碩士學位論文

블록체인 기반 어류품질검사 데이터의 무결성 서비스

濟州大學校 大學院

컴 퓨 터 工 學 科

金 滓 民

2020年 8月

블록체인 기반 어류품질검사 데이터의 무결성 서비스

指導敎授 金 度 縣

金 滓 民

이 論文을 컴퓨터工學 碩士學位 論文으로 提出함

2020 年 8 月

金滓民의 工學 碩士學位 論文을 認准함

審査委員長 김 수 훈 ㊞

委 員 이 윤 정 ㊞

委 員 김 도 현 ㊞

濟州大學校 大學院

2020 年 8 月

Data Integrity Service of Fish Quality Inspection Based on Blockchain

Jae Min Kim
(Supervised by professor Do Hyeun Kim)

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Science in Computer Engineering

2020. 8.

This Thesis has been examined and approved.

Thesis drector, Soo Kyun Kim
Thesis drector, Yunjung Lee
Thesis drector, DoHyeun Kim
August 2020

Department of Computer Engineering
GRADUATE SCHOOL
JEJU NATIONAL UNIVERSITY

목 차

그림 목차

표 목차

국 문 초 록

블록체인 기반 어류품질검사 데이터의 무결성 서비스 연구

컴퓨터 공학과 김 재 민

지도교수 김 도 현

본 논문에서는 블록체인 기술을 접목하여 어류품질 안전성 검사의 신뢰성 및 투명성, 무결성을 보장하기 위한 서비스를 제시한다. 이를 위해 어류품질검사 블록체인 네트워크를 설계하고 구현한다. 어류품질 안전성 검사는 모든 수산물이 유통되기 전 즉 판매자와 구매자의 거래 전에 필수적으로 유통어류에 대한 안전성 검사를 받고 적합판정결과가 나와야 판매 구매 할 수 있다. 유통 전 실시하는 안전성 검사를 블록체인으로 구현함으로써 각 기관과 유통자들의 상호 감시체제를 형성하고, 악의적인 정보 또는 검사자 실수로 발생할 수 있는 허위 검사결과 발생을 방지하여 기관과 품질의 신뢰도를 높인다.

주제어 : 어류안전성 검사, 블록체인, 무결성, 하이퍼레저 패브릭

Abstract

Data Integrity Service of Fish Quality Inspection Based on Blockchain

Kim, JaeMin

Department of Computer Engineering

Graduate School

Supervised by Professor Kim, DoHyeun

In this paper, we propose a service to ensure the reliability, transparency, and integrity of fish quality safety inspection by integrating blockchain technology. To this end, we design and implement a fish quality inspection blockchain network. Fish quality safety inspection is essential before all marine products are distributed, that is, before the transaction between the seller and the buyer. By implementing a safety inspection conducted before distribution in a block chain, a mutual monitoring system between each agency and distributors is formed, and the quality of the agency and quality is increased by preventing the occurrence of malicious information or false inspection results that can be accidentally caused by the inspector.

Keyword : Fish safety inspection, Blockcahin, Integrity, Hyperledger Fabric

Ⅰ. 서 론

1. 연구배경 및 필요성

2011년 동일본대지진으로 인한 방사능 오염수의 바다 유입사건이 있었다[1]. 이 지진은 태평양 앞바다에서 일어난 해저 거대지진으로 일본 근해에 쓰나미가 일어나 일본의 여러 원자력 발전소 사고가 일어났으며 많은 방사능 유출이 발생했다. 이후 후쿠시마(福島) 외 8개 해역에서 잡은 수산물 수입이 금지되었고 국내 어류 안전성 조사에 대한 중요도가 높아졌다. 국립수산물 품질관리원에서는 매해 안전성 조사를 시행하고 있고, 조사 건수도 매해 증가하고 있는 추세다. 어류 안전성 조사는 세부적으로 안전성 조사와 안전성 검사로 구분되며 모든 수산물은 유통되기 전 필수적으로 유통하려는 어류에 대한 안전성 검사를 받고 적합 판정결과가 나와야 판매 구매 할 수 있다. 즉 판매자와 구매자는 거래 전에 필수적으로 안전성 검사를 받아야 한다. 제주해양수산연구원에서 시행하는 어류품질안전성 검사의 신뢰성 및 투명성, 무결성, 부인방지를 보장하기 위한 데이터의 무결성 서비스 방안으로 블록체인 기술의 접목이 필요하다. 현재 제주특별자치도에서 가장 많이 유통 되는 광어의 안전성 검사과정을 블록체인으로 구현함으로써 각 유관기관과 유통자들의 상호 감시체제를 형성하고, 고의적 정보조작 또는 검사자 실수로 발생 할 수 있는 허위 검사결과 발생을 방지하여 기관과 품질의 신뢰도를 높이기 위한 방안을 제시한다.

2. 연구목적 및 내용

어류품질안전성검사의 중요성을 파악하고 검사기관, 승인기관, 유통자를 중심

으로 어류품질안전성 검사정보 및 거래정보를 생성, 관리, 열람 할 수 있도록 '블록체인 기반 어류품질검사 데이터의 무결성 서비스'를 개발한다. 본 논문에서 다루고자 하는 연구내용의 범위는 다음과 같다.

첫째, 어류품질안전성 검사정보의 사용자 관리 및 접근 권한 설계

어류품질안전성 검사정보에 대한 소유권한, 접근권한, 관리 권한이 유통자 자기정보통제권에 있어서 중요한 요소이다. 현재 유통자의 어류품질안전성 검사정보는 검사기관과 승인기관에 분산 저장·관리 되고 유통자는 결과정보를 결과지 또는 결과 로데이터(rawdata)형태로 만 보유하고 있으므로 검사정보에 대한 관리 및 접근권한이 통제 되지 않고 있다. 블록체인 기반 어류품질검사 데이터의 무결성 서비스는 유통자의 검사정보에 대하여 소유권한, 접근권한, 관리 권한을 가지고 정보를 통제할 수 있어야 한다.

둘째, 어류품질안전성 검사정보의 투명한 공개가 가능한 블록체인 개발

어류품질안전성 검사정보를 현재는 검사기관, 승인기관이 보유하고 있으며 이 정보를 열람한 관리자나 열람한 유통자 파악이 불가능하다. 어류품질안전성 검사정보는 소유자가 공개를 허용한 대상에게만 공개하도록 설계 돼야 하며, 승인기관, 검사기관의 직접관리자가 정보를 확인한 경우 검사결과 열람정보가 기록으로 남아서 해당 검사정보 소유자가 그 기록을 확인 할 수 있어야 한다.

셋째, 어류품질안전성 검사정보의 신뢰성 및 부인방지 관리 서비스 개발

검사기관이 생성한 검사자료가 직접 측정 되어 변경 없이 수집 되었는지 검사과정에 발생 할 수 있는 검사자의 실수가 발생 했는지에 대한 무결성 이슈를 해결해야 한다. 제안하는 블록체인 기반 어류품질검사 데이터의 무결성 서비스는 허가된 자(검사자, 승인자, 유통자)가 자료를 생성한 것과 생성된 자료가 변경되지 않고 수집, 저장, 열람됨을 확인 할 수 있어야 한다. 또한 허가된 자의 실수로

발생한 자료에 대한 책임 부인방지가 되어야 한다.

3. 논문의 구성

논문의 구성은 다음과 같다. 2장은 관련연구 부분으로 어류품질안전성 검사의 중요성 및 법제 상황을 설명한다. 그리고 제주특별자치도에서 시행하고 있는 어류품질안전성 검사 구조를 파악하고 각 기관별로 어류품질안전성 검사 정보의 이동을 살펴본다. 이 검사정보의 이동단계별 보안위협 및 보안요구사항들에 대해 파악하고 블록체인 기반 무결성 서비스를 분석하여 본 제안방법의 기반이 되는 기술인 블록체인기반 무결성 서비스 사례를 살펴본다. 3장에서는 블록체인 기반 어류품질검사 데이터의 무결성 서비스에 도입하기 위한 분석에 대해 설명하고 안전하게 생성, 관리, 열람하기 위한 블록체인 기반 어류품질검사 데이터의 무결성 서비스의 구성요소와 요구되는 기능, 시스템을 통해 어류품질안전성 검사정보가 열람되는 절차에 대해 소개한다. 4장에서는 블록체인 기반 어류품질검사 데이터의 무결성 서비스의 프로토타입 구현을 통해 기본적으로 필요시 되는 기능을 구현한다. 5장은 결론으로 제안기법에 대해 간략하게 요약하고 논문의 목적 및 향후 연구방향에 대해 소개한다.

II. 관련연구

1. 어류품질안전성 검사 현황

국내에서 생산 또는 유통되는 수산물의 유통단계를 파악하고 그 과정에 시행되는 어류품질안전성 검사의 방법 및 처리방법 등을 알아보고 관련 법제를 파악하고자 한다. 또한 제주도특별자치도에서 시행하는 어류품질안전성 검사 현황을 파악하여 제주특별자치도의 어류품질안전성 검사를 블록체인에 도입하는 방안에 대해서 분석한다.

우리나라는 수산물의 품질향상과 안전한 수산물을 생산하고 공급하기 위해 농수산물 품질관리법 제 61조[2] 및 수산물 안전성조사업무 처리요령[행정규칙][3]에 근거로 안전성 조사 및 검사를 시행하고 있다. 국립수산물품질관리원[4] 통계에 따르면 매해 안전성 조사건수는 증가하고 있으며 2018년 총 건수는 9,000건 이상에 육박하고 있다. 안전성조사는 양식산, 연근해산, 원양산, 수출, 인증 등 용도별로 분석하고 수산물에 잔류된 중금속・항생물질・식중독균・방사능 등의 유해물질을 총리령으로 정하는 허용기준 및 식품위생법 등의 관계법령에 따라 잔류허용기준을 넘는지 여부를 조사하는 것이다. 식품의약품안전처장이나 시・도지사는 유통 또는 판매 중인 농산물 및 저장 중이거나 출하되어 거래되기 전의 수산물에 대하여 안전성조사[3]를 해야 하며 안전성 검사기관을 지정하여 시험분석 업무를 대행하고 있다.

1.1 어류품질안전성 검사의 중요성

2011년 동일본대지진으로 인한 방사능 오염수 바다 유입사건[1] 이후 후쿠시마(福島) 외 8개 해역에서 잡은 수산물은 수입이 금지되었고 국내 어류 안전성 조사에 대한 중요도가 높아졌다. 국내 안전성 조사 건수도 매해 증가하고 있으며

가장 최근 자료 2018년 총 조사 건수는 9284건에 이른다.

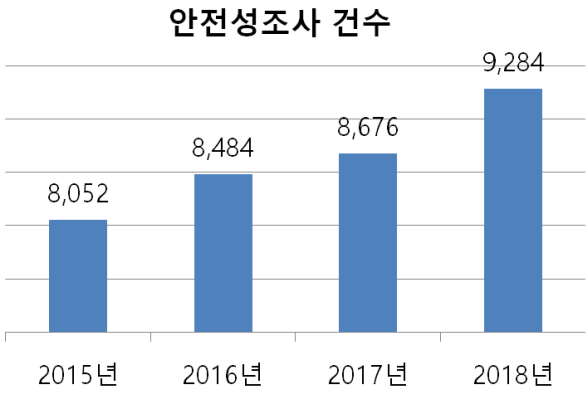


그림 1. 국립수산물품질관리원 안전성조사 건수

1.2 어류안전성 조사 법제 현황

인간의 건강에 큰 영향을 미치는 중금속, 항생물질, 식중독균, 방사능, 항생제 등을 어류 안전성 조사로 잔류 허용치를 조사하여 유통단계부터 제약을 한다. 본 절에서는 어류 안전성 조사의 대표적인 법제를 살펴본다.

어류안전성 조사는 농산물 등의 안전성조사 업무처리요령의 제1조(목적) 농수산물품질관리법이 지정되어 있고 그 세부항목으로 제61조와 행정규칙으로 구분하여 명시되어 있다.

1.2.1 농수산물 품질관리법 제61조

농수산물 품질관리법 제 61조에서 “안전성조사란 식품의약품안전처장이나 시·도지사는 농수산물의 안전관리를 위하여 농수산물 또는 농수산물의 생산에 이용·사용하는 농지·어장·용수(用水)·자재 등에 대하여 다음 각 호의 조사(이하 "안전성조사"라 한다)를 하여야 한다.”[2] 라고 명시 되어 있으며 시행 주체는 식품의약품안전처 및 지자체가 하게 되어 있다. 안전성 조사는 농산물과 수산물

로 구분되어 지며 수산물은 생산단계에서 총리령으로 정하는 안전기준에의 적합 여부를 조사해야 하고 저장단계 및 출하되어 거래되기 이전 단계에서는 식품위생법 등 관계법령에 따른 잔류허용기준 등의 초과 여부를 조사해야 한다고 명시되어있다.

1.2.2 수산물 안전성조사업무 처리요령

안전성 조사 처리요령은 각 생산 유통단계별로 세부 행위항목이 있으며 검사대상물의 시료수거항목, 시료수거방법, 시료분석방법 , 분석결과조치, 시료의 보관 및 폐기 방법 등의 항목이 있다. 안전성 조사의 행위로 안전성 검사의 행위가 있으며 안전성 검사는 각 지자체에서 시행을 하고 있다. 제주특별자치도는 수산물방역 및 안전성 검사에 관한 조례 및 시행규칙에 따라 제주특별자치도 해양수산연구원에서 위탁 검사를 시행하고 있다.

표 1-1. 안전성 검사표(제주특별자치도)

구 분	안전성 검사	안전성 조사
시행근거	제주특별자치도 수산물 방역 및 안전성 검사에 관한 조례 제4조	농수산물 품질관리법 제61조
시행주체	제주특별자치도	식품의약품안전처, 지자체
검사기관	도 해양수산연구원	국립수산물품질관리원 국립수산과학원
내 용	식용을 목적으로 양식수산물(활어 한정)을 도외로 반출하거나 도내에서 유통하기 위하여 생산자가 동물용의약품을 사용한 경우 출하 전 단계에서 안전성 검사	수산물의 안전관리를 위하여 수산물 또는 해당 수산물 생산에 이용되는 용수, 자재 등에 대하여 조사

표 1-2. 안전성 검사표(제주특별자치도)

구 분	안전성 검사	안전성 조사
검사시기	매 출하 전 단계	가. 생산단계 나. 저장단계 및 출하되어 거래되기 이전 단계
대상품목	광어(활어)	수산물(가공, 양식 포함)
검사항목	항생제	항생제, 중금속, 식중독균, 패류독소, 복어독, 말라카이트 그린 등
부적합 발생시 조치	출하정지(재검사 후 출하)	사용금지, 출하연기, 용도전환, 폐기명령
위반시 벌칙	과태료(100만원~500만원)	1년 이하의 징역 또는 1천만원 이하의 벌금

1.3 광어 안전성 검사 현황

제주특별자치도에 유통되는 수산물 중 가장 큰 비중을 차지하는 어류는 광어라 할 수 있다. 제주특별자치도는 광어의 안전성 검사기관을 제주특별자치도 해양수산연구원으로 위탁 지정하여 시행하고 있으며 제주도 광어가 생산, 유통되기 전에 필수적으로 안전성 검사를 받아야 한다.

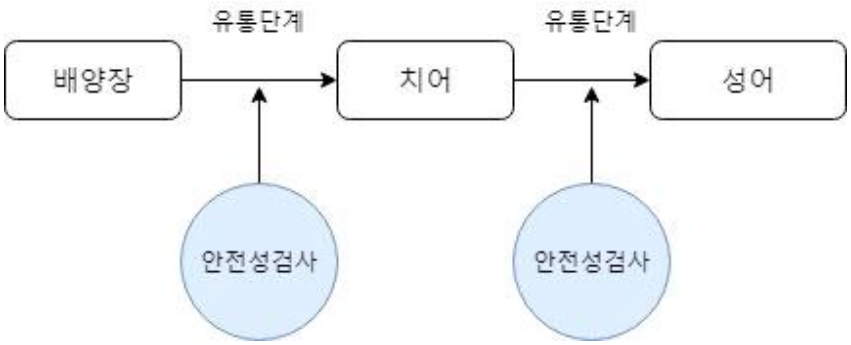


그림 2. 제주특별자치도 어류유통단계별 안전성 검사

광어를 생산하여 유통하고자 하는 생산자는 먼저 해양수산연구원에 어류안전

성 검사 신청을 해야 한다. 해양수산연구원에서 안전성 검사를 받은 후 검사결과를 제주도 어류양식수협에서 수령 및 확인 후 각 양어장에 검정증명서를 발급하고 검정증명서 결과의 적합판정이 나와야 양식장에서 광어를 유통 할 수 있다.

[그림 2]에서 보는 것과 같이 안전성검사는 배양장에서 치어양식장으로 유통되기 전에 안전성 검사를 받고 적합 판정을 받은 후 유통 할 수 있고, 치어 양식장에서 성어 양식장으로 유통하기 전에 안전성 검사를 받고 적합 판정을 받아야 유통 할 수 있다. 정리하면 각 유통단계마다 안전성 검사를 시행하고 있다는 것을 알 수 있다.

이 유통단계마다 사람이 간섭하게 되고 고의적 행위로 정보가 변경될 수 있으며 검사자의 실수로 정보의 오류가 발생할 수 있다. 잘못된 검사결과로 유통품질의 문제가 발생할 수 있으며 검사기관 및 유통업체의 신뢰도가 떨어질 수 있다. 이처럼 안전성 검사는 신뢰성 있게 이루어 져야 하며 검사기관, 승인기관, 생산자, 유통자에게 투명하게 제공 되어야 한다.

2. 어류품질안전성 검사정보 보안 위협 및 부인방지 요구사항

어류품질안전성 검사는 사람이 직접적 관여를 하므로 정보의 유출과 조작 또한 쉽게 발생 할 수 있다. 때문에 이에 대한 보호가 필요하다. 안전성 검사 정보 위협 사항을 정보 보안, 신뢰성 및 부인방지로 파악하고 본 문서에서 설명하고 있는 위협을 통하여 필요한 보안요구사항을 도출한다.

2.1 정보 보안 위협

매년 마다 버라이즌(verizon)은 데이터 유출 보안사고 조사 보고서를 발행하고 있다. “2019 Data Breach Investigations Report”에 자료를 보면 사람의 실수, 오용, 물리적 도난, 해킹, 악성코드 등으로 보안사고가 발생되며, 이중에서도 [그림 3]에 결과와 같이 해킹에 의한 보안사고가 가장 높다.

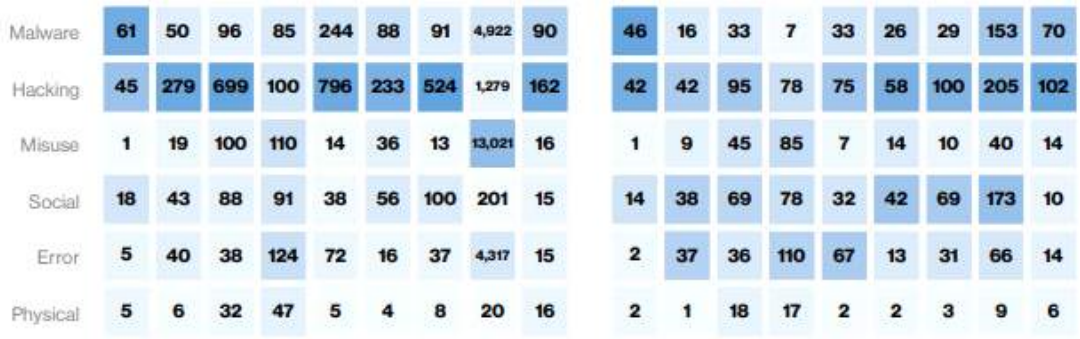


그림 3. 2019 Data Breach Investigations Report

검사 정보는 항상 보안 위협상황에 있으며 이와 같은 다양한 위협에 대해 안전성 검사의 과정, 기록, 열람, 유통품질의 측면에서 관련성 있는 보안위협을 정리하면 다음과 같다.

2.1.1 안전성 검사 과정

- 검사자 또는 승인기관의 직접 관리자가 고의로 정보가 변경될 수 있다.
- 검사자의 실수로 정보의 오류가 발생할 수 있다.

2.1.2 안전성 검사 기록

- 비 관리자에 의해 정보가 기록될 수 있다.
- 비 관리자에 의해 의도된 기록 변경 또는 관리자에 의한 의도치 않은 기록 변경이 발생할 수 있다.
- 해킹에 의해 정보가 위조 되거나 변조 될 수 있다.
- 내부관계자, 컴퓨터 관리자등 서로 다른 관계자들이 공모하여 정보를 위조 , 변조 하거나 유출을 시도 할 수 있다.

2.1.3 안전성 검사 열람

- 유통자의 생산품에 대한 안전성 검사는 해당 유통자에게는 중요한 정보이다. 생산품의 부적합 검사결과를 외부인 또는 내부 비 관리자에게 승인 없이 열람이 되어서는 안 된다.

• 승인 없는 열람으로 또 다른 유통자에게 검사결과 노출로 인해 유통자의 다른 생산품의 안전성 검사에 영향을 끼칠 수 있다.

2.2 안전성 검사 요구사항

위 2.1 정보 보안 위협에서 설명한 위협사항을 해결하기 위해서는 안전성 검사의 신뢰성 및 투명성, 무결성이 보장되어야 한다. 세부적으로 접근제어, 행동제어, 인증, 부인방지, 정보의 무결성, 정보의 투명성 등이 해결 되어야 한다.

표 2. 안전성 검사 요구사항

제목	설명
접근제어	허가된 사용자만 안전성 검사정보에 접근할 수 있어야 한다.
행동제어	허가된 사용자의 정보조작은 허가된 정책에 의해서 행해져야 한다.
인증	안전성 검사정보의 생성, 저장, 공개는 인증이 처리된 정당한 주체여야 한다.
부인방지	모든 주체의 행위는 기록되어 져야 한다.
정보의 무결성	저장, 열람된 안전성 검사정보는 악의적 또는 실수로 변경됐더라도 확인이 가능해야 한다.
정보의 투명성	안전성 검사정보는 정보와 관계가 되는 모든 주체에게 투명하게 이력이 공개 되야 한다.

3. 블록체인 기반 무결성 서비스 분석

블록체인은 암호화된 블록이 연결되면서 성장하는 블록목록이다. 각 블록은 이전 블록의 암호해시와 타임스탬프 및 거래데이터를 포함하여 블록이 생성된다. 이진트리에 트랜잭션정보는 즉 거래 데이터는 암호화 하여 저장되며 이 암호화된 전체 데이터를 암호화 하여 머클루트(Merkle Root)에 저장된다. 이 기술은 사토시 나카모토의 논문 “Bitcoin: A Peer to Peer Electronic Cash System”에서

처음 구현되었다[5]. 최초 비트코인을 위해 개발된 기술로 가상화폐의 거래를 위해 고안된 기술이지만 현재는 여러 가지 형태로 변형되어 다양한 환경에 적용되어 사용 되고 있다.

3.1 블록체인 구조 및 핵심요소

블록체인은 [그림 4]와 같이 크게는 해더와 트랜잭션으로 구분된다. 해더에는 현재블록의 생성 해시값과 이전블럭의 해시값, 그리고 트랜잭션의 머클루트(Merkle root) 해시값으로 구성된다. 타임스탬프 , 전자서명 등 부수적으로 값을 넣을 수 있다. 머클루트는 실제 거래가 이루어지는 트랜잭션 정보의 해시값이 저장된다. 최초 블록해쉬가 생성된 후 거래가 이루어지는 시점부터 블록체인이 형성되며 구성은 링크드리스트(Linkded List) 형태로 조립된다.

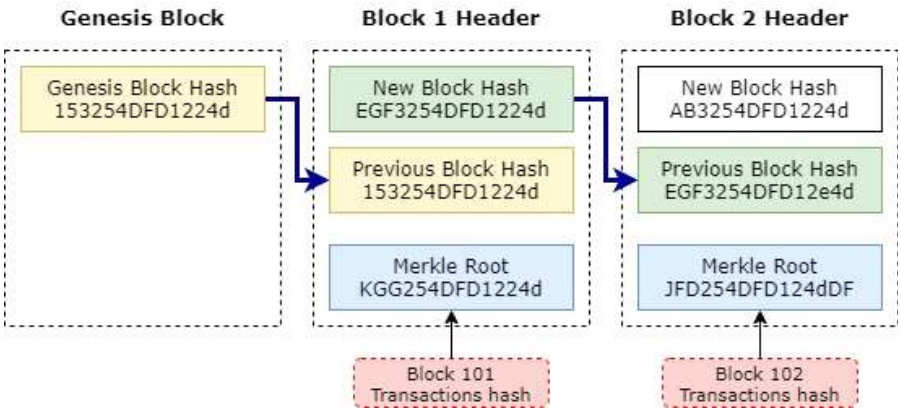


그림 4. 블록체인의 구조

머클 트리(Merkle Tree)는 트리 구조의 형태로 [그림 5]와 같은 구조를 갖는다. 최하위 자식 노드는 실제 거래가 이루어지는 정보 등의 데이터를 나타내며, 상위 노드들은 각각 자식 노드들의 해시 값을 나타낸다. 최상위 노드 해시값이 블록체인 해더의 머클루트 해시값이 된다. “머클 트리는 여러 블록들의 자료가 변조되지 않았음을 보장하는 용도로 사용 된다”고 발명자 랄프 머클은 설명하고 있다[5].

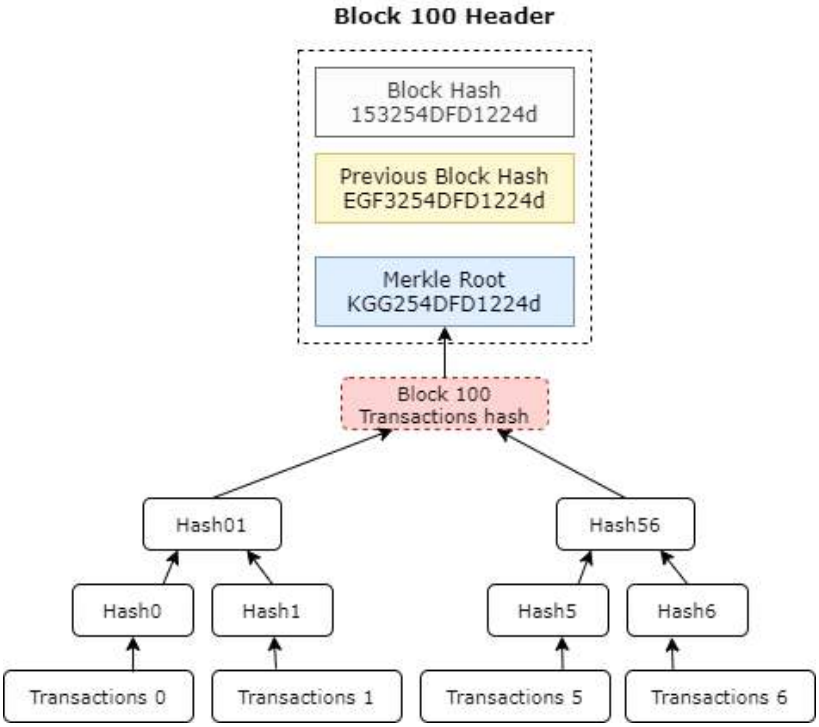


그림 5. Merkle Tree 의 구조

머클트리의 장점은 데이터를 검증하고자 할 때 루트 노드의 해시 값만 알면 데이터가 옳은 데이터인지 검증할 수 있다. 그리고 일부만 검증하고자 할 때에도 자식노드 가운데 하나의 해시 값만 알면 그 노드의 모든 자식노드에 대한 자료를 검증 할 수 있는 장점이 있다[6].

이러한 블록체인의 구조는 악의적인 의도로 데이터를 수정하는데 저항력을 가지고 있으며 부인방지가 가능하다. 이것은 거래의 구성원들이 공동으로 데이터를 기록・검증・저장하게 함으로써 무결성 및 신뢰성 또한 확보 할 수 있다.

블록체인에 기존 기술을 병합해서 새로운 플랫폼들이 만들어 지고 있다. 블록체인에 P2P 네트워크(peer-to-peer network), 합의 알고리즘(Consensus Algorithm), 전자서명(Digital Signature)과 스마트계약(Smart Contract) 기술 등이 통합 된 시스템이 블록체인 플랫폼이다.

3.1.1 P2P 네트워크(peer-to-peer network)

P2P 네트워크는 통신망에서 클라이언트나 서버라는 개념이 없이, 각각의 같은 통신망에 연결된 노드들이 서로 클라이언트와 서버역할을 동시에 서비스를 하게 된다. 보통 웹서버가 있고 웹사이트를 접속하는 클라이언트가 통상 우리가 경험하는 중앙집중식 네트워크이다. 반대로 P2P는 중앙집중식이 아닌 서로 상호적으로 연결되어 데이터를 주고 받는 분산형 네트워크이다. 국내에는 오래 전에 소리바다라는 MP3 음악 파일을 공유하는 플랫폼이 있었고 전형적인 P2P 네트워크 형이다. 최근에는 비트토렌토라는 자료 공유형 플랫폼이 P2P 네트워크라고 볼 수 있다.

3.1.2 합의 알고리즘(Consensus Algorithm)

합의 알고리즘(Consensus Algorithm)은 분산형 네트워크에 참여한 노드들의 자료들이 동일하게 분산 저장되기 위해 만들어졌다. 중앙서버가 없이 각 노드들 간의 자료교환이 이루어지는데 노드들 간에 통신두절, 지연으로 인한 자료의 진위정보 유지가 어렵다. 이런 상황에서 합의알고리즘은 블록의 일관성을 유지하게 해주며 현재는 작업증명, 지분증명 등 다양한 알고리즘이 개발되었다.

3.1.3 전자서명(Digital Signature)

전자서명(Digital Signature)은 블록체인 네트워크에 노드들의 신원을 증명하거나 거래의 위변조를 막기 위해 도입되었다. 암호화 방식 중 비대칭키 방식을 적용한 방법으로 공개키와 개인키 쌍으로 이루어져 있으며 단방향 암호화 , 대칭키 암호화 방식과 달리 기밀성, 인증, 무결성, 부인방지가 가능하다. 전자 서명은 전송자의 개인키로 서명하는 비대칭 방식을 사용하고 전송은 수신자의 공개키로 암호화하는 대칭키 방식을 사용하여 전송하는 방식이다. 수신자는 본인의 개인키로 해독하고 송신자의 개인키로 서명확인을 한다. 전자서명 방식은 블록체인 네

트워크에서 모든 거래 즉, 트랜잭션이 발생할 때마다 개인키를 생성하고 각 노드들의 거래가 암호화 되어서 거래가 이루어진다.

3.1.4 스마트계약(Smart Contract)

스마트계약(Smart Contract)은 1994년 닉자보[7]가 처음 제안한 대안으로 '이해당사자 간 공유 네트워크를 통하여 계약과 계약의 결과에 대한 신뢰를 쌓아 나갈 수 있는 기반 하에 확보된 자동화된 계약처리의 형태' 라고 제시했다. 블록체인 플랫폼에 스마트 계약을 저장하고 있으면 각 노드는 계약 조건이 만족할 경우에만 코드를 실행하고 새로운 거래 기록을 저장하게 한다. 금융거래, 부동산계약, 공증등 다양한 형태로 계약을 체결 이행할 수 있다.

위에 서술한 기술들이 통합된 블록체인 플랫폼이 최근에 상업적 모델로 많이 배포 되고 있다. 그 유형도 다양하게 변화 되고 있는 추세이다.

3.2 블록체인 유형

블록체인의 유형은 퍼블릭 블록체인(Public Blockchain)과 프라이빗 블록체인(Private Blockchain) 그리고 컨소시엄 블록체인(Consortium Blockchain)[8]으로 구분된다. 이러한 구분은 참여자가 승인되어 노드에 참여할 수 있는지 없는지 같은 목적을 가지고 여러 기관이 함께 네트워크에 참여하는지에 따라 구분된다.

퍼블릭블록체인(Public Blockchain)은 승인 없이 누구든 쉽게 참여 할 수 있는 공개형 블록체인이다. 비트코인과 이더리움이 정형적인 퍼블릭 블록체인이며 불특정다수가 쉽게 참여할 수 있으므로 노드가 많아지면 네트워크 지연이 발생하고 그로 인해 거래가 지연되는 단점이 있다.

프라이빗블록체인(Private Blockchain)은 노드에 참여하는 사용자들을 제한하여 독립된 기관에서 독자적으로 운영하는 블록체인이다. 금융기관용 블록체인 개발 업체 클리어마틱스(Clearmatics)의 대표이사 로버트샘스(Robert Sams)가 처

음 고안했다. 인트라넷으로 독자적으로 운영하는 블록체인으로 1개의 주체가 블록체인을 관리한다.

표 3. 블록체인의 종류

구분	개념 및 기능	전제조건
Public Blockchain	- 인터넷을 통해 모든 사람이 열 수 있는 거래장부 - 네트워크 확장이 어렵고 거래가 느리다	-안정적인 생태계가 필요 - 위기 관리가 필요
Private Blockchain	- 개인화 된 블록체인 - 하나의 개체가 블록체인으로 내부 네트워크를 관리한다.	- 시스템 변경 및 안전 보장 - 한 단체의 글로벌 지사 형태
Consortium Blockchain	- 미리 선택된 개체만 참가할 수 있다. - 개체간에 합의 된 규칙을 통한 공증 된 참여 - 쉬운 네트워크 확장 및 빠른 트랜잭션 속도	- 참가자 간의 비즈니스 계약 - 시스템 안전 보장

컨소시엄블록체인(Consortium Blockchain)은 같은 목적을 가지고 있는 다수의 기관이 하나의 컨소시엄을 구성하여 각 기관 간에 데이터를 공유하며 투명성과 확장성을 보완하기 위해 운영하는 블록체인이다. 반중앙형 블록체인으로 다수의 사전에 선정된 주체만 참여 가능하다. 컨소시엄 블록체인은 다수 참여자의 협의가 필요한 분야에서는 프라이빗 블록체인 보다 효과적이다. 본 연구에서는 검사기관, 승인기관, 생산자, 유통자간의 블록체인을 운영하기 위해 컨소시엄 블록체인으로 구성할 것이다.

블록체인의 인기로 현존 하는 블록체인 플랫폼은 Bitcoin, Ethereum, Hyperledger, R3 Corda, Nexledger 등 여러 가지가 오픈소스 형태로 배포 되고 있다. 이 플랫폼 중 최근 많이 인기를 얻고 있는 하이퍼레저(Hyperledger)를 선택해서 개발한다.

3.3 블록체인 기반 무결성 서비스 사례분석

블록체인 기술은 다양한 분야에서 활발하게 진행되고 있는데 정보의 위변조 방지와 정보추적이 가능하고 신뢰성이 보장되기 때문이다. 현재는 금융, 의료, 물류 뿐만 아니라 IOT까지 그 범위를 가늠하기 어려울 정도로 블록체인 적용 사례가 늘어나고 있다.

식품 유통 관련 블록체인 사례는 중국 월마트(Walmart)가 도입한 돼지고기식품안전성 위변조 방지 블록체인이 있다. 중국 최대 규모의 가공 돈육제조사의 불법사료 첨가물을 사용한 제품 오염이 발생하고, 매장에서 가짜 돼지고기 판매, 유통기한이 지난 오리고기 판매가 적발되었다. 돼지고기 유통 사기들이 많이 발생함으로써 정부의 식품안전시스템 부재가 지적되고 소비자들을 불안하게 하였다[9].

이에 중국 월마트는 IBM과 함께 돼지고기식품안전성 위변조 방지 블록체인을 적용하여 식품 공급 망 이력을 추적하였다. 돼지고기의 생산지에서 월마트의 진열대까지 전체 공급 과정 정보를 추적하여 품질 관리가 용이해졌다. 돼지고기가 유통될 때마다 생산지 정보, 도축 및 가공 정보, 운송정보 등을 블록체인에 저장함으로서 위변조 방지가 가능했다.

교육 관련 블록체인 사례는 MIT는 대학 최초로 111명의 졸업생들에게 전통적인 졸업장 이외에도 블록체인 기반의 졸업학위를 앱을 통해 수여 받을 수 있는 시범 프로그램을 진행했다[10][11]. 이 앱은 Blockcerts Wallet으로 학생들이 검증 가능한 변조 방지 버전을 고용주, 학교, 가족 및 친구들과 공유할 수 있도록 하였다.

IT 시스템 관련 블록체인 사례는 도메인 네임 시스템(Domain Name System)으로 보안 측면에서의 취약점인 위변조를 블록체인을 도입하여 방지하는 것이다. 위변조의 대표적인 사례는 1997년 미국에서 일어난 공격으로 당시의 도메인 관리의 최상위 기관인 InterNIC(TheInternet's Network Information Center, 이하

InterNIC)의 웹 사이트에 대한 접속 트래픽을 제3의 다른 웹 사이트로 전환되도록 DNS캐시 포이즈닝을 이용해 공격한 사건이다[12]. 이것은 공격자가 DNS정보를 위변조하여 제3의 웹 사이트로 접속자를 가로 챌 수 있다는 것을 보여준다. DNS 레코드를 블록체인 기술을 적용하면 DNS 캐시 위변조를 방어 할 수 있다.

해외국가들의 블록체인을 도입해서 활용하려는 동향을 정리해 보면 다음과 같다.

표 4-1. 해외 국가들의 식품분야 블록체인 도입 동향[9]

구분	주요 내용
연방과학산업연구소 (호주)	연방과학산업연구소(CSIRO)는 미래 농식품산업 성장을 위한 식품이력추적 관련 주요 기술로 '디지털 추적(RFID 칩, 바코드, QR코드, 블록체인)'을 핵심지원 대상기술을 담은 '식품 및 농산업 로드맵' 발표('17)
와게닝헨 대학 등 (네덜란드)	네덜란드 와게닝헨 대학(WUR)의 농식품을 위한 블록체인 프로젝트는 남아프리카 식용 포도 유통과정에 블록체인 기술을 적용하여 해당 기술이 농식품에 미치는 영향, 기술 적용에 필요한 사항 등 도출
IBM 등 (미국)	IBM은 식품안전 강화에 블록체인을 활용하는 프로젝트 착수하여 주요 식품업체가 데이터 공유 및 시범 운영 참여하여 식품의 생산, 유통 전 과정에 블록체인을 적용하여 투명성, 신뢰성 확보 계획
징동닷컴 (중국)	징동닷컴은 호주 육류도매업체InterAgri(사)와 협력 맺고 중국에 수입하는 해당 육류제품의 전 과정을 블록체인 플랫폼을 통해 추적하여 호주에서 수입된 것인지를 증명할 수 있는 진품 확인체제를 갖춰 수입 제품 품질에 대한 중국 소비자의 신뢰도 향상

표 4-2. 해외 국가들의 식품분야 블록체인 도입 동향[9](계속)

구분	주요 내용
알리바바 등(중국)	중국 최대 전자상거래 기업인 알리바바는 블록체인을 활용한 '푸드 트러스트 프레임워크(Food Trust Framework)'를 만든다는 계획을 세워 블록체인 기술이 구매자와 판매자 사이에 훨씬 투명하고, 보안성이 높고, 사기 위험을 낮춘 방법으로 실시간 유통 경로를 파악할 수 있음
까르푸 (프랑스)	유럽에서 최초로 식품 블록체인을 도입한 까르푸는 방사 사육 닭의 생산을 추적할 수 있는 블록체인 기술을 이미 도입하고 있고 2018년 말까지 달걀, 치즈, 우유, 오렌지, 토마토, 연어, 햄버거 등 신선도에 민감한 제품에 블록체인 기반의 식품이력추적제 확대 도입 계획 발표('18.3.6)

이처럼 블록체인은 각국에서 IT 기업들과 정부 주도로 여러 분야에 활용하는 시범사업 또는 중점사업으로 추진하고 있다.

III. 블록체인 기반 어류품질검사 데이터의 무결성 서비스 분석

자기정보통제 및 데이터 무결성 측면에서 많은 장점을 가지고 있는 블록체인을 어류품질안전성 검사 정보에 구현함으로써 위 요구사항을 해결할 수 있다. 각 기관과 유통자들의 상호 감시체제를 형성하여 악의적인 정보 또는 검사자 실수로 발생할 수 있는 허위 검사결과 발생을 방지하여 기관과 품질의 신뢰도를 높이기 위한 방안을 제안한다.

1. 블록체인 기반 어류품질검사 데이터의 무결성 서비스 도입 방안

본 문제를 해결하기 위한 방법으로, 검사기관, 승인기관, 유통자(판매자, 구매자)를 대상으로 한다. [그림 6] 어류품질 안전성 검사 흐름도를 보면 판매자가 안전성 검사 요청을 한 후 검사기관은 안전성 검사를 시행하고 정보를 블록체인에 등록한다. 이후 검사정보는 승인기관의 인증을 거쳐 블록체인에 승인정보가 등록된다. 유통 희망자는 유통 전에 유통물의 검사결과를 확인하고 구매자 역시 유통물의 검사결과를 확인 할 수 있다.

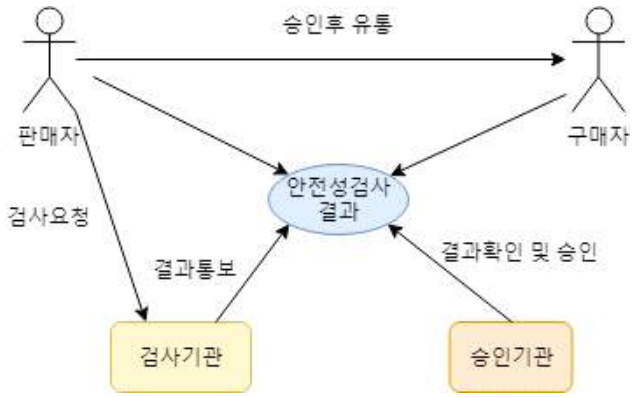


그림 6. 어류품질 안전성 검사 흐름도

이러한 기능을 통해 안전성 검사 정보의 신뢰성과 유통어류의 거래까지 신뢰성을 보장 할 수 있다. 그리고 거래된 기록은 접수자의 인증을 통해 어류품질검사 블록체인 장부에서 공개되고 접수자의 허가로 판매자가 볼 수 있으므로 해당 블록의 보안성도 안전하다. 이를 통해 유통자가 어류검사결과를 확인하는 과정의 번거로움이 줄어들고, 이 과정에서 발생하던 어류품질관리의 신뢰성도 높아질 수 있다.

2. 블록체인 기반 어류품질검사 데이터의 무결성 서비스 검사 과정

판매자가 최초 유통을 하기위해 검사기관에 검사요청을 한다. 검사기관의 검사자가 안전성 검사를 실시 후 검사정보를 어류품질검사 블록체인에 등록한다.

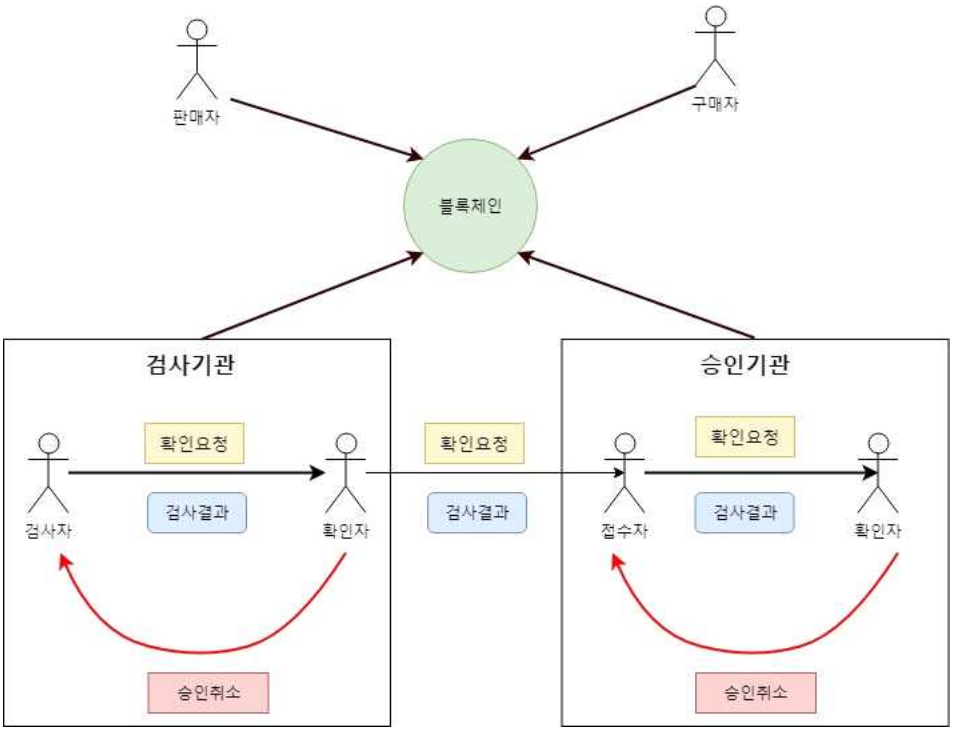


그림 7. 어류품질검사 블록체인 검사 과정

등록된 검사결과는 검사기관의 선정된 관리자의 스마트컨트렉트가 완료된 후 승

인기관에게 승인요청이 발생하고, 승인기관 역시 선정된 관리자의 의해 승인결과가 완료된 후 접수자에게 최종 블록정보를 전송하게 된다.

각 기관별로 안전성검사정보가 전자서명 한 후 전송되고 스마트컨트렉트로 승인되며, 이 블록정보는 그림[8]의 Sequence Diagram에 보이는 블록생성 순서로 지속적으로 쌓인다. 최종 판매자에게 검사결과가 통보되기 전에 검사기관, 승인기관의 결과확인 단계가 있으므로 허위정보 또는 검사자 실수가 발생하더라도 최종 통보에서 무결성을 보장할 수 있으며 부인방지가 가능하며 이를 이용하여 안전성 검사 정보의 신뢰도를 평가할 수 있다.

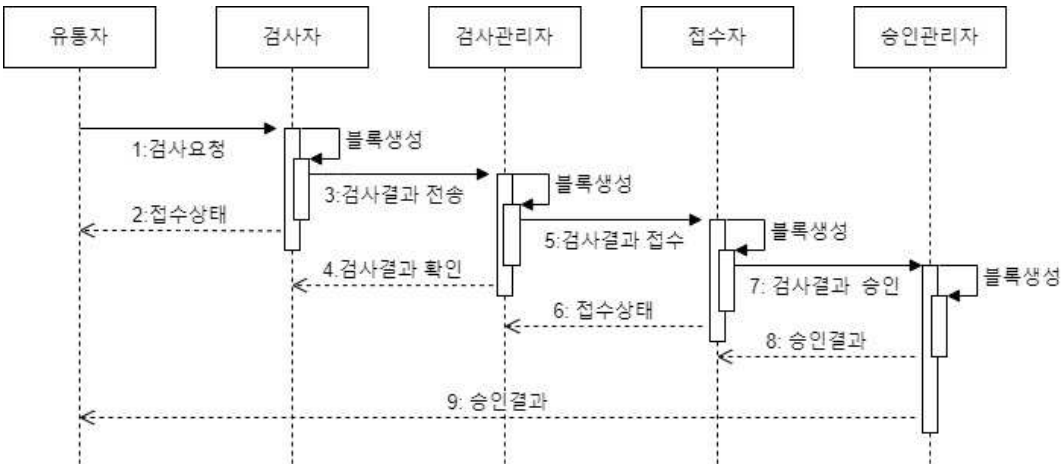


그림 8. 어류품질검사 블록체인 시퀀스 다이어그램

유통 희망자는 유통 전에 유통물의 검사결과를 확인하고 구매자 역시 판매자의 공유수락으로 유통물의 검사결과를 확인 할 수 있다. 거래된 기록은 판매자의 인증을 통해 블록체인 장부에서 공개되고 판매자의 허가로 구매자가 볼 수 있으므로 해당 블록의 보안성도 안전하다. 이를 통해 유통자가 어류품질검사결과를 확인하는 과정의 번거로움이 줄어들고, 유통자간의 상호 신뢰성도 높아질 수 있다.

2.1 어류품질검사 블록체인 조직 구성 정보

조직구성은 위 [그림 6, 7]에 구성인 검사기관, 승인기관, 유통자 그룹으로 구성된다. 검사기관에는 여러 검사자와 관리자가 존재하고 승인기관 역시 여러 구성원이 있다. 유통자는 판매자 또는 구매자이며 구매자는 판매자가 될 수도 있고 유통자가 될 수도 있다. 여기서는 조직그룹을 검사기관, 승인기관, 유통자를 서비스 참여자로 정하였다. 각 Member는 개별적으로 유효한 Identity를 가지고 노드를 통하여 거래가 이루어진다.

표 5. 제안된 어류품질검사 블록체인 조직구성정보

조직	멤버	설명
검사기관	검사자	안전성 검사를 실시 후 관리자에게 제출
	관리자	안전성 검사 정보를 승인 후 승인기관에 제출
승인기관	접수자	안전성 검사를 접수 승인여부를 관리자에게 요청
	관리자	안전성 검사를 승인 후 판매자에게 제출
유통자	판매자	안전성 검사를 요청 및 조회
	구매자	안전성검사를 요청 및 조회

블록체인 기반 어류품질검사 데이터의 무결성 서비스에서의 조직구성은 제주특별자치도에서 시행되는 광어 안전성 검사를 예로 들면 유통자는 제주도 내의 양식장이며 이 판매자는 치어 양식장에서 성어양식장으로 판매가 이루어진다. 이 유통단계에서 양식장의 어류 안전성 검사 정보를 조회하거나 자신이 소유한 어류의 안전성 검사정보를 요청한 구매자에게 안전성 검사를 공유하도록 허가 할 수 있는 인가자의 역할을 한다.

검사기관은 어류안전성 검사를 실시하는 기관으로 제주특별자치도에서는 해양수산연구원이 위탁대행 하고 있다. 검사기관은 안전성검사정보 제공자로서 양식장의 시료자료를 토대로 검사하는 검사자 역할과 검사기록을 검수 및 관리하는 관리자의 역할을 한다.

승인기관은 검사기관에서 실시한 검사결과를 접수 받는 접수자로서 제주특별

자치도에서는 어류양식수협에서 역할을 담당하고 있다. 승인기관은 검사결과를 토대로 유통의 승인 , 유통의 취소, 폐기 처리 등 결과를 판매자에게 제공하고 유통의 직접적 관여를 한다.

2.2 어류품질검사 블록체인 시료 정보

어류품질검사 블록체인에서 실제 트랜잭션이 이루어지는 정보는 안전성 검사에 필요한 시료정보와 그 시료를 통해 실시된 안전성 검사 결과 정보이다. 이 정보는 제주특별자치도 어류양식수협에서 제주도해양수산연구원과 교환하는 정보를 토대로 작성이 됐다. 안전성 검사결과는 시료정보와 같이 제공되고 있으며, 이 정보를 기준으로 어류품질검사 블록체인 트랜잭션 정보로 이용한다.

[표 6]에 시료미수는 3미 또는 5미를 채취하며 양식장의 어류상태에 따라 3미를 채취 할 경우는 시료중량 650g, 670g, 800g을 채취하며 5미를 채취 할 경우에는 240g, 360g, 400g, 350g, 420g을 채취한다. 검사항목은 여러 항목이 있으나 본 연구에서는 항생물질 하나의 값만을 가지고 구현한다. 검사단계는 채취가 이루어지고 검사기관의 검사가 시작되면 진행 중이며 검사완료 후 검사정보가 승인기관으로 제출이 된 후 완료 처리가 된다. 수조번호는 시료가 채취된 양식장 수조의 번호이다. 시료는 한 수조에서만 채취되지 않고 각 1미 마다 각각 다른 수조에서 채취를 한다. 검사결과는 정성검사법, 정량 검사법이 있으며 정성검사법은 양성, 음성만 판단하는 것이며 정량검사법은 기준치를 초과했는지 초과되지 않았는지를 검사하는 방식이다.

표 6. 어류품질검사 블록체인 시료정보

데이터	키	설명
시료미수	sampling_cnt	3~5미
시료중량	sampling_weight	240~800g
검사항목	inspection_item	항생물질
검사단계	inspection_state	진행중,완료
수조번호	fishbowl_number	C1, C2, C3

검사결과	inspection_resullt	양성, 음성
유효기간	validity_date	2020.01.01(한달)
접수자식별아이디	owner	A0001
접수일자	apply_date	2020-01-01

본 연구에서는 정성검사법인 양성, 음성만을 판별하는 것으로 양성과 음성의 값을 넣는다. 안전성 검사 결과의 유효기간은 시료 채취 기준일로 30일이며 유효기간이 지난 안전성 검사는 검사결과에 유효기간 만료로 처리된다. 접수자 식별아이디는 어류품질검사 블록체인에 여러 정보들 중 실제 소유자의 고유 식별 정보가 등록된다. 접수일자는 시료 채취 날짜가 등록된다.

2.3 어류품질검사 블록체인

위 [표 6]에 안전성 검사시료정보는 [그림 4] 블록체인의 구조에 트랜잭션에 거래 형태에 따라 생성된다. 유통자가 어류를 유통하기 위해 API 서버를 통해 안전성 검사 요청을 하면 [그림 4] 최초 블록 1이 생성되고 검사기관에서 시료를 채취해서 시료정보를 등록함으로써 트랜잭션이 시작된다. 이때 유통자의 확인으로 거래가 이루어진다.

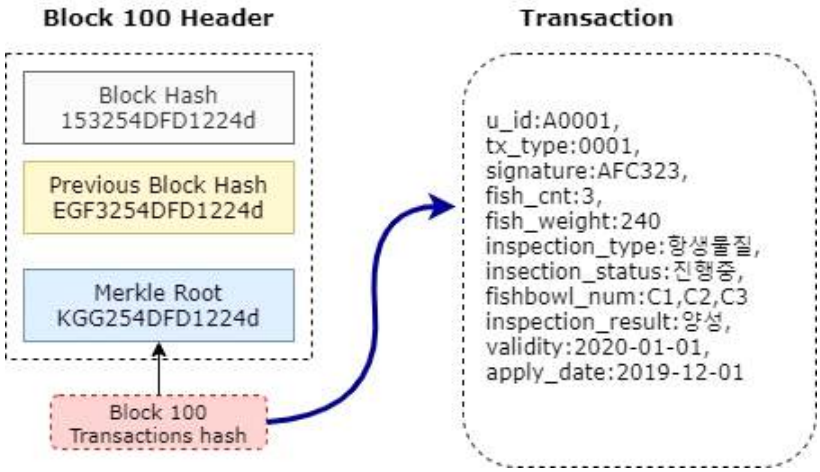


그림 9. 어류품질검사 블록체인 트랜잭션 정보

어류품질검사 블록체인 정보는 [그림 9] 어류품질검사 블록체인 트랜잭션 정보와 같이 Key, Value 형태를 취하며 Json 타입으로 생성된다. 좀 더 면밀히 살펴보면 검사기관의 검사자가 검사결과를 관리자에게 제출하고 관리자의 확인으로 다시 거래가 이루어지면서 계속적으로 블록이 생성되는데 이 때 새로운 값이 블록으로 생성되고 저장된다. 검사기관이 확인을 완료한 후 검사정보를 승인기관으로 제출하게 되고 승인기관은 제출된 안전성 검사를 접수 처리하는 과정에서 다시 블록이 생성된다. 이렇게 순차적으로 검사정보가 이동하는 동안 트랜잭션의 정보 상태는 변화되고 그 변화되는 정보들이 순차적으로 어류품질검사 블록체인으로 형성된다.

2.4 어류품질검사 블록체인 분산 합의 알고리즘

각 정보의 전달과정 중에는 분산 합의 알고리즘이 내부적으로 실행되고 있다. 이더리움의 포스(POS)나 이오에스(EOS)의 이포스(dPos)에 사용되는 비잔티합의 검증과는 다르게 하이퍼레저 패브릭의 합의 알고리즘은 Kafka 알고리즘[1]을 사용하고 있다. 블록체인에서의 합의 과정은 모든 참여자가 피어 역할을 하면서 원장의 일치성을 확보한다.

[그림 10] 어류품질검사 블록체인 네트워크 구조를 보면 유통자가 API 서버를 통해 검사를 요청 하면 어류품질검사 블록체인 네트워크의 각 노드들이 상호작용으로 합의 알고리즘을 형성한다. 피어들은 네트워크에 참여한 모든 기관 참가자들이다. 각 참가자들은 노드를 통해 서로 작용하고 있고 상황에 따라 보증노드(Endorsing Node)가 될 수도 있고 안 될 수도 있다. 보증노드는 안전성 검사정보를 전송하고자 하는 기기가 등록된 디바이스인지, 유통자는 올바른 정당한 유통자인지 확인한 뒤 트랜잭션을 허용해 주는 보증 역할을 수행한다. 여기서 검사요청을 한 유통자의 제안을 각 보증노드들이 Kafka 알고리즘 검증으로 보증이 완료된 후에 API 서버는 이 트랜잭션을 오더러 서비스로 제출 할 수 있다.

1) kafka는 아파치 소프트웨어 재단이 스칼라로 개발한 오픈 소스 메시지 브로커 프로젝트로 실시간 데이터를 통일성, 높은 처리량, 낮은 지연시간으로 처리 관리하기 위한 플랫폼을 제공한다.

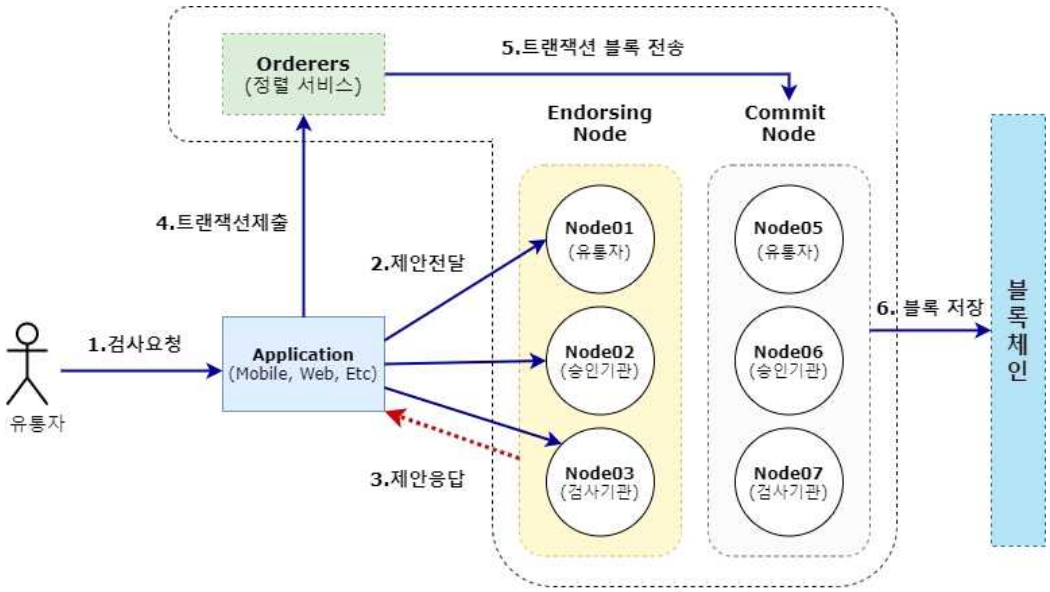


그림 10. 어류품질검사 블록체인 네트워크 구조

오더러 서비스는 어류품질검사 블록체인 네트워크에 제출된 트랜잭션정보를 받아서 추가 또는 삭제되는 정보들을 순서에 맞게 정렬하여 블록체인을 형성한다. 실제 오더러 서비스는 체인을 형성하는 역할만 하며 저장은 하지 않는다. 오더러 서비스에서 정렬을 마친 후 보증노드를 포함해서 모든 노드들에게 커밋(Commit)을 전달하고 이 커밋을 받은 노드들은 트랜잭션에 대한 검증과 인증서를 검증하는 작업을 수행한 후 문제가 없을 시 어류품질검사 블록체인을 분산 저장하게 된다.

좀 더 면밀히 살펴보면 [그림 11] 어류품질안정성검사 블록체인 네트워크 시퀀스다이어그램을 보자. 유통자 즉 사용자 애플리케이션이 거래를 위한 트랜잭션이 발생한다. 발생한 트랜잭션은 어류품질 안전성검사 블록체인 네트워크에 있는 사전 정의된 보증노드들에게 트랜잭션이 전송되고 이 다수의 보증노드들이 트랜잭션의 일치성을 검증하여 트랜잭션 보증을 한다. 보증이 완료된 트랜잭션은 다시 사용자에게 리턴 됨과 동시에 참여노드들에게 브로드캐스팅 되면서 정렬서비스 전송된다. 일반적으로 사용자가 하나의 트랜잭션을 요청한 경우 하나의 리스펀스(Response)값을 요청자에게 리턴 해 주는데 어류품질검사 블록체인 네트워크는 여기서 단일 응답을 전송하지 않고 이벤트 함수를 사용하여 전체 참여노드들에게 응답정보를 브로드캐스팅 한다.

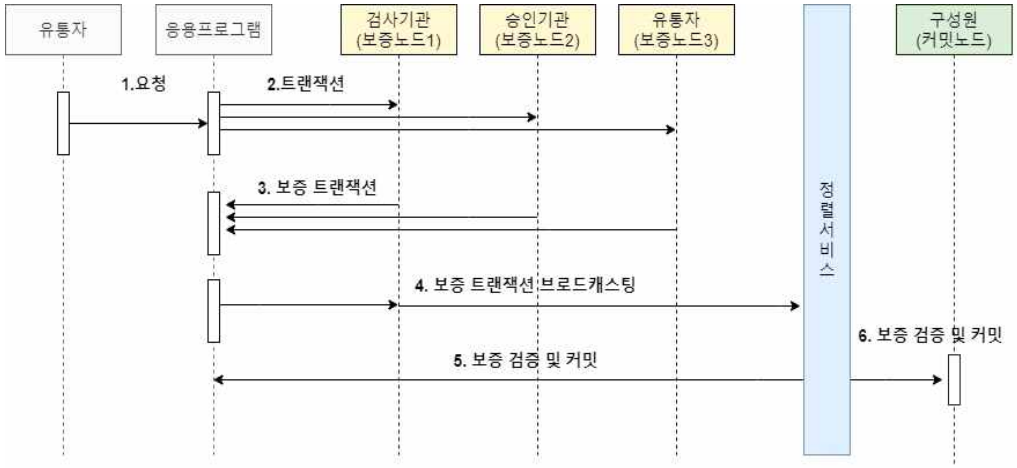


그림 11. 어류품질검사 블록체인 네트워크 시퀀스 다이어그램

이 브로드캐스팅된 정보는 모든 노드들이 정보를 수집하게 되고 정렬서비스노드 역시 이 정보를 수집하여 재정렬 및 검증과정을 다시 한 번 더 처리한다. 정렬서비스에서 검증과정을 마친 후 참여노드 모두에게 정보의 무결성이 없으면 커밋 처리가 가능하도록 참여노드들에게 허락해 준다.

IV. 설계 및 구현

본 논문에서 개발한 블록체인기반 어류품질검사 무결성 서비스는 어류품질검사 블록체인 네트워크 시스템과 사용자용 애플리케이션시스템으로 구성된다. 하이퍼레저 패브릭 플랫폼 서비스는 여러 가지 모듈이 통합된 플랫폼으로서 다양한 방법으로 분산 적재할 수 있도록 설계되었다. 또한 필요한 모듈만 빼내어서 다른 서비스에 적용 할 수 있는 이식성이 높다. 본 구현은 버츄얼박스(VirtualBox)에 가상화 서버를 구축하여 필수구성 요소인 어류품질검사 블록체인 네트워크를 구축하고 그 안에 독립적으로 사용자용 애플리케이션 서버를 구축한다. 어류품질검사 블록체인 네트워크 구축에 필요한 체인코드(chaincode)는 자체 플랫폼에서 지원하는 고(Go)언어를 사용한다. 애플리케이션 서버는 사용자에 뷰(View)를 제공해주는 프론트엔드(Frontend) 와 기능을 처리해주는 백엔드(Backend)로 구성된다. 언어는 자바스크립트(Javascript)로 구현하며 사용자가 블록체인 네트워크에 접근할 수 있는 웹서비스로 구현 한다. 기능은 유통자의 검사정보 조회 기능, 검사정보 추가 기능의 프로토타입으로 구현한다.

1. 블록체인 기반 어류품질검사 데이터의 무결성 서비스 설계

하이퍼레저 패브릭은 프레임워크로 패키지 형태로 체인 코드 기능이 탑재 되어 있어 기본 내장된 체인코드를 이용하여 다양한 설계를 할 수 있다. 그리고 패브릭 네트워크에 외부에서 접근할 수 있도록 SDK (Software Development Kit)를 제공하고 있어 이 SDK를 사용하여 어류품질검사 블록체인 네트워크 및 애플리케이션 서버를 구축 할 수 있다.

어류품질검사 블록체인 네트워크 서비스에 접근을 하기 위해 REST API를 이용해야 한다. 아직 공식적으로 하이퍼레저 패브릭에서 REST API 를 사용할 수

없다.(2020.05.24) 제공된 SDK를 사용하여 API 서버를 구현한다.

본 연구에서는 어류품질검사 블록체인 네트워크 서버, API 애플리케이션 서버 구성이 필요하다

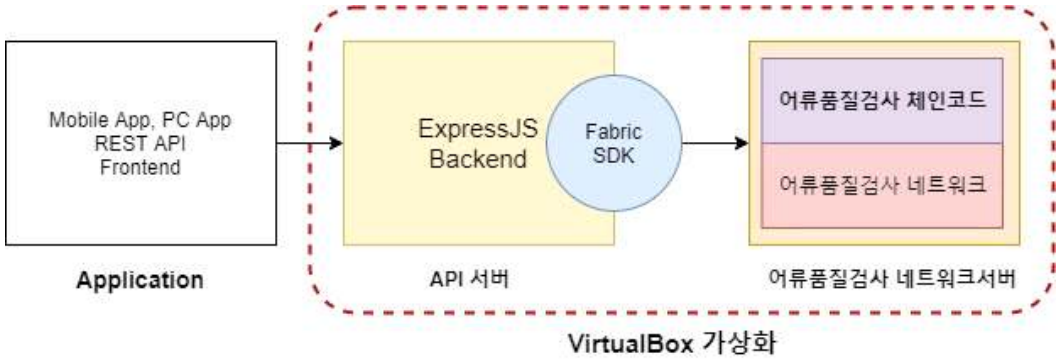


그림 12. 가상화 기반 어류품질검사 네트워크 구조

[그림 12] API 서버, 어류품질검사 네트워크 서버 가상화 구조를 보면 최종 사용자가 API 서버에게 쿼리로 요청한다. 이 쿼리를 받은 API 서버, 여기서는 ExpressJS 로 구축된 API 서버이다. 이 ExpressJS API 서버는 사용자로 부터 받은 요청을 상황에 맞춰서 어류품질검사 블록체인 네트워크로 보내어 처리한다. API 서버는 플랫폼에서 제공되는 SDK를 이용하여 자바스크립트로 구현한다.

1.1 하이퍼레저 패브릭을 이용한 어류품질검사 블록체인 네트워크 설계

어류품질검사 블록체인 네트워크를 구축하기 위해서는 크립토 환경설정, 트랙잭션 환경설정, 도커컴포저 환경설정을 해야 한다. 먼저 크립토 환경설정을 작성한다. [표 7] 제안된 크립토 환경설정 구성요소는 블록체인 기반 어류품질검사 데이터의 무결성 서비스 분석에서 확인 된 [표 5] 어류품질검사 블록체인 조직구성정보를 기준으로 승인기관, 검사기관, 유통자의 정보를 토대로 작성된 크립토 환경설정 구성요소이다. [표 7]의 네임(Name)은 각 노드를 식별하기 위한 이름이

며 도메인(Domain)은 도커 생성 시 식별하기 식별자이다. 템블릿카운트(Template Count)는 도커에 생성할 노드의 숫자이며 어류품질검사 블록체인 네트워크는 각 조직을 2개의 노드를 생성한다. 유저카운트(Users Count)는 각 조직의 생성할 사용자 수이다. [표 5] 어류품질검사 블록체인 조직구성정보에서 분석한 승인기관에 접수자, 관리자가 검사기관에 검사자, 관리자 유통자에 판매자, 구매자가 유저카운트에 필요한 구성사용자가 된다.

표 7. 제안된 크립토 환경설정 구성요소

구분	승인기관	검사기과	유통자
Name	OrgApprover	OrgInspector	OrgTrader
Domain	orgApporver.example.com	orgInspectoer.example.com	orgTrader.example.com
Template Count	2	2	2
Users Count	2	2	2

[표 8] 제안된 트랙잭션 환경설정 구성요소는 어류품질검사 블록체인 네트워크 채널(channel)에 참여하는 구성 피어들을 선언하고 하이퍼레저 패브릭에서 기본 제공하는 멤버쉽서비스(MSP)를 조직에 맞게 설정한다. 그리고 제네시스블록(genesis block)을 생성하도록 설정하고 또한 이 후 설정할 도커컴포저 환경설정을 통해 생성되는 도커노드들과 인증서를 교환 할 수 있도록 인증위치정보를 설정한다. 정렬서비스인 오더러서비스 형태를 싱글모드인 솔로(solo)로 할지 카프카 알고리즘을 사용할지 설정할 수 있다. 트랙잭션의 일괄처리 시간과 일괄처리 크기 등을 설정할 수 있다. 어류품질검사 블록체인 네트워크는 오더러 서비스를 기본값을 사용하여 적용한다.

표 8. 제안된 트랜잭션 환경설정 구성요소

구분	승인기관	검사기과	유통자
Name	OrgApprover	OrgInspector	OrgTrader
MSP Name	OrgApproverMSP	OrgInspectorMSP	OrgTraderMSP
MSP ID	OrgApproverMSP	OrgInspectorMSP	OrgTraderMSP
AnchorPeers	peer0.orgApprover .example.com	peer0.orgInspector .example.com	peer0.orgTrader. example.com
Port	7051	9051	10051

다음은 도커컴포져 환경설정 구성요소를 분석한다. [표 9]와 [표 10]은 어류품질검사 블록체인 도커 이미지를 생성하기 위한 도커컴포저 환경설정 구성정보이다. 도커펌포저 환경설정은 오더러서비스와 노드서비스를 구성한다. [표 9]는 오더러서비스 구성정보이다. 구성정보는 키와 값의 형태로 설정되며 서비스(services)키는 생성할 도커이미지 식별자 이름을 선언하고 각 서비스키의 값으로 콘테이너이름(container_name), 볼륨, 포트를 정의한다. 노드서비스 구성정보는 서비스키, 콘테이너이름, 볼륨, 포트는 오더러서비스와 같은 키를 가지고 있으며 추가적으로 환경(environment)키가 더 존재한다.

표 9. 제안된 오더러서비스 도커컴포저 환경설정 구성정보

구분	설정값
services	orderer.example.com
container_name	orderer.example.com
orderer volumes	../channel-artifacts/genesis.block:/var/hyperledger/orderer /orderer.genesis.block ../crypto-config/ordererOrganizations/example.com/ordere rs/orderer.example.com/msp:/var/hyperledger/orderer/msp ../crypto-config/ordererOrganizations/example.com/ordere rs/orderer.example.com/tls/:/var/hyperledger/orderer/tls orderer.example.com:/var/hyperledger/production/orderer
orderer ports	7050:7050

볼륨키는 배열형태의 정보로 배열정보의 첫 번째 값은 [그림 15]의 트랜잭션 환경설정을 통해 생성된 제네시스블록을 도커이미지에 복사 하는 역할을 한다. 두 번째 값은 [그림13]크립토 환경설정을 통해 생성한 멤버쉽서비스를 도커미이지에 복사 하는 역할을 한다. 세 번째 값은 [그림13]크립토 환경설정을 통해 생성한 인증서를 도커이미지에 복사하는 역할을 한다. 포트는 각 노드들이 통신을 할 때 이용할 포트번호를 설정한다.

표 10. 제안된 노드 도커컴포저 환경설정 구성정보

구분	설정값
services	peer0.orgApprover.example.com: peer1.orgApprover.example.com:
container_name	peer0.orgApprover.example.com: peer1.orgApprover.example.com:
peer0 environment	CORE_PEER_ID=peer0.orgApprover.example.com CORE_PEER_ADDRESS=peer0.orgApprover.example.com:7051 CORE_PEER_LISTENADDRESS=0.0.0.0:7051 CORE_PEER_CHAINCODEADDRESS=peer0.orgApprover.example.com:7052 CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:7052 CORE_PEER_GOSSIP_BOOTSTRAP=peer1.orgApprover.example.com:8051 CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.orgApprover.example.com:7051 CORE_PEER_LOCALMSPID=OrgApproverMSP
peer0 volumes	../channel-artifacts/genesis.block:/var/hyperledger/orderer/orderer.genesis.block ../crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/msp:/var/hyperledger/orderer/msp ../crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/tls/:/var/hyperledger/orderer/tls orderer.example.com:/var/hyperledger/production/orderer
orderer ports	7051:7051

환경설정 구성을 완료한 후 어류품질검사 블록체인에 이용할 어류품질검사 체인코드를 작성해야 한다. 본 연구에서는 [표 6] 어류품질검사 블록체인 시료정보에서 분석한 시료정보를 바탕으로 어류품질검사 체인코드(Inspection chaincode)를 작성 한다. 하이퍼레저 패브릭 체인코드는 현재 Go, Javascript 2가지 언어를 지원하며 본 연구에서는 Go 샘플 파일을 가지고 구현한다.

표 11. 어류품질검사 블록체인체인코드 함수

function	설명
Invoke	블록에 데이터를 쓰기 위해 사용한다.
queryInspection	체인 코드가 올바르게 인스턴스화되고 상태 DB가 채워 졌는지 확인한다.
initLedger	체인코드가 인스턴스화 중에 호출된다.
createInspection	원장을 업데이트하고 새로운 블록을 추가한다.
queryAllInspections	블록의 모든 데이터를 조회한다.
changeInspectionOwner	블록의 소유자를 변경한다.

1.2 트랜잭션 처리를 위한 어류품질검사 블록체인 API 웹서비스 설계

각 조직 기관들은 API 서버를 이용하여 어류품질검사 블록체인 네트워크 인스턴스 노드에 연결하여 거래가 이루어진다. 이 API 서버는 기관 참여자들의 개인키를 발급하여 어류품질검사 블록체인 네트워크에 접근할 수 있도록 한다. API 서버의 기능적 요구사항은 아래 [표 8]과 같다.

표 12. 어류품질검사 블록체인 조직구성별 기능

사용자	기능
유통자	- 안전성 검사정보 조회(검사 승인상태) - 안전성 검사정보 공유 수락, 거부 - 공유된 정보 이력 조회 - 안전성 검사정보 이력 조회
검사기관	- 안전성 검사정보기록 - 안전성 검사정보 제출기록(관리자에게 제출) - 안전성 검사제출 이력 조회
승인기관	- 안전성 검사결과 접수 - 안전성 검사정보기록 - 안전성 검사정보 제출기록(관리자에게 제출) - 안전성 검사제출 이력 조회 - 유통 승인 정보 조회 - 유통자 정보 조회

API 서버는 어류품질검사 블록체인 네트워크에 접속하는 인터페이스 역할을 한다. 블록체인 API를 통해 노드와 통신하며 다른 API 서버와 상호작용을 하거나 내부적으로 자료를 처리하는 기능을 한다. 유통자는 API 서버를 이용하여 어류품질검사 블록체인 정보를 조회하고 판매자에게 공유 승인할 수 있다. 검사기관은 API 서버를 통해 어류품질검사 블록체인 정보를 기록하고 해당 정보를 관리자에게 제출하여 검사내용을 확인하고 관리자는 승인기관으로 제출할 수 있도록 전자서명을 한다. 승인기관은 검사기관으로부터 제출된 어류품질검사를 접수하고 승인여부를 판단하여 전자서명을 통해 유통적합, 유통부적합 처리를 한다.

본 연구에서는 [표 13] 어류품질검사 블록체인 트랜잭션 처리 메소드와 같이 REST API의 4가지 메소드(Mothod)을 작성하여, 실제 데이터를 조회하고 추가하는 과정을 구현한다. 클라이언트 사용자는 API 서버로 GET 메소드를 사용하여 /api/allInspections 쿼리로 요청을 보낸다. 요청을 받은 API 서버는 어류품질검사 체인코드에서 구현된 해당 함수를 호출하여 어류품질검사 블록체인에 모든 정보를 읽어 와서 리턴 한다. /api/inspection?id=key 로 쿼리 요청시 해당하는

킷값에 블록 하나의 정보를 읽어 와서 보여준다. 어류품질검사 블록체인 정보를 추가하거나 거래를 요청시에는 POST 메소드 또는 PUT 메소드를 사용하며 어류품질 검사정보를 등록하고자 할때는 /api/addInspection 으로 쿼리를 요청하여 어류품질검사 블록을 추가시킨다. 사용자가 거래가 이루어지는 트랜잭션이 발생할 때는 PUT 메소드를 사용하여 /api/changeowner/inspection 쿼리를 전송하여 실행 시키며 이때 전송할 정보는 Body 에 Json 타입으로 정보를 실어서 전송한다.

표 13. 어류품질검사 블록체인 트랜잭션 처리 메소드

Method	Uri	설명
GET	/api/allInspections	모든 검사정보를 리턴한다.
GET	/api/inspection?Id=key	검사정보 아이디를 받아서 해당 검사정보를 리턴 한다.
POST	/api/addInspection/	검사정보를 추가한다.
PUT	/api/changeowner/inspection	검사정보의 소유자를 변경한다.

2. 블록체인 기반 어류품질검사 데이터의 무결성 서비스 구현환경

하이퍼레저 패브릭은 구축환경이 다양한 형태로 유연하게 만들 수 있는 장점이 있다. 단일서버에 가상화 환경으로 구축하거나 물리적 서버를 여러 개 구축하여 각 기능별로 따로 구축 할 수도 있다. 본 연구에서는 가상화 환경에 도커를 설치하여 구축한다. 운영체제는 우분투 18.4를 이용하고 하이퍼레저 패브릭에서 기본으로 제공하는 패브릭 SDK를 설치하여 언어는 Go언어를 사용한다. API 서버는 Node 서버를 이용하여 자바스립트로 애플리케이션을 개발한다. 개발 IDE는 마이크로소프트에서 개발하여 무료 배포하는 비주얼스튜디오코드(Visual Studio Code)로 개발한다. 해당 시스템을 구현하기 위한 개발 도구 및 사양은 [표 14] 어류품질검사 블록체인 구현환경과 같다.

표 14. 어류품질검사 블록체인 구현환경

항목	도구
OS	Ubuntu 18.4
언어	Go
	Java Scripts
플랫폼	Node.js
	Hyperledger Fabric 1.4
	Docker 18.09.5
	Fabric SDK 1.4
	VirtualBox 6.0
IDE	Visual Studio Code

고언어는 2009년 구글이 개발한 오픈소스 프로그래밍 언어이다. 고언어는 쉽고 간결한 문법이 특징이며 정적 타입 컴파일 언어의 효율적인 개발을 가능하게 한다. 멀티 코어 및 네트워크 머신을 최대한 활용하는 프로그램을 쉽게 작성할 수 있으며, 유연한 모듈형태 프로그램을 구성 할 수 있다[13]. 구글의 일부시스템은 고언어로 구축되어 있고 드롭박스나 클라우드플레어, 사운드클라우드, 넷플릭스 등에서도 기존의 기능들을 새롭게 고언어를 이용해 바꿨다[14]. 어류품질검사 블록체인을 형성하는 체인코드 역시 고언어로 작성한다.

노드제이에스(Node.js)는 네트워크 애플리케이션 개발에 사용되는 소프트웨어 플랫폼으로 서버사이드 개발에 주로 사용된다. 자바스크립트 언어로 작성하며 내장 HTTP 서버 라이브러리를 포함하고 있어 별도의 애플리케이션 서버 없이 웹 서비스가 가능하다[14]. 자바스크립트언어가 웹브라우저에서 실행되는 것과는 달리 노드제이에스는 V8 (자바스크립트 엔진)이 빌드 된 런타임 환경을 제공하여 자바스크립트언어가 서버 측에서 실행된다[15]. 블록체인 기반 어류품질검사 데이터의 무결성 서비스를 사용자에게 제공하기 위해서는 노드제이에스의 런타임 환경이 구축 되 야 한다. 이 런타임 환경에서 자바스크립트언어를 사용하여 블록체인서비스를 제어한다.

도커(Docker)는 리눅스의 응용 프로그램들을 격리된 소프트웨어 도커 컨테이너 안에 배치시키는 오픈 소스 프로젝트이다[16]. 하이퍼브이(Hyper-V), 버추얼박스, 브이엠웹어(VMware)와 같은 가상화 소프트웨어는 운영체제를 직접 가상화 함으로서 호스트 운영체제의 자원을 낭비하는 단점이 있다. 하지만 도커 컨테이너 기술은 운영체제를 가상화하지 않아 타 가상화 소프트웨어보다 훨씬 가벼우며 브이엠(VM)을 포함하여 한대의 서버에 여러 개의 서비스를 구동하기 좋다. 도커 컨테이너는 소프트웨어의 실행에 필요한 코드, 런타임, 시스템 도구, 시스템 라이브러리 등 서버에 설치되는 모든 것을 포함하는 완전한 파일 시스템 안에 감싼다[17]고 도커 웹 페이지에서는 설명하고 있다. 어류품질검사 블록체인은 도커컨테이너에 적재되어 각 노드들과 트랙잭션이 이루어지도록 한다.

하이퍼레저는 리눅스 재단(Linux Foundation)이 주도하고 진행하고 있는 오픈 소스 프로젝트이며, Hyperledger Fabric, Hyperledger Iroha, Hyperledger Sawtooth, Hyperledger Burrow등 많은 프레임워크가 설계, 개발 및 사용되어 지고 있다. 이 중 하이퍼레저 패브릭이 설계, 개발에 더 편리하게 할 수 있도록 하이퍼레저 콤포저(Hyperledger Composer), 하이퍼레저 탐색기(Hyperledger Explorer) 등 여러 도구들을 제공해줌으로써 가장 비지니스모델로 많이 적용되고 있다. 하이퍼레저 패브릭이 다른 블록체인과 다른 기능적 특징을 보면, 비트코인이나 이더리움의 블록체인은 모든 역할을 한 가지 모듈이 처리하는 반면 하이퍼레저 패브릭은 각 역할을 모듈 별로 구분되어 수행토록 구성되어 있다. 이렇게 하이퍼레저 패브릭은 모듈화 구성이 가능한 아키텍처를 갖추고 있어 뱅킹, 금융, 보험, 건강관리, 인적 자원, 공급망 및 디지털 음악 전달 등 다양한 산업 최적화를 가능하게 한다

3. 블록체인 기반 어류품질검사 데이터의 무결성 서비스 구현

버츄얼박스에 어류품질검사 블록체인 네트워크 서비스용으로 우분투 운영체제를 설치한다. 우분투 설치 후 하이퍼레저 패브릭을 설치하기 전에 패브릭에 각

노드들이 서비스될 도커를 우선 설치한다. sudo apt-get -y install docker-compose을 터미널에 작성한 후 실행 하면 도커가 설치된다. 하이퍼레저 패브릭에서 제공하는 블록체인 네트워크의 체인코드를 작성하기 위해 Go언어를 설치한다. wget https://golang.org/dl/go1.11.11.linux-amd64.tar.gz 명령어 터미널에서 실행하여 압축을 푼 후 설치한다.

3.1 어류품질검사 블록체인 네트워크 구현

어류품질검사 블록체인 네트워크 분석 과정에서 [표 7] 제안된 크립토 환경설정 구성요소, [표 8] 제안된 트랙잭션 환경설정 구성요소, [표 9] 제안된 도커컴포져 환경설정 구성요소 정보를 작성 한다. [그림 13]과 같이 먼저 크립토 환경설정을 작성 한다.

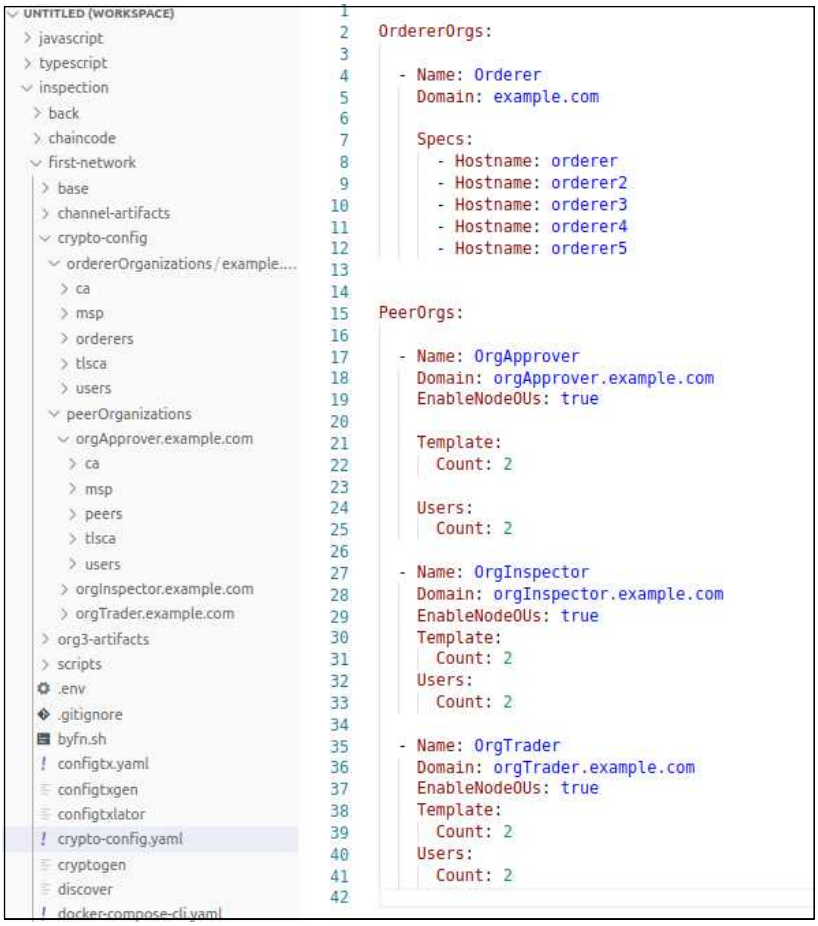


그림 13. 제안된 크립토 환경설정(crypto-config.yaml)

크립토 환경설정은 [그림 13]제안된 크립토 환경설정의 crypto-config.yaml에 작성하며 어류품질검사 블록체인 네트워크에서 사용될 오더러 서비스와 각 노드 서비스의 속성을 정의 하는 파일이다.

어류품질검사 블록체인 네트워크가 인스턴스화 될 때 생성될 오더러 서비스의 이름, 도메인, 생성 수 등을 작성하고 노드역할을 할 조직의 이름, 도메인, 생성 수 및 각 조직의 사용자 수 등을 작성한다. [그림 13]은 crypto-config.yaml에 작성한 파일 정보이다. 작성 후 하이퍼레저 패브릭에서 제공하는 바이너리 파일 cryptogen을 실행한다. cryptogen은 crypto-config 폴더가 자동 생성되고 ordererOrganizations폴더와 peerOrganizations폴더 2개의 폴더를 자동 생성된다.

```
frdave@hyperledger-fablic:~/inspection/first-network$ tree crypto-config -d -L 3
crypto-config
├── ordererOrganizations
│   └── example.com
│       ├── ca
│       ├── msp
│       ├── orderers
│       ├── tlsca
│       └── users
└── peerOrganizations
    ├── orgApprover.example.com
    │   ├── ca
    │   ├── msp
    │   ├── peers
    │   ├── tlsca
    │   └── users
    ├── orgInspector.example.com
    │   ├── ca
    │   ├── msp
    │   ├── peers
    │   ├── tlsca
    │   └── users
    └── orgTrader.example.com
        ├── ca
        ├── msp
        ├── peers
        ├── tlsca
        └── users

26 directories
```

그림 14. 생성된 크립토 환경설정 폴더 와 파일

ordererOrganizations폴더에는 개별 인증서가 생성되고 crypto-config.yaml에서 설정한 5개의 오더러 서비스의 개별 인증서가 자동 생성된다. peerOrganizations 폴더에는 orgApprover, orgInspector, orgTrader의 개별 도메인과 개별인증서를

생성하고 각 도메인별 2개의 피어가 생성된다. 피어는 peer0, peer1로 시작하는 호스트네임으로 orgApporver는 peer0.orgApporver.example.com 과 peer1.orgApporver.example.com 2개의 피어를 생성한다. 크립토 환경설정 실행 명령어는 ./cryptogen generate --config=./crypto-config.yaml 이다.

크립토 환경설정을 끝낸 후 트랙잭션 환경설정을 한다. 트랙잰션 환경설정 파일명은 configtx.yaml 파일이다. [그림13] 어류품질검사 블록체인 트랙잭션 환경설정은 어류품질검사 블록체인 네트워크에 구성 피어들이 채널(channel)에 참여할 수 있도록 인스턴스화 설정을 하고 구성조직을 생성한다. 트랙잭션 환경설정은 하이퍼레저 패브릭에서 제공하는 바이너리 파일 configtxgen을 이용하며 제네시스블록 파일, 채널트랜잭션파일, 피어트랜잭션파일 생성한다.

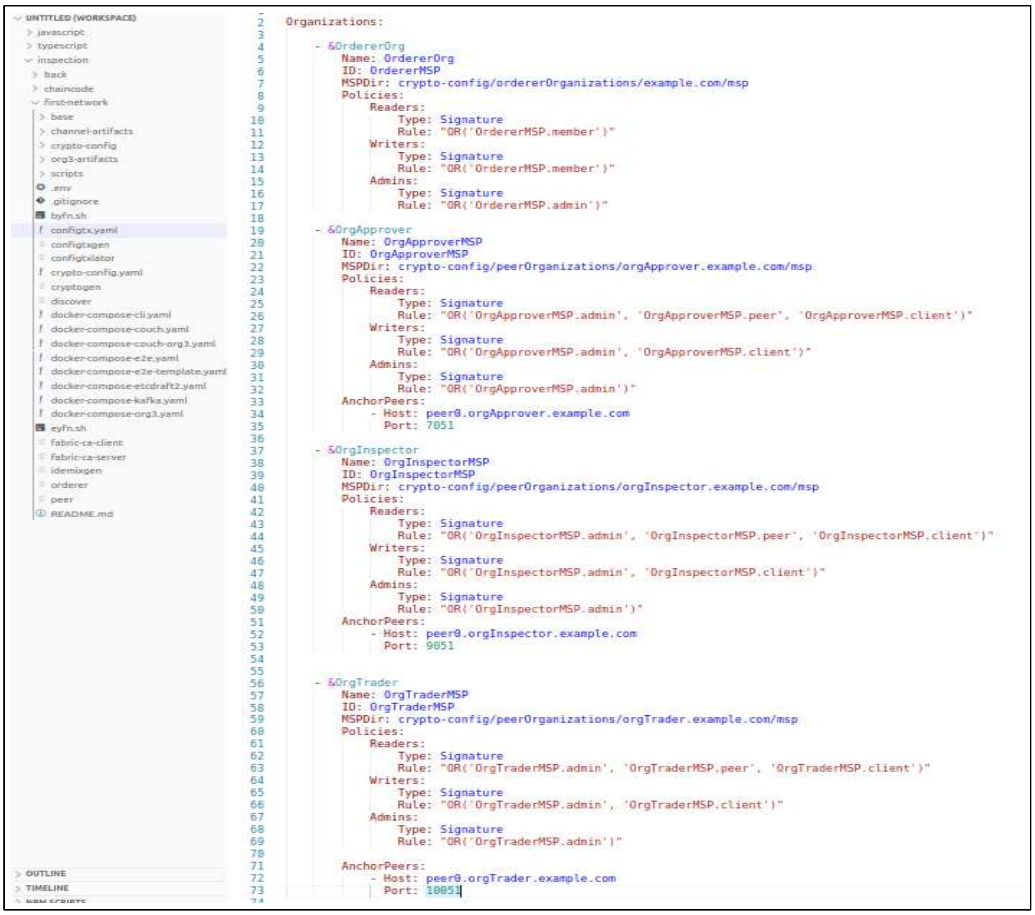


그림 15. 제안된 트랜잭션 환경설정(configtx.yaml)

먼저 제네시스블록 파일을 생성한다. 실행명령어는 ./configtxgen -profile TwoOrgsOrdererGenesis -outputBlock ./channel-artifacts/genesis.block이다. 명령어 실행 후 channel-artifacts 위치로 이동 후 생성된 genesis.block 파일을 확인 한다. 다음 채널트랜잭션 파일을 생성한다. 채널명은 기본값 mychannel 로 생성한다. 채널생성 명령어는 ./configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./channel-artifacts/ -channelId mychannel이다. 명령어 실행 후 channel-artifacts 위치로 이동 후 생성된 channel.tx 파일을 확인한다. 다음 피어트랜잭션 파일을 생성한다. [그림 20]의 configtx.yaml에 설정한 OrgApproverMSP, OrgInpectorMSP, OrgTraderMSP 3개의 피어트랜잭션을 생성한다. 피어트랙잭션 명령어는 다음과 같다.

./configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/OrgApproverMSPanchors.tx -channelID mychannel -asOrg OrgApproverMSP

./configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/OrgInspectorMSPanchors.tx -channelID mychannel -asOrg OrgInspectorMSP

./configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/OrgTraderMSPanchors.tx -channelID mychannel -asOrg OrgTraderMSP

명령어 실행 후 channel-artifacts 위치로 이동 후 [그림 16]트랜잭션 파일 목록과 같이 생성된 OrgApproverMSPanchors.tx, OrgInspectorMSPanchors.tx, OrgTraderMSPanchors.tx 파일을 확인한다.

```
frdave@hyperledger-fablic:~/inspection/first-network$ tree channel-artifacts
channel-artifacts
├── channel.tx
├── genesis.block
├── OrgApproverMSPanchors.tx
├── OrgInspectorMSPanchors.tx
└── OrgTraderMSPanchors.tx

0 directories, 5 files
```

그림 16. 트랜잭션 파일 목록

트랜잭션환경설정을 끝낸 후 도커컴포져 환경설정을 한다. 도커컴포져 환경설정 파일명은 여러 개의 파일이 존재한다. 본 구현은 docker-compose-base.yaml, docker-compose-cli.yaml 2개의 파일만 설정을 하고 코치디비 외 기타 도커컴포저 파일은 기본 구성설정을 그대로 사용한다. 도커콤포저는 docker-compose-cli.yaml 파일을 실행해서 설정정보에 맞춰서 도커 이미지를 생성하는 역할을 한다.

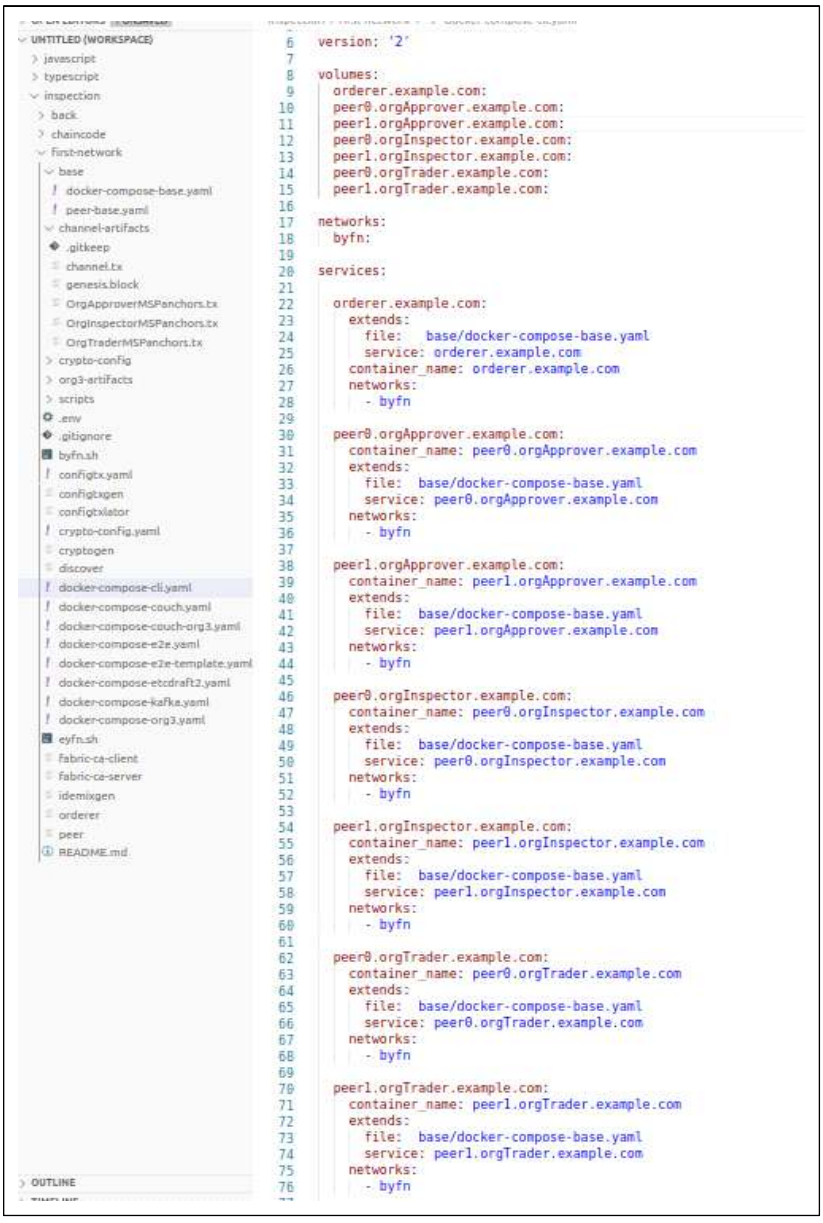


그림 17. 도커이미지 생성 환경설정
cli(docker-compose-cli.yaml)

서비스(services)는 docker-compose-base.yaml에 정보를 가져오는 file: base/docker-compose-base.yaml 로 선언한다. docker-compose-base.yaml 파일은 docker-compose-cli.yaml 파일의 변수로 지정된 정보로 어류품질검사 블록체인 네트워크 관련정보는 [그림 23]에 docker-compose-base.yaml 에 설정한다.

```
version: '2'

services:

  orderer.example.com:
    container_name: orderer.example.com
    extends:
      file: peer-base.yaml
      service: orderer-base
    volumes:
        - ../channel-artifacts/genesis.block:/var/hyperledger/orderer/orderer.genesis.block
        - ../crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/msp:/var/hyperledger/orderer/msp
        - ../crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/tls/:/var/hyperledger/orderer/tls
        - orderer.example.com:/var/hyperledger/production/orderer
    ports:
      - 7050:7050

  peer0.orgApprover.example.com:
    container_name: peer0.orgApprover.example.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer0.orgApprover.example.com
      - CORE_PEER_ADDRESS=peer0.orgApprover.example.com:7051
      - CORE_PEER_LISTENADDRESS=0.0.0.0:7051
      - CORE_PEER_CHAINCODEADDRESS=peer0.orgApprover.example.com:7052
      - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:7052
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer1.orgApprover.example.com:8051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.orgApprover.example.com:7051
      - CORE_PEER_LOCALMSPID=OrgApproverMSP
    volumes:
        - /var/run/:/host/var/run/
        - ../crypto-config/peerOrganizations/orgApprover.example.com/peers/peer0.orgApprover.example.com/msp:/etc/hyperledger/fabric/msp
        - ../crypto-config/peerOrganizations/orgApprover.example.com/peers/peer0.orgApprover.example.com/tls:/etc/hyperledger/fabric/tls
        - peer0.orgApprover.example.com:/var/hyperledger/production
    ports:
      - 7051:7051

  peer1.orgApprover.example.com:
    container_name: peer1.orgApprover.example.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=peer1.orgApprover.example.com
      - CORE_PEER_ADDRESS=peer1.orgApprover.example.com:8051
      - CORE_PEER_LISTENADDRESS=0.0.0.0:8051
      - CORE_PEER_CHAINCODEADDRESS=peer1.orgApprover.example.com:8052
      - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:8052
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.orgApprover.example.com:8051
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.orgApprover.example.com:7051
      - CORE_PEER_LOCALMSPID=OrgApproverMSP
    volumes:
        - /var/run/:/host/var/run/
        - ../crypto-config/peerOrganizations/orgApprover.example.com/peers/peer1.orgApprover.example.com/msp:/etc/hyperledger/fabric/msp
        - ../crypto-config/peerOrganizations/orgApprover.example.com/peers/peer1.orgApprover.example.com/tls:/etc/hyperledger/fabric/tls
        - peer1.orgApprover.example.com:/var/hyperledger/production

    ports:
      - 8051:8051

  peer0.orgInspector.example.com:
    container_name: peer0.orgInspector.example.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
```

그림 18. 도커 노드 이미지 생성 설정 파일(docker-compose-base.yaml)

설정을 완료 한후 도커컴포저 명령어를 실행한다. ./docker-compose -f docker-compose-cli.yaml up -d 도커컴포저 명령어를 실행한 후 docker ps 명령어를 이용하여 생성된 도커이미지를 [그림 19]와 같이 볼 수 있다.

```
root@hyperledger-fabric:~/inspection/first-network$ docker ps
CONTAINER ID        IMAGE                               COMMAND                  CREATED             STATUS              PORTS                     NAMES
67e3d3bf018e        hyperledger/fabric-tools:latest     "/bin/bash"              About a minute ago  Up 58 seconds                                 cli
4688a9d1e840        hyperledger/fabric-orderer:latest   "orderer"                About a minute ago  Up 59 seconds       0.0.0.0:10050->7050/tcp   orderer4.example.com
868b4827440a        hyperledger/fabric-orderer:latest   "orderer"                About a minute ago  Up 59 seconds       0.0.0.0:9050->7050/tcp    orderer3.example.com
8ca8296c195b        hyperledger/fabric-orderer:latest   "orderer"                About a minute ago  Up 59 seconds       0.0.0.0:11050->7050/tcp   orderer5.example.com
fc4f34d3788f        hyperledger/fabric-orderer:latest   "orderer"                About a minute ago  Up 58 seconds       0.0.0.0:8050->7050/tcp    orderer2.example.com
6a202c79a6ba        hyperledger/fabric-peer:latest      "peer node start"        5 hours ago         Up 5 hours          0.0.0.0:12051->12051/tcp  peer1.orgTrader.example.com
9116daaf803a        hyperledger/fabric-peer:latest      "peer node start"        5 hours ago         Up 5 hours          0.0.0.0:9051->9051/tcp    peer0.orgInspector.example.com
704d6dea6eb1        hyperledger/fabric-peer:latest      "peer node start"        5 hours ago         Up 5 hours          0.0.0.0:7051->7051/tcp    peer0.orgApprover.example.com
01e1cdf6247f        hyperledger/fabric-orderer:latest   "orderer"                5 hours ago         Up 5 hours          0.0.0.0:7050->7050/tcp    orderer.example.com
077133a2cc24        hyperledger/fabric-peer:latest      "peer node start"        5 hours ago         Up 5 hours          0.0.0.0:8051->8051/tcp    peer1.orgApprover.example.com
413fbafd2ce9        hyperledger/fabric-peer:latest      "peer node start"        5 hours ago         Up 5 hours          0.0.0.0:10051->10051/tcp  peer1.orgInspector.example.com
1fc854d96663        hyperledger/fabric-peer:latest      "peer node start"        5 hours ago         Up 5 hours          0.0.0.0:11051->11051/tcp  peer0.orgTrader.example.com
```

그림 19. 생성된 도커 이미지 목록

도커컴포저 환경설정을 끝낸후 어류품질검사 블록체인 네트워크에 인스턴트화 할 체인코드를 작성한다. 체인코드를 개발하기 위해 어류품질검사 블록체인 구조체를 먼저 선언한다. 구조체 변수정보는 [표 6]에서 정의한 시료정보를 기준으로 작성한다.

```go
package main

/* Imports
 * 4 utility libraries for formatting, handling bytes, reading and
 * 2 specific Hyperledger Fabric specific libraries for Smart Contr
 */
import (
    "bytes"
    "encoding/json"
    "fmt"
    "strconv"

    "github.com/hyperledger/fabric/core/chaincode/shim"
    sc "github.com/hyperledger/fabric/protos/peer"
)

// Define the Smart Contract structure
type SmartContract struct {
}

// Define the inspection structure, with 4 properties.  Structure t
type Inspection struct {
    SamplingCnt       string `json:"sampling_cnt"`
    SamplingWeight    string `json:"inspection_weight"`
    InspectionItem    string `json:"inspection_item"`
    FishbowlNumber    string `json:"fishbowl_number"`
    InspectionResult string `json:"inspection_result"`
    ValidityDate      string `json:"validity_date"`
    ApplyDate         string `json:"apply_date"`
    Owner             string `json:"owner"`
 }
```

그림 20. 제안된 chaincode 구조체 코드

하이퍼레저 패브릭에서 기본 제공되는 샘플 Go 파일을 이용하여 파일 작성을

한다. [그림 20]과 같이 Inspectioin.go 파일을 만들어 기본 구조체를 작성한다. 이 파일은 어류품질검사 블록체인 네트워크가 인스턴스화 된 후 블록체인에 저장될 각 정보를 정의 하는 것이며 시료정보의 값이 저장되는 변수를 설정한다.

구조체 작성 후 Invoke 함수코드를 작성한다. Invoke 함수 코드는 어류품질검사 블록체인에 데이터를 조회하거나 쓰기 위해서 요청되는 쿼리를 파악하여 기능에 맞게 함수를 호출하도록 한다. [그림 21]에서 보면 queryInspection이 요청되면 Inovoke 함수는 [그림 22]에 작성된 queryInspection함수의 결과값을 리턴해 준다.

```
/*
 * The Invoke method is called as a result of an application request to run the Smart Contr
 * The calling application program has also specified the particular smart contract functio
 */
func (s *SmartContract) Invoke(APIstub shim.ChaincodeStubInterface) sc.Response {

    // Retrieve the requested Smart Contract function and arguments
    function, args := APIstub.GetFunctionAndParameters()
    // Route to the appropriate handler function to interact with the ledger appropriately
    if function == "queryInspection" {
        return s.queryInspection(APIstub, args)
    } else if function == "initLedger" {
        return s.initLedger(APIstub)
    } else if function == "createInspection" {
        return s.createInspection(APIstub, args)
    } else if function == "queryAllInspections" {
        return s.queryAllInspections(APIstub)
    } else if function == "changeInspectionOwner" {
        return s.changeInspectionOwner(APIstub, args)
    }

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

그림 21. 제안된 Invoke 함수 코드

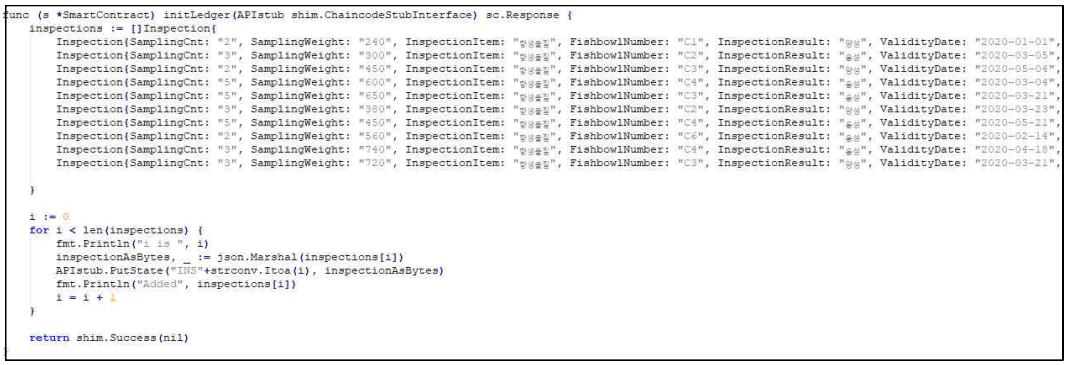
다음은 어류품질검사 블록체인 네트워크에 데이터를 조회하는 함수로 queryInspection 함수를 작성한다. queryInspection 요청시 args 옵션에 원하는 값을 입력하면 그 값에 해당하는 정보를 조회하여 결과값을 리턴해 준다.

```
func (s *SmartContract) queryInspection(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {

    if len(args) != 1 {
        return shim.Error("Incorrect number of arguments. Expecting 1")
    }

    inspectionAsBytes, _ := APIstub.GetState(args[0])
    return shim.Success(inspectionAsBytes)
}
```

그림 22. 제안된 queryInspection 함수 코드

다음은 어류품질검사 블록체인 데이터를 초기화 하는 initLedger 함수를 작성한다. [그림 23]에서는 초기화시 안전성검사 정보 10개를 미리 설정하고 초기화

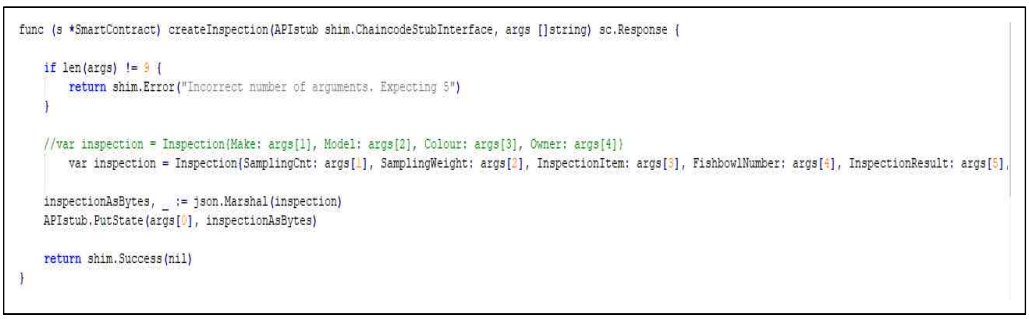
를 한다. 어류품질검사 블록체인 네트워크가 인스턴스화 된 후 블록체인이 제대로 활성화 됐는지 파악하기 위해 10개의 어류품질검사 블록체인 더미블록을 미리 생성 시킨다.



```
func (s *SmartContract) initLedger(APIstub shim.ChaincodeStubInterface) sc.Response {
    inspections := []Inspection{
        Inspection{SamplingCnt: "2", SamplingWeight: "240", InspectionItem: "항생물질", FishbowlNumber: "C1", InspectionResult: "양성", ValidityDate: "2020-01-01",
        Inspection{SamplingCnt: "3", SamplingWeight: "300", InspectionItem: "항생물질", FishbowlNumber: "C2", InspectionResult: "음성", ValidityDate: "2020-03-05",
        Inspection{SamplingCnt: "2", SamplingWeight: "450", InspectionItem: "항생물질", FishbowlNumber: "C3", InspectionResult: "양성", ValidityDate: "2020-05-04",
        Inspection{SamplingCnt: "5", SamplingWeight: "600", InspectionItem: "항생물질", FishbowlNumber: "C4", InspectionResult: "음성", ValidityDate: "2020-03-04",
        Inspection{SamplingCnt: "5", SamplingWeight: "650", InspectionItem: "항생물질", FishbowlNumber: "C3", InspectionResult: "음성", ValidityDate: "2020-03-21",
        Inspection{SamplingCnt: "3", SamplingWeight: "380", InspectionItem: "항생물질", FishbowlNumber: "C2", InspectionResult: "양성", ValidityDate: "2020-03-23",
        Inspection{SamplingCnt: "5", SamplingWeight: "450", InspectionItem: "항생물질", FishbowlNumber: "C4", InspectionResult: "음성", ValidityDate: "2020-05-21",
        Inspection{SamplingCnt: "2", SamplingWeight: "560", InspectionItem: "항생물질", FishbowlNumber: "C6", InspectionResult: "음성", ValidityDate: "2020-02-14",
        Inspection{SamplingCnt: "3", SamplingWeight: "740", InspectionItem: "항생물질", FishbowlNumber: "C4", InspectionResult: "음성", ValidityDate: "2020-04-18",
        Inspection{SamplingCnt: "3", SamplingWeight: "720", InspectionItem: "항생물질", FishbowlNumber: "C3", InspectionResult: "양성", ValidityDate: "2020-03-21",

    }

    i := 0
    for i < len(inspections) {
        fmt.Println("i is ", i)
        inspectionAsBytes, _ := json.Marshal(inspections[i])
        APIstub.PutState("INS"+strconv.Itoa(i), inspectionAsBytes)
        fmt.Println("Added", inspections[i])
        i = i + 1
    }

    return shim.Success(nil)
}
```

그림 23. 제안된 initLedger 함수 코드

다음은 어류품질검사 블록체인에 데이터를 생성하는 createInspection 함수를 작성한다. 어류품질안전성검사의 시료정보를 args에 입력하고 실행하면 새로운 안전성검사 블록이 생성된다.



```
func (s *SmartContract) createInspection(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {

    if len(args) != 9 {
        return shim.Error("Incorrect number of arguments. Expecting 5")
    }

    //var inspection = Inspection{Make: args[1], Model: args[2], Colour: args[3], Owner: args[4]}
        var inspection = Inspection{SamplingCnt: args[1], SamplingWeight: args[2], InspectionItem: args[3], FishbowlNumber: args[4], InspectionResult: args[5],

    inspectionAsBytes, _ := json.Marshal(inspection)
    APIstub.PutState(args[0], inspectionAsBytes)

    return shim.Success(nil)
}
```

그림 24. 제안된 createInspection 함수 코드

다음은 어류품질검사 블록체인에 모든 데이터를 조회하는 queryAllInspections 함수를 작성한다. [그림 25]에 startKey INS0부터 endKey INS999까지 설정하고 queryAllInspectins이 실행되면 총 1,000개의 안전성 검사정보를 조회해서 리턴한다. 웹서비스에서 args 값을 입력하여 startKey와 endKey 값을 받아서 조회되는 정보의 수를 변경할 수 있다.

```
func (s *SmartContract) queryAllInspections(APIstub shim.ChaincodeStubInterface) sc.Response {

    startKey := "IN50"
    endKey := "IN5999"

    resultsIterator, err := APIstub.GetStateByRange(startKey, endKey)
    if err != nil {
        return shim.Error(err.Error())
    }
    defer resultsIterator.Close()

    // buffer is a JSON array containing QueryResults
    var buffer bytes.Buffer
    buffer.WriteString("[")

    bArrayMemberAlreadyWritten := false
    for resultsIterator.HasNext() {
        queryResponse, err := resultsIterator.Next()
        if err != nil {
            return shim.Error(err.Error())
        }
        // Add a comma before array members, suppress it for the first array member
        if bArrayMemberAlreadyWritten == true {
            buffer.WriteString(",")
        }
        buffer.WriteString("{\"Key\":")
        buffer.WriteString("\"")
        buffer.WriteString(queryResponse.Key)
        buffer.WriteString("\"")

        buffer.WriteString(", \"Record\":")
        // Record is a JSON object, so we write as-is
        buffer.WriteString(string(queryResponse.Value))
        buffer.WriteString("}")
        bArrayMemberAlreadyWritten = true
    }
    buffer.WriteString("]")

    fmt.Printf("- queryAllInspections:\n%s\n", buffer.String())

    return shim.Success(buffer.Bytes())
}
```

그림 25. 제안된 queryAllInspections 함수 코드

다음은 어류품질검사 블록체인에 데이터 정보를 변경하는 changeInspectionOwner 함수 작성를 작성한다. [그림 26]은 소유자의 정보를 변경하는 함수로 안전성 검사정보의 거래가 이루어졌을 때 변경기록이 되도록 한다. 소유자정보 외에 다른 값들이 변경 됐을 때도 블록체인에 변경정보가 기록되도록 할 수 있다.

```
func (s *SmartContract) changeInspectionOwner(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {

    if len(args) != 2 {
        return shim.Error("Incorrect number of arguments. Expecting 2")
    }

    inspectionAsBytes, _ := APIstub.GetState(args[0])
    inspection := Inspection{}

    json.Unmarshal(inspectionAsBytes, &inspection)
    inspection.Owner = args[1]

    inspectionAsBytes, _ = json.Marshal(inspection)
    APIstub.PutState(args[0], inspectionAsBytes)

    return shim.Success(nil)
}
```

그림 26. 제안된 changeInspectionOwner 함수 코드

여기 까지 작성 하면 기본적인 기능은 구현이 됐으며 본 연구는 프로토타입형태로 구현 할 예정이므로 상세한 제약이나 오류 사항에 대한 기능은 작성하지 않는다. 이제 어류품질검사 블록체인 네트워트 서비스를 시작하면 된다. 네트워크 서비스 시작은 ./startInspection.sh 으로 시작 하며 시작과 동시에 작성된 코드를 통해서 어류품질검사 블록체인 네트워크가 인스턴스화 된다.

3.2 어류품질검사 블록체인 네트워크 실행결과

인스턴스화 된 어류품질검사 블록체인 네트워크에 정보를 제어하기 위해서 하이퍼레저 패브릭 샘플로 제공되는 Javascript를 이용한다. 정보를 제어하기 전에 먼저 어류품질검사 블록체인 네트워크에 접근하기 위한 관리자와 사용자를 등록해야 한다. 관리자와 사용자 등록은 Fabric-CA 서버가 처리를 하며 Fabric-CA는 등록된 사용자의 인증서를 보관하고 있고 접근하는 사용자는 개인키 인증서를 가지고 Fabric-CA가 보관한 인증서와 검증을 거친 후 네트워크에 접근해야 한다.

관리자등록은 enrollAdmin.js 파일을 실행하면 Fabric-CA 서버가 관리자 정보를 받아 등록 처리하고 관리자에게 인증서를 발급한다.

그림 27. 관리자 등록 명령어

사용자등록은 registerUser.js 파일을 실행하면 Fabric-CA 서버가 사용자 정보를 받아 등록 처리하고 사용자에게 인증서를 발급한다.

그림 28. 사용자 등록 명령어

등록을 완료 한 후 프레임워크에서 지원하는 query.js 파일을 실행 한다. 실행명령어는 터미널에서 query.js 파일을 실행하면 [그림 29]에 Go언어로 작성된 queryAllInspections 함수 코드를 실행하게 되고 queryAllInspections가 리턴하는 결과값을 출력한다. 결과 값은 [그림 24] 제안된 createInspection 함수 코드로 초기화 된 기본 데이터들이 json 형태로 파싱 된다. qurey.js 는 블록체인에 구축되어 있는 모든 데이터를 불러오는 함수로 구성되어 있고 [그림 29] chaincode 조회결과 처럼 모든 자료가 노출된 걸 확인 할 수 있다.

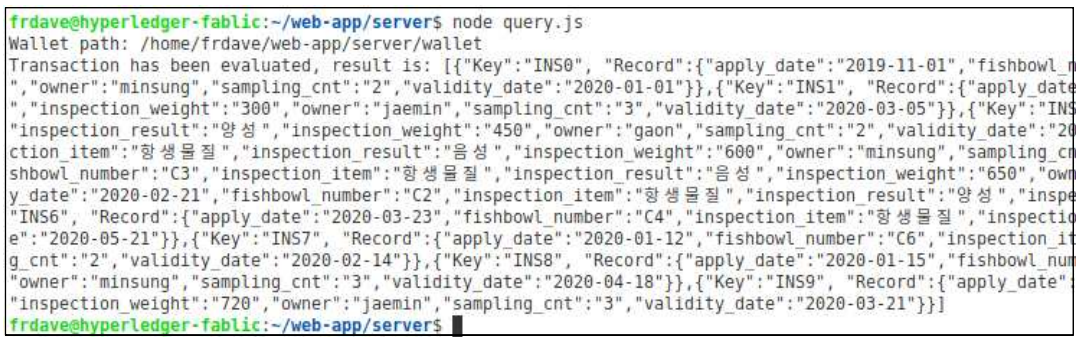


그림 29. chaincode 조회결과

데이터생성은 터미널에서 node invoke.js를 파일을 실행하여 할 수 있다. [그림 24] 제안된 createInspection 함수 코드가 실행되고 정보를 생성한다. invoke.js파

일에 추가할 데이터를 invoke.js 파일 미리 작성한다.

```
await contract.submitTransaction('createInspection', 'INS15', '3', '550', '항생물질', 'C3','양성','2020-02-21','2020-03-19','jaemin');
console.log('Transaction has been submitted Data INS15');
```

그림 30. chaincode 생성 파라미터

submitTransaction 함수에 파라미터로 'createinspection', 'INS10', '3', '550' ,'항생물질', 'C3', '음성', '2020-02-21', '2020-03-19', 'jaemin' 적용한 후 invoke.js 실행한다. 파라미터는 [그림 24]에 createInspection 함수에 args로 입력되고 처리된다.

```
frdave@hyperledger-fablic:~/web-app/server$ node invoke.js
Wallet path: /home/frdave/web-app/server/wallet
Transaction has been submitted Data INS15
frdave@hyperledger-fablic:~/web-app/server$
```

그림 31. chaincode 생성 명령어

node invoke.js 실행후 등록 메세지를 확인 할 수 있다. 등록된 정보를 확인하기 위해 query.js 실행 해서 블록체인에 해당 자료가 실제 등록되었는지 확인한다. [그림 32]에 등록된 블록 내용이 확인이 된다. 어류품질검사 체인코드의 기본적인 프로토타입이 완성 된다.

그림 32. chaincode 조회 명령어

3.3 어류품질검사 블록체인 API 서비스 구현

Frontend 에서 상황에 따라 발생된 요청을 이 API 서버(Backend)가 모든 처리를 한다. API 서버는 Node.js 서버를 이용한다. Node.js 서버를 구축하기 위해서 Node.js 와 Node패키지관리도구 NPM을 설치한다.

```
curl -sL https://deb.nodesource.com/setup_8.x | sudo bash -
sudo apt install nodejs
```

본 구현은 모든 검사정보를 조회하는 기능, 키 정보를 이용해 단일 검사정보를 조회하는 기능, 검사정보를 추가하는 기능, 검사정보의 소유자를 변경하는 기능을 구현 하였다. [그림 33] api 서버 구현 파일에 보면 소스코드는 app.js 와 network.js 2개 파일로 구성되며 app.js는 요청 쿼리의 controller 역할을 하고 network.js 는 해당요청의 처리를 위한 service 역할을 한다. API 서버 역할만을 필요로 하므로 view 구성해야 하는 html 페이지는 구현하지 않는다.

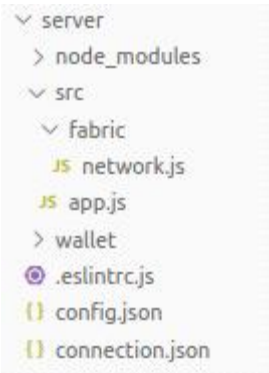


그림 33. api 서버
구현 파일

[그림 34-37]은 app.js 파일의 노드서버의 자바스크립트 소스코드이며 위 [표 13]에 정의된 4가지 요청사항의 대한 소스코드이다. 소스는 하이퍼레져 패브릭 샘플파일 query.js 와 invoke.js를 참고하여 작성한다.

3.3.1 app.js 소스 코드

[그림 34]는 어류품질검사 체인코드의 queryAllInspections을 호출하여 블록체인 정보의 모든 검사정보를 조회하는 기능이다.

```
app.get('/queryAllInspections', (req, res) => {
    network.queryAllInspections()
        .then((response) => {
            let InspectionsRecord = JSON.parse(response);
            res.send(InspectionsRecord);
        });
});
```

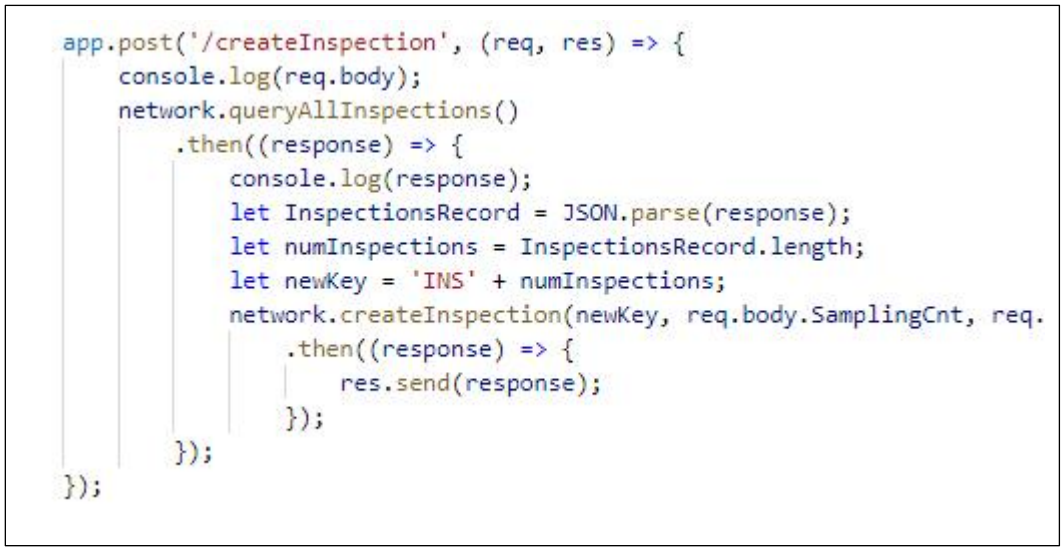
그림 34. 제안된 queryAllInspections 구현 코드

[그림 35]는 query로 키값을 입력받아서 해당 키값에 따라 검사정보를 받아서 단일 검사정보를 조회하는 기능을 한다. 어류품질검사 체인코드의 querySingleInspecition 함수를 호출하여 정보를 리턴 받는다.

```
app.get('/querySingleInspection', (req, res) => {
    network.querySingleInspection(req.query.key)
        .then((response) => {
            let InspectionsRecord = JSON.parse(response);
            res.send(InspectionsRecord);
        });
});
```

그림 35. 제안된 querySingleInspection 구현 코드

[그림 36]은 어류품질검사 블록체인 네트워크에 블록을 생성하는 기능이다. 안전성검사 체인코드에 createInspection 함수를 호출하며 블록생성시 안전성검사 고유키인 'INS' 에 어류품질검사 블럭의 총 개수를 연결해서 고유 키 값을 만든다. 클라이언트는 POST 메소드를 이용하여 안전성검사 정보를 바디에 넣어서 Json 형태로 전송한다.



```
app.post('/createInspection', (req, res) => {
    console.log(req.body);
    network.queryAllInspections()
        .then((response) => {
            console.log(response);
            let InspectionsRecord = JSON.parse(response);
            let numInspections = InspectionsRecord.length;
            let newKey = 'INS' + numInspections;
            network.createInspection(newKey, req.body.SamplingCnt, req.
                .then((response) => {
                    res.send(response);
                });
        });
});
```

그림 36. 제안된 createInspection 구현 코드

[그림 37]은 어류품질검사 블록체인 네트워크에 변경기록을 블록으로 생성하는 기능을 한다. 안전성검사 체인코드에 changeInspectionOwner 함수를 호출하며 입력받은 킷값에 따라 변경되는 정보를 어류품질검사 블록체인에 블록을 생성한다. 클라이언트는 POST 메소드를 이용하여 변경정보를 바디에 넣어서 Json 형태로 전송한다.

```
app.post('/changeInspectionOwner', (req, res) => {
    network.changeInspectionOwner(req.body.key, req.body.newOwner)
        .then((response) => {
            res.send(response);
        });
});
```

그림 37 제안된 changeInspectionOwner 구현 코드

3.3.2 network.js 소스 코드

[그림 38-41]은 network.js 소스코드이며 app.js 로 들어온 요청을 처리한다. [표 13]에 정의된 4가지 메소드 방식으로 클라이언트의 요청이 있고 이 요청은 라우터 역할을 하는 app.js에서 받아서 network.js 파일의 해당 함수를 호출하게 된다. [그림38] 제안된 createInspection 구현 코드는 app.js 로 들어온 addInspect

ion Uri 로 요청된 쿼리를 처리하는 역할을 한다. POST 메소드로 요청을 받고 안정성 검사 체인코드에 createInspection 함수를 호출을 하여 어류품질검사 블록체인에 정보를 추가한다.

```
// create Inspection transaction
exports.createInspection = async function(key, SamplingCnt, SamplingWeight, InspectionItem, FishbowlNumber, InspectionResult, ValidityDate, ApplyDate,
    let response = {};
    try {

        // Create a new file system based wallet for managing identities.
        const walletPath = path.join(process.cwd(), '/wallet');
        const wallet = new FileSystemWallet(walletPath);
        console.log(`Wallet path: ${walletPath}`);

        // Check to see if we've already enrolled the user.
        const userExists = await wallet.exists(userName);
        if (!userExists) {
            console.log('An identity for the user ' + userName + ' does not exist in the wallet');
            console.log('Run the registerUser.js application before retrying');
            response.error = 'An identity for the user ' + userName + ' does not exist in the wallet. Register ' + userName + ' first';
            return response;
        }

        // Create a new gateway for connecting to our peer node.
        const gateway = new Gateway();
        await gateway.connect(ccp, { wallet, identity: userName, discovery: gatewayDiscovery });

        // Get the network (channel) our contract is deployed to.
        const network = await gateway.getNetwork('mychannel');

        // Get the contract from the network.
        const contract = network.getContract('fabcar');

        // Submit the specified transaction.
        await contract.submitTransaction('createInspection', key, SamplingCnt, SamplingWeight, InspectionItem, FishbowlNumber, InspectionResult, Validi
        console.log('Transaction has been submitted');

        // Disconnect from the gateway.
        await gateway.disconnect();

        response.msg = 'createInspection Transaction has been submitted';
        return response;

    } catch (error) {
        console.error(`Failed to submit transaction: ${error}`);
        response.error = error.message;
        return response;
    }
};
```

그림 38. 제안된 createInspection 구현 코드

[그림 39]제안된 changeInspectionOwner 구현 코드는 app.js 로 들어온 change owner/Inspection Uri 로 요청된 쿼리를 처리하는 역할을 한다. 본 함수를 수정하여 어류품질검사 블록체인 정보 값을 변경하도록 구현가능하며 본 구현에서는

어류품질검사 블록체인 정보의 소유자정보를 변경하는 기능을 하는 함수로 구현한다. PUT 메소드를 형태로 요청되며 검사정보의 소유자를 변경하는 함수이다.

```
// change Inspection owner transaction
exports.changeInspectionOwner = async function(key, newOwner) {
    let response = {};
    try {

        // Create a new file system based wallet for managing identities.
        const walletPath = path.join(process.cwd(), '/wallet');
        const wallet = new FileSystemWallet(walletPath);
        console.log(`Wallet path: ${walletPath}`);

        // Check to see if we've already enrolled the user.
        const userExists = await wallet.exists(userName);
        if (!userExists) {
            console.log('An identity for the user ' + userName + ' does not exist in the wallet');
            console.log('Run the registerUser.js application before retrying');
            response.error = 'An identity for the user ' + userName + ' does not exist in the wallet. Register ' + userName + ' first';
            return response;
        }

        // Create a new gateway for connecting to our peer node.
        const gateway = new Gateway();
        await gateway.connect(ccp, { wallet, identity: userName, discovery: gatewayDiscovery });

        // Get the network (channel) our contract is deployed to.
        const network = await gateway.getNetwork('mychannel');

        // Get the contract from the network.
        const contract = network.getContract('fabcar');

        // Submit the specified transaction.
        await contract.submitTransaction('changeInspectionOwner', key, newOwner);
        console.log('Transaction has been submitted');

        // Disconnect from the gateway.
        await gateway.disconnect();

        response.msg = 'changeInspectionOwner Transaction has been submitted';
        return response;

    } catch (error) {
        console.error(`Failed to submit transaction: ${error}`);
        response.error = error.message;
        return response;
    }
};
```

그림 39. 제안된 changeInspectionOwner 구현 코드

[그림 40]제안된 queryAllInspections 구현 코드은 app.js 로 들어온 queryAllInspections Uri 로 요청된 쿼리를 처리하는 역할을 하는 함수이다. 어류품질검사

블록체인 체인코드의 queryAllInspection 함수를 호출하며 어류품질검사 블록체인에 접근해서 모든 검사정보 자료를 조회하여 결과를 리턴 해주는 역할을 한다.

```
// query all Inspections transaction
exports.queryAllInspections = async function() {

    let response = {};
    try {
        console.log('queryAllInspections');

        // Create a new file system based wallet for managing identities.
        const walletPath = path.join(process.cwd(), '/wallet');
        const wallet = new FileSystemWallet(walletPath);
        console.log(`Wallet path: ${walletPath}`);

        // Check to see if we've already enrolled the user.
        const userExists = await wallet.exists(userName);
        if (!userExists) {
            console.log('An identity for the user ' + userName + ' does not exist in the wallet');
            console.log('Run the registerUser.js application before retrying');
            response.error = 'An identity for the user ' + userName + ' does not exist in the wallet. Register ' + userName + ' first';
            return response;
        }

        // Create a new gateway for connecting to our peer node.
        const gateway = new Gateway();
        await gateway.connect(ccp, { wallet, identity: userName, discovery: gatewayDiscovery });

        // Get the network (channel) our contract is deployed to.
        const network = await gateway.getNetwork('mychannel');

        // Get the contract from the network.
        const contract = network.getContract('fabcar');

        // Evaluate the specified transaction.
        const result = await contract.evaluateTransaction('queryAllInspections');
        //console.log(`Transaction has been evaluated, result is: ${result.toString()}`);

        return result;

    } catch (error) {
        console.error(`Failed to evaluate transaction: ${error}`);
        response.error = error.message;
        return response;
    }
};
```

그림 40. 제안된 queryAllInspections 구현 코드

[그림 41] 제안된 querySingleInspection 구현 코드는 app.js 로 들어온 Inspection?id=parameter Uri 로 요청된 쿼리를 처리하는 역할을 하는 함수이다. GET 메소드로 요청을 받고 id 파라미터를 안전성 검사 체인코드 querySingleInspection 함수에 args로 호출하여 어류품질검사 블록체인의 단일 정보를 조회한다.

```
// query the Inspection identified by key
exports.querySingleInspection = async function(key) {

    let response = {};
    try {
        console.log('querySingleInspection');

        // Create a new file system based wallet for managing identities.
        const walletPath = path.join(process.cwd(), '/wallet');
        const wallet = new FileSystemWallet(walletPath);
        console.log(`Wallet path: ${walletPath}`);

        // Check to see if we've already enrolled the user.
        const userExists = await wallet.exists(userName);
        if (!userExists) {
            console.log('An identity for the user ' + userName + ' does not exist in the wallet');
            console.log('Run the registerUser.js application before retrying');
            response.error = 'An identity for the user ' + userName + ' does not exist in the wallet. Register ' + userName + ' first';
            return response;
        }

        // Create a new gateway for connecting to our peer node.
        const gateway = new Gateway();
        await gateway.connect(ccp, { wallet, identity: userName, discovery: gatewayDiscovery });

        // Get the network (channel) our contract is deployed to.
        const network = await gateway.getNetwork('mychannel');

        // Get the contract from the network.
        const contract = network.getContract('fabcar');

        // Evaluate the specified transaction.
        console.log(key);

        const result = await contract.evaluateTransaction('queryInspection', key);
        console.log(`Transaction has been evaluated, result is: ${result.toString()}`);

        return result;

    } catch (error) {
        console.error(`Failed to evaluate transaction: ${error}`);
        response.error = error.message;
        return response;
    }
};
```

그림 41. 제안된 querySingleInspection 구현 코드

3.4 어류품질검사 블록체인 API 서비스 실행 결과

자신의 로컬PC에서 브라우저를 통해 테스트 한다. 버츄얼박스에 가상화로 구축된 블록체인 기반 어류품질검사 데이터의 무결성 서비스 서버의 IP는 192.168.0.52 이며 접속 포트는 8081이다. [표 9]에서 정의한 4가지 메소드 Uri를 직접 쓰고 접속한다. [그림 42]는 http://192.168.0.52/api/allInspections 주소로 접속하여 어류품질검사 블록체인의 모든 자료를 조회한 결과이다.

어류품질안전성검사 블록체인

요청하기

key	요청일	수조번호	검사항목	검사결과	시료중량	소유자	시료미수	유효기간	자세히
INS0	2019-11-01	C1	항생물질	양성	240	minsung	2	2020-01-01	Detail
INS1	2020-01-11	C2	항생물질	음성	300	jaemin	3	2020-03-05	Detail
INS2	2020-02-21	C3	항생물질	양성	450	gaon	2	2020-05-04	Detail
INS3	2020-01-14	C4	항생물질	음성	600	minsung	5	2020-03-04	Detail
INS4	2020-01-19	C3	항생물질	음성	650	jieun	5	2020-03-21	Detail
INS5	2020-02-21	C2	항생물질	양성	380	gaon	3	2020-03-23	Detail
INS6	2020-03-23	C4	항생물질	음성	450	jaemin	5	2020-05-21	Detail
INS7	2020-01-12	C6	항생물질	음성	560	jieun	2	2020-02-14	Detail
INS8	2020-01-15	C4	항생물질	음성	740	minsung	3	2020-04-18	Detail
INS9	2020-02-19	C3	항생물질	양성	720	jaemin	3	2020-03-21	Detail

그림 42. allInspections 실행결과

[그림43]은 Key 값을 파라미터로 전송해서 단일 블록의 자료를 조회한 결과이다.

어류품질안전성검사 블록체인

요청일	2019-11-01
수조번호	C1
검사항목	항생물질
검사결과	양성
시료중량	240
소유자	minsung
시료미수	2
유효기간	2020-01-01

목록으로 거래하기

그림 43. singleInspection 실행결과

[그림 44]는 어류품질검사 블록체인 정보를 등록하기 위한 페이지 이다. 자료를 등록하는 메소드는 PUT 방식으로 API 서버로 Form 방식으로 전송한다.

그림 44. addInspection 실행결과

[그림45]는 자료를 PUT 한 후 실제 자료가 등록된 후 등록된 정보를 다른 사용자에게 전송하기 위한 페이지이다. 쿼리를 전송하여 실제 자료가 등록된 것을 확인할 수 있다.

그림 45. changeowner 실행결과

3.5 어류품질검사 블록체인 네트워크 성능평가

리눅스재단이 진행하는 하이퍼레저 프로젝트 중 하나인 하이퍼레저 캘리퍼(Hyperledger Caliper)는 블록체인 네트워트의 성능을 테스트하는 도구이다[19]. 하이퍼레저 캘리퍼는 TPS, 트랜잭션 대기시간, 리소스 사용률 등을 보고서 형태로 작성하여 보여준다. 성능평가는 CPU E3-1230 V2 @ 3.30GHZ 의 워크스테이션에 버츄얼 박스를 설치하고 평가한다. 버츄얼박스에 가상 CPU는 4개로 설정하고 가상 메모리는 8기가를 설정한다.

어류품질검사 블록체인 네트워크 성능평가를 위해 먼저 하이퍼레저 캘리퍼를 설치한다. 설치파일은 깃업(Gitup) 통해 다운로드 받는다. 다운로드 명령어는 터미널을 열고 git clone https://github.com/hyperledger/caliper-benchmarks.git 작성한다

```
frdave@hyperledger-fablic:~$ git clone https://github.com/hyperledger/caliper-benchmarks.git
Cloning into 'caliper-benchmarks'...
remote: Enumerating objects: 267, done.
remote: Counting objects: 100% (267/267), done.
remote: Compressing objects: 100% (131/131), done.
remote: Total 4447 (delta 132), reused 196 (delta 93), pack-reused 4180
Receiving objects: 100% (4447/4447), 19.35 MiB | 6.92 MiB/s, done.
Resolving deltas: 100% (1809/1809), done.
frdave@hyperledger-fablic:~$
```

그림 46. 하이퍼레저 캘리퍼 설치

다운로드 후 caliper-benchmarks 폴더로 이동한다. git checkout v0.2.0 명령어로 태그 버전 체크아웃을 한다.

```
frdave@hyperledger-fablic:~/caliper-benchmarks$ git checkout v0.2.0
Note: checking out 'v0.2.0'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at 68fd459 fix goLang metadata paths in network files
frdave@hyperledger-fablic:~/caliper-benchmarks$
```

그림 47. 하이퍼레저 캘리퍼 버전 체크아웃

하이퍼레저 캘리퍼는 노드모듈을 이용하여 성능평가를 한다. npm init -y 명령어로 초기화 한다.

```
frdave@hyperledger-fablic:~/caliper-benchmarks$ npm init -y
Wrote to /home/frdave/caliper-benchmarks/package.json:

{
  "name": "caliper-benchmarks",
  "version": "1.0.0",
  "description": "This repository contains sample benchmarks that may be used by Caliper, a blockcha
in performance benchmark framework. For more information on Caliper, please see the [Caliper main re
pository](https://github.com/hyperledger/caliper/)",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/hyperledger/caliper-benchmarks.git"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/hyperledger/caliper-benchmarks/issues"
  },
  "homepage": "https://github.com/hyperledger/caliper-benchmarks#readme"
}
```

그림 48. 노드모듈 초기화

다음은 npm install --only=prod @hyperledger/caliper-cli@0.2.0 명령어로 하이퍼레저 캐플러의 종속성 패키지 노드모듈을 설치한다.

```
frdave@hyperledger-fablic:~/caliper-benchmarks$ npm install --only=prod @hyperledger/caliper-cli@0.2.0
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

> keccak@2.1.0 install /home/frdave/caliper-benchmarks/node_modules/keccak
> npm run rebuild || echo "Keccak bindings compilation fail. Pure JS implementation will be used."


> keccak@2.1.0 rebuild /home/frdave/caliper-benchmarks/node_modules/keccak
> node-gyp rebuild

make: Entering directory '/home/frdave/caliper-benchmarks/node_modules/keccak/build'
  CXX(target) Release/obj.target/keccak/src/addon.o
  CC(target) Release/obj.target/keccak/src/libkeccak-64/KeccakSpongeWidth1600.o
  CC(target) Release/obj.target/keccak/src/libkeccak-64/KeccakP-1600-opt64.o
  SOLINK_MODULE(target) Release/obj.target/keccak.node
  COPY Release/keccak.node
make: Leaving directory '/home/frdave/caliper-benchmarks/node_modules/keccak/build'

> secp256k1@3.8.0 install /home/frdave/caliper-benchmarks/node_modules/secp256k1
> npm run rebuild || echo "Secp256k1 bindings compilation fail. Pure JS implementation will be used."


> secp256k1@3.8.0 rebuild /home/frdave/caliper-benchmarks/node_modules/secp256k1
> node-gyp rebuild

make: Entering directory '/home/frdave/caliper-benchmarks/node_modules/secp256k1/build'
  CXX(target) Release/obj.target/secp256k1/src/addon.o
  CXX(target) Release/obj.target/secp256k1/src/privatekey.o
../src/privatekey.cc: In function 'Nan::NAN_METHOD_RETURN_TYPE privateKeyNegate(Nan::NAN_METHOD_ARGS_TYPE)':
../src/privatekey.cc:73:30: warning: ignoring return value of 'int secp256k1_ec_privkey_negate(const secp256k1_context
   secp256k1_ec_privkey_negate(secp256k1ctx, &private_key[0]);
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  CXX(target) Release/obj.target/secp256k1/src/publickey.o
  CXX(target) Release/obj.target/secp256k1/src/signature.o
  CXX(target) Release/obj.target/secp256k1/src/ecdsa.o
  CXX(target) Release/obj.target/secp256k1/src/ecdh.o
  CC(target) Release/obj.target/secp256k1/src/secp256k1-src/src/secp256k1.o
  CC(target) Release/obj.target/secp256k1/src/secp256k1-src/contrib/lax_der_parsing.o
  CC(target) Release/obj.target/secp256k1/src/secp256k1-src/contrib/lax_der_privatekey_parsing.o
  SOLINK_MODULE(target) Release/obj.target/secp256k1.node
  COPY Release/secp256k1.node
make: Leaving directory '/home/frdave/caliper-benchmarks/node_modules/secp256k1/build'
npm notice created a lockfile as package-lock.json. You should commit this file.
+ @hyperledger/caliper-cli@0.2.0
```

그림 49. 종속성 노드모듈 패키지 설치

npx caliper bind --caliper-bind-sut fabric --caliper-bind-sdk 1.4.4 명령어를 통해 캘리퍼 SDK를 바인딩 한다.

어류품질검사 블록체인 네트워크 성능평가를 하기 전에 2가지 사전준비사항이 있다. 첫 번째로 본 연구에서 작성한 어류품질검사 블록체인 체인코드 inspection.go파일을 /caliper-benchmarks/src/fabric/scenario/inspection/go/ 폴더에 복사해서 넣는다.

```
frdave@hyperledger-fablic:~/caliper-benchmarks/src/fabric/scenario/inspection$ tree go
go
├── go.mod
├── go.sum
└── inspection.go

0 directories, 3 files
```

그림 50. 어류품질검사 체인코드 적재

다음은 [표 13] 어류품질검사 블록체인 트랜잭션 처리 메소드를 작성한 자바스크립트 소스코드를 /caliper-benchmarks/benchmarks/scenario/inspection/ 폴더로 복사한다.

```
frdave@hyperledger-fablic:~/caliper-benchmarks/benchmarks/scenario$ tree inspection
inspection
├── changeInspectionOwner.js
├── config.yaml
├── createInspection.js
├── helper.js
├── queryAllInspections.js
└── queryInspection.js

0 directories, 6 files
```

그림 51. 어류품질검사 블록체인 메소드 적재

다음은 [그림 52]에 /caliper-benchmarks/networks/fabric/fabric-v1.4.1/3org2pergoleveldb/fabric-go-tls.yaml 파일을 열어서 채널 키 하위에 체인코드 키를 찾아서 path 키에 어류품질검사 블록체인 체인코드가 있는 위치fabric/scenario/inspection/go를 지정해준다.

```yaml
channels:
  mychannel:
    configBinary: networks/fabric/config_solo/mychannel.tx
    created: false
    orderers:
    - orderer.example.com
    peers:
      peer0.org1.example.com:
        eventSource: true
      peer0.org2.example.com:
        eventSource: true
      peer0.org3.example.com:
        eventSource: true

    chaincodes:
    - id: inspection
      version: v0
      language: golang
      path: fabric/scenario/inspection/go
```

그림 52. 캘리퍼 fabric-go-tls.yaml

다음은 [그림 53] /caliper-benchmarks/benchmarks/scenario/inspection/config.yaml 파일을 열어서 위에서 복사한 체인코드 메소드 파일의 위치를 작성한다. assets 은 총 100개를 지정한다.

```yaml
test:
  name: inspection
  description: This is an Inspection benchmark for caliper, to test the backend DLT's
    performance with Inspection account opening & querying transactions
  clients:
    type: local
    number: 5
  rounds:
  - label: Change inspection owner.
    txDuration:
    - 30
    rateControl:
    - type: fixed-backlog
      opts:
        unfinished_per_client: 5
    arguments:
      assets: 100
    callback: benchmarks/scenario/inspection/changeInspectionOwner.js
  - label: Query all inspection.
    txDuration:
    - 30
    rateControl:
    - type: fixed-backlog
      opts:
        unfinished_per_client: 5
    arguments:
      assets: 100
      startKey: '1'
      endKey: '50'
    callback: benchmarks/scenario/inspection/queryAllInspections.js
  - label: Query a inspection.
    txDuration:
    - 30
    rateControl:
    - type: fixed-backlog
      opts:
        unfinished_per_client: 5
    arguments:
      assets: 100
    callback: benchmarks/scenario/inspection/queryInspection.js
  - label: Create a inspection.
    txDuration:
    - 30
    rateControl:
    - type: fixed-backlog
      opts:
        unfinished_per_client: 5
    callback: benchmarks/scenario/inspection/createInspection.js
monitor:
  type:
  - docker
  - process
  docker:
    name:
    - all
  process:
  - command: node
    arguments: local-client.js
    multiOutput: avg
  interval: 1
```

그림 53. 캘리퍼 config.yaml

다음은[그림 54]에 npx caliper benchmark run --caliper-workspace . --caliper-benchconfig benchmarks/scenario/inspection/config.yaml --caliper-networkconfig networks/fabric/fabric-v1.4.1/3org2peergoleveldb/fabric-go-tls.yaml 명령어로 갤리퍼를 실행 한다.

```
frdave@hyperledger-fablic:~/caliper-benchmarks$ npx caliper benchmark run --caliper-workspace .
 --caliper-benchconfig benchmarks/scenario/inspection/config.yaml --caliper-networkconfig netwo
rks/fabric/fabric-v1.4.1/3org2peergoleveldb/fabric-go-tls.yaml
Benchmark for target Blockchain type fabric about to start
2020.07.06-13:38:27.818 info  [caliper] [benchmark-validator]   No observer specified, will def
ault to `none`
2020.07.06-13:38:27.820 info  [caliper] [caliper-flow]  ####### Caliper Test #######
2020.07.06-13:38:27.831 info  [caliper] [null-observer]         Configured observer
2020.07.06-13:38:27.838 info  [caliper] [caliper-utils]         Executing command: cd /home/frd
ave/caliper-benchmarks;export FABRIC_VERSION=1.4.1;docker-compose -f networks/fabric/docker-com
pose/3org2peergoleveldb/docker-compose-tls.yaml up -d;sleep 3s
Creating network "3org2peergoleveldb_default" with the default driver
Creating ca.org3.example.com ...
Creating ca.org2.example.com ...
Creating orderer.example.com ...
Creating ca.org1.example.com ...
Creating ca.org3.example.com
Creating ca.org2.example.com
Creating orderer.example.com
Creating orderer.example.com ... done
```

그림 54. 캘리퍼 실행

캘리퍼는 실행결과로 성능평가보고서를 생성 한다.

Summary

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
Change inspection owner.	1037	0	34.6	1.28	0.15	0.56	34.0
Query all inspection.	2877	0	95.7	0.96	0.02	0.17	95.5
Query a inspection.	3117	0	103.7	0.81	0.01	0.16	103.4
Create a inspection.	1099	0	36.5	1.22	0.17	0.54	36.3

그림 55. 어류품질검사 블록체인 성능 Summary 보고서

[그림 55]에 Query all inspection을 보면 총 전송률이 초당 95.7으로 최고 지연은 0.96, 최저 0.02, 평균 0.17로 지연이 거의 없음을 알 수 있다. Query all inspection은 GET 메소드로 블록을 단지 읽기만 하므로 지연이 거의 발생하지 않는다. Change inspection owner 실패는 없으며 총 100개의 안전성 검사블록을 한번에 트랜잭션 하는데 최고 지연시간은 1.28초 이며 최소 지연시간은 0.15이며 평균 0.56의 지연시간을 보여주고 있다. Chage inspection owner 와 Cretae a inspection은 PUT 메소드로 블록의 검증과정을 거치므로 지연시간이 GET 메소드보다 더 걸리는걸 알 수 있다.

Change inspection owner.

Performance metrics

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
Change inspection owner.	1037	0	34.6	1.28	0.15	0.56	34.0

Resource consumption

Type	Name	Memory(max)	Memory(avg)	CPU% (max)	CPU% (avg)	Traffic In	Traffic Out	Disc Read	Disc Write
Docker	dev-peer0.org3.example.com-inspection-v0	6.8MB	6.6MB	7.18	2.96	2.9MB	1.2MB	0B	0B
Docker	dev-peer0.org1.example.com-inspection-v0	6.8MB	6.6MB	8.51	2.93	2.9MB	1.2MB	0B	0B
Docker	dev-peer0.org2.example.com-inspection-v0	7.0MB	6.8MB	8.08	3.01	2.9MB	1.2MB	0B	0B
Docker	peer0.org2.example.com	179.2MB	161.8MB	37.63	18.91	10.7MB	5.7MB	0B	12.8MB
Docker	peer0.org1.example.com	180.4MB	163.9MB	36.37	19.07	10.7MB	6.7MB	0B	12.8MB
Docker	peer0.org3.example.com	177.1MB	160.4MB	40.66	19.29	10.7MB	5.7MB	0B	12.8MB
Docker	ca.org1.example.com	8.5MB	8.5MB	0.46	0.01	371B	0B	0B	0B
Docker	ca.org3.example.com	8.5MB	8.5MB	0.01	0.00	371B	0B	0B	0B
Docker	ca.org2.example.com	7.7MB	7.7MB	0.01	0.00	371B	0B	0B	0B
Docker	orderer.example.com	213.4MB	82.7MB	37.26	11.02	10.4MB	24.9MB	0B	8.8MB

그림 56. 어류품질검사 블록체인 성능 보고서(Change inspection owner)

Query all inspection.

Performance metrics

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
Query all inspection.	2877	0	95.7	0.96	0.02	0.17	95.5

Resource consumption

Type	Name	Memory(max)	Memory(avg)	CPU% (max)	CPU% (avg)	Traffic In	Traffic Out	Disc Read	Disc Write
Docker	dev-peer0.org3.example.com-inspection-v0	7.1MB	7.0MB	15.35	11.85	4.8MB	1.8MB	0B	0B
Docker	dev-peer0.org1.example.com-inspection-v0	7.1MB	7.0MB	14.66	11.59	4.8MB	1.8MB	0B	0B
Docker	dev-peer0.org2.example.com-inspection-v0	7.3MB	7.2MB	16.91	12.13	4.9MB	1.9MB	0B	0B
Docker	peer0.org2.example.com	180.0MB	179.8MB	46.30	40.50	5.7MB	8.3MB	0B	0B
Docker	peer0.org1.example.com	181.2MB	181.0MB	49.24	40.77	5.7MB	9.2MB	0B	0B
Docker	peer0.org3.example.com	177.9MB	177.6MB	45.92	40.02	5.6MB	8.2MB	0B	0B
Docker	ca.org1.example.com	8.5MB	8.5MB	0.00	0.00	70B	0B	0B	0B
Docker	ca.org3.example.com	8.5MB	8.5MB	0.00	0.00	70B	0B	0B	0B
Docker	ca.org2.example.com	7.7MB	7.7MB	0.00	0.00	70B	0B	0B	0B
Docker	orderer.example.com	222.1MB	221.6MB	1.54	0.14	100.1KB	88.4KB	0B	0B

그림 57. 어류품질검사 블록체인 성능 보고서(Query all inspection)

Query a inspection.

Performance metrics

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
Query a inspection.	3117	0	103.7	0.81	0.01	0.16	103.4

Resource consumption

Type	Name	Memory(max)	Memory(avg)	CPU% (max)	CPU% (avg)	Traffic In	Traffic Out	Disc Read	Disc Write
Docker	dev-peer0.org3.example.com-inspection-v0	7.2MB	7.2MB	12.57	9.55	4.7MB	1.6MB	0B	0B
Docker	dev-peer0.org1.example.com-inspection-v0	7.1MB	7.1MB	12.97	9.55	4.7MB	1.6MB	0B	0B
Docker	dev-peer0.org2.example.com-inspection-v0	7.3MB	7.3MB	12.78	9.91	4.7MB	1.6MB	0B	0B
Docker	peer0.org2.example.com	180.1MB	180.0MB	48.28	38.57	5.6MB	8.8MB	0B	0B
Docker	peer0.org1.example.com	181.3MB	181.3MB	45.09	38.99	5.7MB	9.1MB	0B	0B
Docker	peer0.org3.example.com	177.9MB	177.9MB	47.23	39.03	5.6MB	8.8MB	0B	0B
Docker	ca.org1.example.com	8.5MB	8.5MB	0.00	0.00	371B	0B	0B	0B
Docker	ca.org3.example.com	8.5MB	8.5MB	0.00	0.00	371B	0B	0B	0B
Docker	ca.org2.example.com	7.7MB	7.7MB	0.00	0.00	371B	0B	0B	0B
Docker	orderer.example.com	221.4MB	221.4MB	0.06	0.02	1.2KB	766B	0B	0B

그림 58. 어류품질검사 블록체인 성능 보고서(Query a inspection)

Create a inspection.

Performance metrics

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
Create a inspection.	1099	0	36.5	1.22	0.17	0.54	36.3

Resource consumption

Type	Name	Memory(max)	Memory(avg)	CPU% (max)	CPU% (avg)	Traffic In	Traffic Out	Disc Read	Disc Write
Docker	dev-peer0.org3.example.com-inspection-v0	7.2MB	7.2MB	6.72	4.87	1.9MB	714.6KB	0B	0B
Docker	dev-peer0.org1.example.com-inspection-v0	7.1MB	7.1MB	7.34	4.85	1.9MB	721.5KB	0B	0B
Docker	dev-peer0.org2.example.com-inspection-v0	7.3MB	7.3MB	7.60	4.95	1.9MB	724.6KB	0B	0B
Docker	peer0.org2.example.com	185.8MB	182.9MB	39.70	32.71	7.4MB	3.8MB	0B	8.5MB
Docker	peer0.org1.example.com	187.1MB	184.2MB	37.21	32.04	7.4MB	4.0MB	0B	8.5MB
Docker	peer0.org3.example.com	183.7MB	180.8MB	37.08	32.44	7.4MB	3.8MB	0B	8.5MB
Docker	ca.org1.example.com	8.5MB	8.5MB	0.00	0.00	0B	0B	0B	0B
Docker	ca.org3.example.com	8.5MB	8.5MB	0.01	0.00	0B	0B	0B	0B
Docker	ca.org2.example.com	7.7MB	7.7MB	0.00	0.00	0B	0B	0B	0B
Docker	orderer.example.com	253.2MB	240.3MB	27.71	20.95	7.5MB	17.8MB	0B	6.1MB

그림 59. 어류품질검사 블록체인 성능 보고서(Create a inspection)

V. 결론

최근 블록체인을 이용한 다양한 서비스들이 쏟아져 나오고 있으며 많은 기관들이 안전한 시스템 구축을 하면서도 유지비용이 저렴한 블록체인 기술에 대해 연구하고 있다. 무역, 의료, 교육 등 여러 분야에 쓰이고 있는 블록체인기술을 어류안전성 검사에 도입하여 검사결과의 위변조를 방지하고 신뢰도를 높힐 수 방안을 소개 하였다.

블록체인을 활용함으로써 각 기관 간에 상호감시체제를 형성하고 무결성 검사결과를 전파 할 수 있어 근본적으로 검사결과의 허위결과가 생성 되는 것을 방지할 수 있고 무결성, 기밀성, 신뢰성, 부인방지 등이 보장된다는 점에 의의가 있다. 이에 대해 부가적인 기대 효과는 다음과 같다.

구매 희망자의 경우 신뢰성 있는 어류상품을 구매할 수 있으므로 만족도가 높아진다. 판매 희망자의 경우 인증을 통해 진행되므로 데이터를 한 번 더 확인 할 수 있으며 검사기관의 경우 다수의 유통과정이 발생하고 여러 가지 안전성 검사결과를 등록함으로 인해 발생할 수 있는 혼선과 검사자 실수를 줄일 수 있다. 또한 검사신뢰도가 높을수록 검사기관의 신뢰도 역시 높아진다. 승인기관 역시 판매자 , 구매자의 신뢰도를 얻기 위해 추가적인 방책을 운영하는데 소모 되는 비용을 줄일 수 있으므로 제안된 방법을 통해 돈으로 환산할 수 없는 장점을 가지고 있다고 할 수 있다.

어류안전성 검사 블록체인시스템은 어류뿐만 아니라 유사한 농수산물, 식품 또한 적용될 가능성이 높다고 볼 수 있다. 본 서비스를 통하여 효율적인 유통구조 혁신이 가능하며 차후 안전성 검사 블록을 유통이력제에 도입하여 소비자가 물품을 구매할 때 유통이력뿐만 아니라 안전성 검사 이력까지 확인 가능하게 한다면 유통망 신뢰도와 소비자 만족도도 증가 할 것으로 판단된다.

참고문헌

[1] 도호쿠 지방 태평양 해역 지진의 피해
https://ko.wikipedia.org/wiki/도호쿠_지방_태평양_해역_지진의_피해
[2] 농수산물 품질관리법 제 61조
http://www.law.go.kr/법령/농수산물품질관리법/(16277,20190115)
[3] 수산물 안전성조사업무 처리요령
http://law.go.kr/LSW/admRulLsInfoP.do?admRulSeq=2000000024413
[4] 해양수산부 국립수산물품질관리원
https://www.nfqs.go.kr/
[5] Beyond Bitcoin. “BlockChain Technology.” (2015)
[6] Omar Dib, Kei-Leo Brousmiche, Antoine Durand, Eric Thea, Elyes Ben Hamida: Consortium Blockchains: Overview, Applications and Challenges
[7] Smart Contract
https://ko.wikipedia.org/wiki/%EC%8A%A4%EB%A7%88%ED%8A%B8_%EA%B3%84%EC%95%BD
[8] Hyperledger Fabric & composer
https://www.hyperledger.org/projects/composer
[9] 민경세, 신예인. (2018). 미래 식품안전망 강화를 위한 블록체인 활용 연구.식품안전정보원 연구보고서. 3(2), 32,34
[10] 소프트웨어 정책연구소
https://spri.kr/posts/view/21927?code=industry_trend
[11] MIT NEWS(2017.10.17.)
[12] 허재욱, 김정호, 전문석 “블록체인 네트워크 기반의 도메인 네임 시스템 설계 및 구현”
[13] Go (프로그래밍 언어)
https://ko.wikipedia.org/wiki/Go_(%ED%94%84%EB%A1%9C%EA%B7%B8%EB

%9E%98%EB%B0%8D_%EC%96%B8%EC%96%B4)

[14] Go(프로그래밍 언어)

https://namu.wiki/w/Go(%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D%20%EC%96%B8%EC%96%B4)

[15] Node.js

https://ko.wikipedia.org/wiki/Node.js

[16] Node.js

https://namu.wiki/w/Node.js

[17] 도커 (소프트웨어)

https://ko.wikipedia.org/wiki/%EB%8F%84%EC%BB%A4_(%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4)

[18] 도커 - 공식 웹사이트

https://www.docker.com/

[19] 해시넷

http://wiki.hash.kr/index.php/%ED%95%98%EC%9D%B4%ED%8D%BC%EB%A0%88%EC%A0%80_%EC%BA%98%EB%A6%AC%ED%8D%BC