'wallet');
const wallet = new FileSystemWallet(walletPath);
console.log('Wallet path: ${walletPath}');
// 이미 관리자가 존재할 시 메시지 출력
const adminExists = await wallet.exists('admin');
if (adminExists) {
  console.log('이미 관리자가 등록되어 있습니다. ');
  return;
}
// 새로운 관리자 생성 시 관리자 ID 및 비밀번호 생성
const enrollment = await ca.enroll({ enrollmentID: 'Admin', enrollmentSecret: '1234' });
//X509기반 공개키 개인키 발급
const identity =X509WalletMixin.createIdentity('OrgMSP', enrollment.certificate, enrollment.key.toBytes());
//발급된 관리자의 아이디 wallet폴더에 저장
wallet.import('admin', identity);
```

**Fig. 4.8** 관리자 등록 기능을 위한 미들웨어 구현

그림 4.9는 사용자 등록 기능을 나타낸다. 관리자가 등록되었다면 트랜잭션을 생성할 사용자를 생성한다. 사용자의 ID 및 비밀번호를 발급한다. 사용자는 소속되는 조직의 MSP로부터 X509기반의 공개키, 개인키를 발급받고 관리하기 위해 wallet폴더에 저장한다.

```

// 'user1' 사용자ID 및 비밀번호 생성
const secret =await ca.register({enrollmentID: 'user1'}, adminIdentity);
const enrollment = await ca.enroll({ enrollmentID: 'user1', enrollmentSecret:
1234 });
// 조직 1에 속한 user1의 X509기반 개인키 공개키발급 및 wallet에 저장
const userIdentity= X509WalletMixin.createIdentity('Org1MSP', enroll
ment. certificate, enrollment.key.toBytes());
wallet.import('user1', userIdentity);

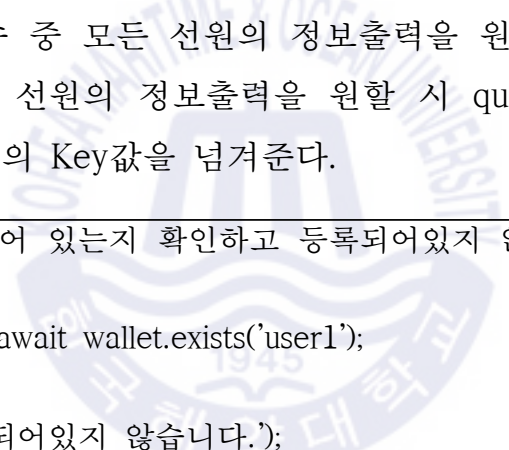
```

**Fig. 4.9** 사용자 등록 기능을 위한 미들웨어 구현

그림 4.10은 원장에서 쿼리를 수행하기 위한 기능이다. 쿼리를 수행하기 위해서 사용자가 등록되어 있는지 확인 후 기능을 수행한다. 체인코드에서 구현된 쿼리 함수 중 모든 선원의 정보출력을 원할 시 queryAll 함수를 호출하거나 특정 선원의 정보출력을 원할 시 querySailor함수와 함께 매개변수로 선원객체의 Key값을 넘겨준다.

```

// 사용자가 등록되어 있는지 확인하고 등록되어있지 않으면 에러메세지 출
력
const userExists = await wallet.exists('user1');
if (!userExists) {
  console.log('등록되어있지 않습니다.');
```



```

  return;
}
// 네트워크에 배포된 체인코드로 접근
const contract = network.getContract('mychaincode');
```

// 체인코드에서 구현된 queryall함수 호출

```

const result = await contract.evaluateTransaction('queryAll');
```

// 특정 선원의 정보를 쿼리할 시 key를 매개변수로 넘겨준다.

```

const result = await contract.evaluateTransaction('querySailor', 'MAN4')
console.log(' result : ${result.toString()}');
```

**Fig. 4.10** Query기능을 위한 미들웨어 구현

그림 4.11은 원장으로 Invoke를 수행하기 위한 기능이다. Invoke를 수행하기 위해서 사용자가 등록되어 있는지 확인 후 기능을 수행한다. 체인코드 중 선원 객체생성 및 하선일 입력 함수는 Invoke 기능이 포함되어 있어 이를 사용하기 위해 작성된다. 선원의 객체를 생성할 시 필요한 데이터인 이름, 선박번호, 선원수첩번호, 승선일을 매개변수로 받아 트랜잭션을 작성하여 제출한다. 제출된 트랜잭션은 원장으로 Invoke된다. 특정 선원의 하선일 입력이 요구되면 DisEmDate함수에 선원의 Key값과 하선하는 날짜를 매개변수로 넘겨주고 원장으로 Invoke한다.

```
// 사용자가 등록되어 있는지 확인하고 등록되어있지 않으면 에러메세지 출력
const userExists = await wallet.exists('user1');
if (!userExists) {
  console.log('등록되어 있지 않습니다.');
```

OCEAN MARITIME & OCEAN UNIVERSITY  
1945  
해양대학교

```
  return;
}
// 네트워크에 배포된 체인코드로 접근
const contract = network.getContract('mychaincode');
```

// 트랜잭션 제출

#1. 선원객체생성 트랜잭션 - 요구되는 매개변수(CreateInfo, '선원객체의 Key값', '이름', '선박번호', '선원수첩번호', '승선일', '하선일')

#2. 하선일변경 트랜잭션 - 요구되는 매개변수(DisEmDate, '선원객체의 Key값', '하선하는 날짜')

```
await contract.submitTransaction('함수이름', '요구되는 매개변수');
```

Fig. 4.11 Invoke기능을 위한 미들웨어 구현

### 4.3 승하선관리시스템 검증

하이퍼레저 패브릭에서 검증을 위해서 가장 먼저 관리자 및 사용자 등록 여부에 따라 네트워크 참여자가 아닐 시 기본적으로 네트워크 접근에 제한을 두어야한다. 네트워크 구성 시 정의한 채널의 수, 조직의 수, 피어의 수가 일치하는지 여부를 확인하여야 하고 네트워크가 구성이 끝난 뒤 여러 실험 시나리오를 통해서 비즈니스 거버넌스에 맞게 트랜잭션이 블록으로부터 호출되거나 블록에 등록되는 결과를 확인할 수 있어야 한다. 실험을 위하여 체인코드를 통해 임의로 작성된 트랜잭션을 블록에 등록할 수 있게 블록의 초기화를 구현했으며 함수의 기능이 올바르게 동작하는지 블록의 호출 및 갱신 실험을 하였다. 다음은 승하선관리시스템 기능 실험 단계이다.

#### (1) 관리자 및 사용자 등록

그림 4.12는 앞서 구현한 관리자, 사용자 등록을 실행 했을 시 결과이다. 관리자와 사용자가 등록되어 개인키, 공개키가 발급된 것을 확인할 수 있고 관리자는 CA와 같은 역할로서 사용자의 개인키를 보관하고 사용자 증명을 하고 트랜잭션이 유효한지 확인하는데 사용된다. 사용자는 자신의 개인키와 공개키를 사용하여 트랜잭션을 디지털 서명한다. 이 서명은 MSP의 일부인 공개키 일치여부를 판별하고 트랜잭션을 증명하는데 사용된다. 등록된 사용자는 블록의 트랜잭션을 생성 및 읽을 수 있다.

```
fabric@fabric-H81H3-M3:~/shipmanage/manager/exe$ tree wallet
wallet
├── admin
│   ├── 5691004dfd03d7580f2d67c546e8f5849cf928bcbbfdc6224b7cb2b896b6012f-priv
│   ├── 5691004dfd03d7580f2d67c546e8f5849cf928bcbbfdc6224b7cb2b896b6012f-pub
│   ├── 89817a2c807572e75fc4dcba68bcb4795d223118bc6a78a758fda08834bb6dda-priv
│   └── admin
└── user1
    ├── 89817a2c807572e75fc4dcba68bcb4795d223118bc6a78a758fda08834bb6dda-priv
    ├── 89817a2c807572e75fc4dcba68bcb4795d223118bc6a78a758fda08834bb6dda-pub
    └── user1
```

Fig. 4.12 우분투 환경에서 관리자 및 사용자 등록 실행화면

## (2) 체인코드 생성 및 배포

그림 4.13은 체인코드 생성 및 배포과정을 보여준다. 3개의 과정으로 코드 생성 완료 및 패키지화된 체인코드는 네트워크에 배포과정에서 설치 후 인스턴스화가 완료되어야 체인코드를 실행할 수 있으며 등록된 사용자들을 통하여 트랜잭션을 체인코드에 Query 또는 Invoke할 수 있다.

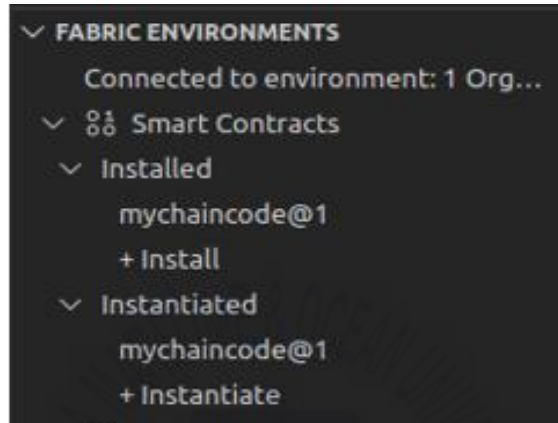


Fig. 4.13 체인코드 설치 및 인스턴스화 실행화면

## (3) 함수 실행 및 확인

### 1) InitLedger

구현한 체인코드기능 동작을 보기위해 앞서 구현한 InitLedger를 통하여 원장을 초기화한다. 기존의 선원객체를 개발자가 미리 등록해두어 승하선 관리시스템에서 선원의 정보를 읽을 수 있다.

### 2) QueryAll

그림 4.14는 queryAll함수 호출 시 출력화면이다. 다음과 같이 선원객체의 Key값을 포함한 선원의 정보가 출력되는 것을 볼 수 있다.

```
[2020. 7. 17. 오후 1:54:43] [SUCCESS] Returned value from queryAll: [{"Key": "Man0", "Info": {"LedgerNum": "A124563", "name": "이정욱", "shipNum": "A-15421", "승선일": "2020-05-09", "하선일": "-"}}, {"Key": "Man1", "Info": {"LedgerNum": "A545121", "name": "김종엽", "shipNum": "A-15213", "승선일": "2020-06-09", "하선일": "-"}}, {"Key": "Man2", "Info": {"LedgerNum": "B545421", "name": "여동규", "shipNum": "B-41244", "승선일": "2020-05-09", "하선일": "-"}}, {"Key": "Man3", "Info": {"LedgerNum": "B858851", "name": "이혜주", "shipNum": "B-45214", "승선일": "2015-05-15", "하선일": "-"}}, {"Key": "Man4", "Info": {"LedgerNum": "B557451", "name": "김정래", "shipNum": "S-54874", "승선일": "2018-07-28", "하선일": "-"}}, {"Key": "Man5", "Info": {"LedgerNum": "B888441", "name": "박진화", "shipNum": "Z-15154", "승선일": "2019-08-15", "하선일": "-"}}, {"Key": "Man6", "Info": {"LedgerNum": "C548421", "name": "김진상", "shipNum": "W-48842", "승선일": "2020-03-02", "하선일": "-"}}, {"Key": "Man7", "Info": {"LedgerNum": "C88842", "name": "유성화", "shipNum": "W-48815", "승선일": "2020-07-05", "하선일": "-}}]
```

Fig. 4.14 QueryAll함수 실행 결과화면

### 3) QuerySailor

그림 4.15은 querySailor함수 호출 시 출력화면이다. 선원객체의 Key값을 매개변수로 작성하여 함수호출하면 다음과 같이 특정 선원의 정보를 쿼리하여 출력한다.

```
querySailor  
["Man0"]  
querySailor: {"ledgerNum":"A124563","name":"이정욱","shipNum":"A-15421","승선일":"2020-05-09","하선일":"-"}  
}
```

Fig. 4.15 QuerySailor함수 실행 결과화면

### 4) CreateInfo

그림 4.16은 CreatInfo함수 호출 시 출력화면이다. CreateInfo함수는 Invoke함수이므로 출력값이 없다. 함수를 호출하여 선원객체의 Key값, 선원의 이름, 선박번호, 선원수첩번호, 승선일을 매개변수로 작성한 후 실행한다. 출력값을 확인하기 위해 QueryAll함수를 사용하면 다음과 같이 모든 선원의 정보에서 ‘아무개’ 라는 선원의 정보가 추가된 것을 확인할 수 있다.

```
createInfo  
["Man8","아무개","S-15111","A764586","2020-05-20"]  
[SUCCESS] Returned value from queryAll: [{"Key":"Man0","Info":{"ledgerNum":"A124563","name":"이정욱","shipNum":"A-15421","승선일":"2020-05-09","하선일":"-"}}, {"Key":"Man1","Info":{"ledgerNum":"A545121","name":"김종엽","shipNum":"A-15213","승선일":"2020-06-09","하선일":"2019-09-10"}}, {"Key":"Man2","Info":{"ledgerNum":"B545421","name":"여동규","shipNum":"B-41244","승선일":"2020-05-09","하선일":"-"}}, {"Key":"Man3","Info":{"ledgerNum":"B858851","name":"이해주","shipNum":"B-45214","승선일":"2015-05-15","하선일":"-"}}, {"Key":"Man4","Info":{"ledgerNum":"B557451","name":"김정래","shipNum":"S-54874","승선일":"2018-07-28","하선일":"-"}}, {"Key":"Man5","Info":{"ledgerNum":"B888441","name":"박진화","shipNum":"Z-15154","승선일":"2019-08-15","하선일":"-"}}, {"Key":"Man6","Info":{"ledgerNum":"CS48421","name":"김진상","shipNum":"W-48842","승선일":"2020-03-02","하선일":"-"}}, {"Key":"Man7","Info":{"ledgerNum":"C888842","name":"유성화","shipNum":"W-48815","승선일":"2020-07-05","하선일":"-"}}, {"Key":"Man8","Info":{"ledgerNum":"A764586","name":"아무개","shipNum":"S-15111","승선일":"2020-05-20","하선일":"-"}}]  
{"ledgerNum":"A764586","name":"아무개","shipNum":"S-15111","승선일":"2020-05-20","하선일":"-"}  
}
```

Fig. 4.16 CreateInfo함수 실행 결과화면

## 5) DisEmDate

그림 4.17은 DisEmDate함수 호출 시 출력화면이다. DisEmDate함수는 CreateInfo함수와 마찬가지로 Invoke함수이므로 출력값이 없고 querySailor 또는 queryAll함수를 통해 확인할 수 있다. DisEmDate함수는 선원객체의 Key값과 하선일을 매개변수로 받고 선원객체의 Key값으로 특정 선원객체를 호출하여 선원의 정보 중 하선일을 입력한다.

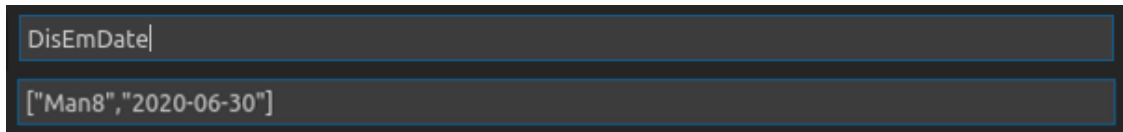


Fig. 4.17 DisEmDate함수 호출 및 매개변수 입력화면

그림 4.18은 그림 4.16에서 생성한 ‘아무개’의 하선일 입력 전과 후를 나타낸다. 출력값을 확인하기 위해 querySailor함수를 사용하여 ‘아무개’ 선원의 정보를 확인하면 다음과 같이 입력된 날짜로 하선일이 변경되었음을 확인할 수 있다.

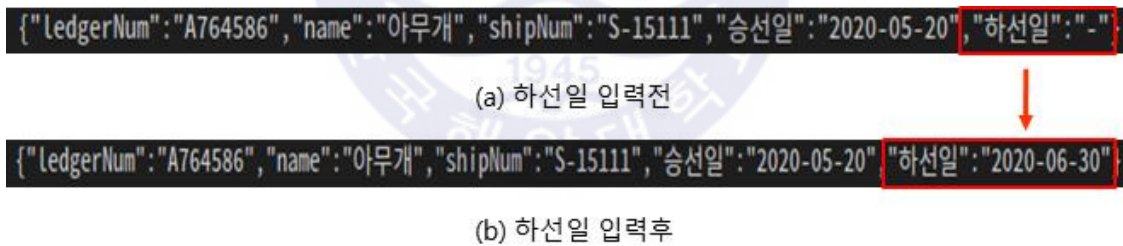


Fig. 4.18 DisEmDate함수 실행 결과화면



## 제 5 장 결 론

본 논문에서는 하이퍼레저 패브릭 기반의 선원 승하선관리시스템을 제안하였다. 현재 선원의 승하선공인신청 시의 과정은 서류를 제출하기 위해 관할 지방해양청에 직접 방문하여야 하는 불편함과 서류 처리시간이 오래 걸린다는 단점이 존재한다. 또한, 제출해야 하는 서류에는 개인정보 및 노출되지 말아야 할 정보가 담겨 있어 보안의 문제와 분실의 위험이 있다. 승하선할 때마다 매번 공인신청을 해야 하기 때문에 일괄적으로 처리할 수 있는 방법이 존재하지 않고 서류제출자의 신원 및 서류의 무결성 또한 문제가 될 수 있고 서류를 처리하는 기관의 중앙집중적 구조로 인해 단일지점의 해킹위험을 지적하였다.

본 논문에서는 탈중앙형태의 대표적 기술인 블록체인을 통해 데이터의 무결성을 제공하고 서류의 디지털화를 통한 서류 보관의 용이함, 승하선공인 시 필요한 서류의 간소화를 제공하였다. 특히 블록체인 기술 중 비즈니스 거버넌스에 맞는 인증된 기관만 네트워크에 참여할 수 있는 허가형 블록체인인 하이퍼레저 패브릭 기술을 적용함으로써 네트워크 참여자들만 정보를 공유하게 해 제 3자에게 개인정보 및 중요한 정보를 노출시키지 않고 체인코드에 구현된 함수를 사용하여 사용자는 승하선공인업무를 수행할 수 있도록 하였다.

향후 연구로는 복잡한 해상통신환경을 고려하여 실시간으로 통신이 가능한 통신장비를 통한 블록체인 네트워크를 구성하여 승하선처리과정을 해상에서도 실시간으로 처리할 수 있도록 하는 것이 필요하다. 또한, 선원의 선원수첩을 디지털화하여 해상환경에서 실시간으로 위변조없이 신원을 확인하고 데이터를 교환할 수 있는 시스템의 개발이 요구된다.

## 참고문헌

- [1] 해양수산부(선원정책과), 2019. 선원법. 법률 제 16281호.
- [2] 국제해사기구(IMO), 2019. 제6차 인적요소, 훈련 및 당직 전문위원회 결과보고서. The 6th Session of HTW Sub-COMMITTEE, pp. 114-118.
- [3] 한경현, 황성운, 2018. Introduction of blockchain. 정보과학회지, 36(12), pp. 11-16.
- [4] 정성교, 2018. 블록체인 기술 및 연구 동향 분석. KEREC(Korea-EU Research Centre).
- [5] 이경호. 2018. 블록체인 유형에 대한 연구: 이더리움, 하이퍼레저 패브릭. 한국통신학회 학술대회논문집, pp. 442-443.
- [6] 남윤형, 우요섭. 2019. PKI 가 필요없는 종이문서 위변조 검증 시스템 구현. 한국정보과학회 학술발표논문집, pp. 2151-2153.
- [7] 박승규, 박종진, 이동욱. 2018. 해시 트리 기반의 대규모 데이터 서명 시스템 구현. 전자공학회논문지, 55(5), pp. 43-50.
- [8] 윤태영. 2019. 블록체인 기술을 이용한 스마트계약 (Smart Contract). 재산법연구, 36(2), pp. 69-88.
- [9] 임종철, 고남석, 2020. 세대별 블록체인 합의 알고리즘. 한국통신학회지, 37(3), pp. 3-12.
- [10] 손상진, 권순홍, 명세인, 이종혁. 2019. 이더리움과 하이퍼레저 패브릭에서의 트랜잭션 및 스마트 컨트랙트 동작 분석. 한국통신학회 학술대회논문집, pp. 205-206.
- [11] 배수환, 조선욱, 신용태. 2019. 하이퍼레저 패브릭 블록체인을 활용한 시스템 복구 기법. 한국정보전자통신기술학회 논문지, 12(2), pp. 155-161.

- [12] 홍상원, 노용두, 박찬익. 2019. 하이퍼레저 패브릭을 위한 확장 가능한 대체 불가능 토큰 프레임워크에 관한 연구. 한국정보과학회 학술발표논문집, pp. 785-787.
- [13] 금창섭. 2018. 블록체인 미디어 적용 가능성과 블록체인 아키텍처 분석. 방송과 미디어, 23(3), pp. 85-91.
- [14] 이부형, 임연주, 이종혁. 2017. 블록체인 플랫폼에서의 합의 알고리즘. 한국통신학회 학술대회논문집, pp. 386-387.
- [15] 박진상, 김정덕. 2019. 컨소시엄 블록체인 거버넌스 프레임워크 개발에 관한 연구. 디지털융복합연구, 17(8), pp. 89-94.
- [16] 박승철. 2019. PKI 인터넷 뱅킹과 블록체인 지불 거래의 비교 분석. 한국정보통신학회논문지, 23(5), pp. 604-612.
- [17] 최대선, 2018. 블록체인과 인증. 한국정보통신학회 한국정보통신학회지, 35(7), pp. 11-17.
- [18] 김순곤, 2018. DApp 개발을 위한 블록체인 2.0 이더리움 플랫폼 분석 연구. 한국정보통신학회 한국정보통신기술학회 논문지, pp 718-723.
- [19] 강희정. 2019. 신뢰할 수 있는 허가형 블록체인 기반 전자투표 시스템 설계 및 구현. 국내석사학위논문 성신여자대학교 대학원, 서울
- [20] 노시완, 이경현. 2018. 퍼블릭 블록체인의 보안 위협과 블록체인 확장성 문제의 연관성에 대한 분석. 정보보호학회지, 28(3), pp. 26-32.
- [21] 하태윤, 이충현, 장건희, 조성래, 2018. Gossip protocol기반의 Hash-graph에 관한 연구. 한국통신학회지, pp 284-285.