International Journal of Innovative
Computing, Information and Control
Volume **18**, Number **5**, October **2022**

ICIC International ©2022 ISSN 1349-4198
pp. **1617–1632**

HYBRID APPROACH OF QUICK RESPONSE CODE AND NON-FUNGIBLE TOKEN IN PRIVATE PERMISSIONED BLOCKCHAIN FOR ANTI-COUNTERFEITING

Wilson Philips and Arya Wicaksana*

Department of Informatics
Universitas Multimedia Nusantara
Jl. Scientia Boulevard, Gading Serpong, Tangerang 15810, Indonesia
wilson4@student.umn.ac.id; *Corresponding author: arya.wicaksana@umn.ac.id

Received April 2022; revised July 2022

Abstract. *Counterfeit goods have challenged the government, business owners, and intellectual property holders, and the establishment of blockchain allowed the creation of digital assets of the physical good stored and distributed securely among participants to answer the problem. The non-fungible token (NFT) combined with a quick response code (QR code) is resourceful in digitally and physically identifying and verifying the physical goods' origin and authenticity. This study further explores the potential of the private permissioned blockchain in creating, storing, and managing the digital assets representation of the physical goods in the form of NFTs on a semi-decentralized application (DApp) named Oricon. This hybrid approach is a new way for low-cost anti-counterfeiting aimed at various physical goods. Test and evaluation show that the latency of the Oricon is 1.028s, and the throughput is 18.2 transactions per second (TPS). The parallel processing of transactions gives higher throughput with 20.454 TPS and a latency of 1.169s. This work shows that with its proven performance and scalability characteristics, Oricon is a promising solution for further deployment and anti-counterfeiting in various industries.*

Keywords: Anti-counterfeiting, Digital asset, NFT, Permissioned blockchain, QR code, Semi-decentralized application

1. **Introduction.** Counterfeit goods bring problems to various industries and harm the economy on a global scale ranging from cosmetics to pharmaceuticals and even luxury goods [1]. The global economic value of counterfeiting and piracy is predicted to reach $2.3 trillion by 2022, significantly increasing from estimated $1.1 trillion in 2013 [2]. According to [3], the damage to the global economy caused by counterfeit goods is estimated to exceed $300 billion in 2018. The reasons which caused counterfeit goods to spread widely are poor governance and misuse of trade solutions [4]. With e-commerce, counterfeit goods become easily accessible to the public. The existence of the counterfeit goods industry negatively impacts parties associated with it. In addition, the counterfeit goods industry also legalizes violations of human labor, human trafficking, and other criminal activities [5].

The emergence of blockchain technology has transformed many industries, such as financial services, healthcare, and government, into having greater transparency and accountability [6-8]. Related works include utilizing blockchain technology for anti-counterfeiting as in [9-13]. In [12-14], a study was conducted to overcome the spread of counterfeit goods in the wine industry by implementing blockchain and near-field communication

DOI: 10.24507/ijicic.18.05.1617

(NFC) on a legacy supply chain management system called NAS (NFC-enabled anti-counterfeiting system) [12]. The study focused on ensuring the integrity of the physical goods during their distribution in the supply chain and came out with dNAS (Decentralized NFC-enabled anti-counterfeiting system). The proposed dNAS system does not have a digital asset management feature and is tailored for the wine industry. It also carries potential security vulnerability due to its architectural design. A similar study on anti-counterfeiting in the wine industry in [10] proposed a blockchain system design tailored for anti-counterfeiting with the case study of the wine industry. Aside from wine counterfeiting, a study in [9] developed a blockchain solution powered by Ethereum and utilized a quick response (QR) code. Ethereum 1.0 suffers from high transaction fees to execute transactions [15]. The study is limited to the design and implementation of the solution without testing and evaluation details. Another study [11] proposed a new blockchain solution tailored to the anti-counterfeiting application. The system came out with its novel consensus protocol powered by Tendermint and utilizes NFC for the anti-counterfeiting part.

The main contribution of this study is the hybrid approach of quick response (QR) code and non-fungible token (NFT) in a private permissioned blockchain such as Hyperledger Fabric for anti-counterfeiting. The QR code serves as a low-cost physical goods identification and verification tag, printed or embedded physically on the goods. The digital assets in the form of NFTs have been created based on the physical goods by the manufacturers in the first place using the proposed solution. The proposed solution is named Oricon, and the architecture is designed to be a semi-decentralized system tailored for low-cost and general-purpose usage by companies and manufacturers. The Hyperledger Fabric is an open-source permissioned blockchain platform to reach many enterprise use cases [16]. The consensus protocols do not require native cryptocurrencies to incentivize costly mining or pay for smart contract execution. Instead, it can be developed within smart contracts using token standards, such as the ERC-20 token standard [17]. Thus, Oricon's main features are to create, store, and manage digital assets of the physical goods registered by the manufacturers and for the customers to verify and validate the authenticity and origin of the physical goods. ORC tokens are created and circulated to carry out transactions on the Hyperledger Fabric blockchain (execute smart contracts). The performance is evaluated based on throughput and latency using Hyperledger Caliper.

The rest of this paper is divided into four sections. Section 2 presents literature related to this work. Section 3 presents the methods, and Section 4 presents the results and analysis. Finally, Section 5 discusses and concludes the work conferred in this paper.

2. **Preliminaries.**

2.1. **Hyperledger Fabric blockchain.**

Hyperledger Fabric is an open source blockchain platform used to deploy and operate permissioned blockchains, which offers practical configuration because of its modular and extensible system [18]. There are several components composing a Hyperledger Fabric network such as Membership Service Provider (MSP), Ordering Service Node (OSN), and the clients [5,12], shown in Figure 1.

The ledger consists of two parts: world state and blockchain. World state is a database that holds current values of a set of ledger states, represented by key-value pairs. Blockchain is a transaction log storing all changes applied to ledger states that determines the current values. In other words, the world state is derived from the blockchain. There are three types of nodes participated in the network, as follows.

1) Clients: Function to send transaction proposals to be executed, collect endorsements, and broadcast transaction messages created.

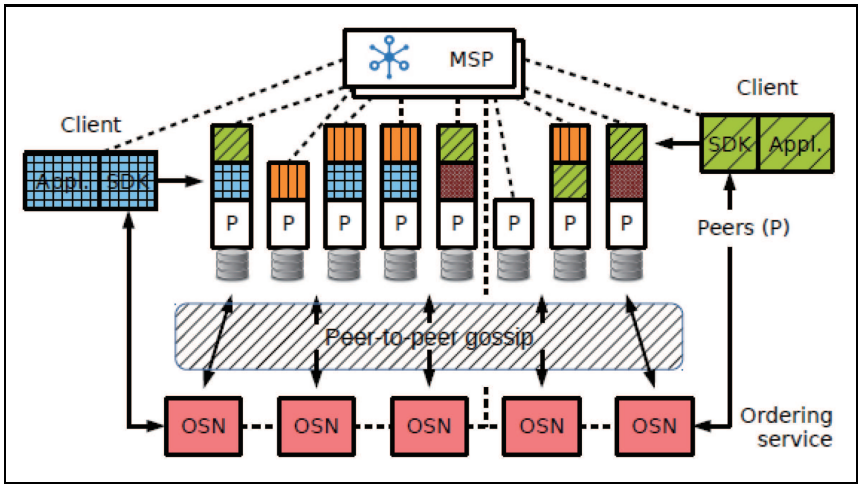

FIGURE 1. Network components [16]

2) Peers: Function to execute transaction proposals and validate transactions. Each peer holds a copy of the ledger, which includes both world state and blockchain. Not all peers execute the transaction proposals, though, only several peers defined in the endorsement policy do so. Peers which execute transactions are called endorsing peers or endorsers, and peers which validate transactions are called committing peers.
3) Ordering Service Nodes: Function to strictly order a set of transactions, put them into blocks, and propagate the blocks to peers who are connected to them. Ordering Service Nodes are called orderers.

MSP stores all user identities on the network, which functions to provide the identity nodes required for transaction authentication, verify transaction integrity, sign endorsements, validate endorsement, and perform other authentication operations needed. MSP is used to define each authorized user on the network and its access control.

All peers are connected by peer-to-peer gossip protocol, which function as follows.

- Manage peer discovery and channel membership by continually identifying each availability to detect active and non-active peers.
- Disseminate ledger data across all peers in a channel, so that peers with missing blocks can synchronize themselves.
- Perform peer-to-peer state transfer updates to newly connected peers.

2.2. **Raft consensus algorithm.** Raft is a consensus algorithm that works by managing replicated logs. Replicated log is an implementation of replicated state machines, which is managed by a set of servers that compute identical copies of the state and can continue to operate even if the server is down. Each server keeps a log containing a set of commands executed by the state machine sequentially. In Hyperledger Fabric, each orderer has a finite state machine used to ensure the order of log entries which contains transactions is deterministic in each orderer. Raft is crash fault tolerant, if the majority of orderers running (which is called quorum) are $n/2 + 1$, where n is the number of participating orderers (which is called consenter set).

Each orderer has three possible states, which acts as a leader, follower, or candidate. There is only a leader and the rest are followers. A follower functions to receive requests from the leader and candidates. A leader functions to handle all requests sent by the client (if there is a request sent from the client to the follower, the follower will send it back to the leader). Candidates function to nominate themselves during the election, one

of which will be selected as a leader. Communication between orderers is done through remote procedure call (RPC) protocol [19].

2.3. **Performance metrics.** The common metrics used to measure the blockchain perfomance are transaction latency and throughput, which are defined as follows [20].

Definition 2.1. *Transaction latency (X) defines the time it takes for a transaction to be available on the network starting from the transaction is submitted, which includes propagation and consensus time. This metric is measured in seconds (s) and expressed as in (1).*

$$X = (\textit{Confirmation time @ network threshold}) - \textit{submit time} \tag{1}$$

Definition 2.2. *Transaction throughput (Y) defines the number of valid transactions that were successfully committed on the blockchain for each node within a certain time period. This metric is measured in transactions per second (TPS) and expressed as in (2).*

$$Y = \textit{Total committed transactions/total time in seconds @ \# committed nodes} \tag{2}$$

Success rates are also included in the perfomance evaluations obtained from Hyperledger Caliper which contain the number of successful and failed transactions in a test cycle [21]. Successful transactions are defined as transactions that were committed to the ledger and have valid statuses. Failed transactions are transactions that were not committed to the ledger (failed at endorsement phase), or were committed to the ledger but have invalid statuses (failed at validation phase).

3. Methods.

3.1. **Oricon design features.** Oricon's primary purpose is to be an anti-counterfeiting application that suits various physical goods, not only for specific industries. The design features are universal and accommodating for various physical goods, yet straightforward to keep it low-cost (no complex smart contracts). The functionalities of Oricon are to identify and verify physical goods with its digital assets created at manufacture and stored in the Hyperledger Fabric blockchain.

The features for product (physical good) management are described in Table 1, user management in Table 2, and ORC tokens in Table 3. Product data stored on-chain (in the blockchain) consist of product code, transaction ID, name, price, country of origin, release date, description, and image. Product code is a unique value used as a key stored in the world state. In the verification process of physical goods, the QR code is scanned to retrieve the corresponding transaction ID, which represents a unique identifier of each submitted transaction. Images are saved as Base64 strings, and the whole product data are saved in JSON.

Table 1. Product management features

Feature	Description	Access control		
		A	C	T
Create product	Store new physical goods	v	–	–
Read product	Retrieve product details	v	–	–
Update product	Update product details	v	–	–
Delete product	Mark deleted product	v	–	–
Read product history	Retrieve product history based on updates that have been made	v	–	–
Search product	Search product scanned via QR codes	v	v	–

Table 2. User management features

Feature	Description	Access control		
		A	C	T
Register user	Register new users to CA	v	–	–
Enroll user	Import registered user identity	v	v	–
Login	Authorize users	v	v	–
Get all users	Retrieve all registered users identities	v	–	–

Table 3. ORC token features

Feature	Description	Access control		
		A	C	T
Get token details	Retrieve token details (name, symbol, decimals, total supply) and the user balance. Total supply is returned only for the token admin.	v	v	v
Mint token	Add the amount of the total token supply	–	–	v
Transfer token	Transfer the amount of tokens	v	v	v
Approve token	Give allowances	v	v	v
Get allowance token	Retrieve the number of allowances	v	v	v
Transfer from token	Transfer allowances	v	v	–

ORC tokens are implemented on smart contracts as fungible tokens and transaction fees using the ERC-20 token standard. The ORC tokens are created with no supply caps defined yet, and the supply of ORC tokens follows supplies and demands from the companies and customers. When executing a smart contract to make transactions, one ORC token gets burned (transferred from the customer's wallet to the token administrator's wallet). Oricon's smart contract enforces access control over the features based on each user role. There are three user roles: company administrators (denoted by A), customers (denoted by C), and token administrators (denoted by T).

3.2. **System architecture.** The semi-decentralized architecture is chosen for Oricon to avoid centralized client-server architecture inherited problems while preserving control for the ORC token and trusted party of the companies. Thus, the system architecture comprises an application, a central server (Node.js/web and database), and the blockchain network, as shown in Figure 2. This semi-decentralized architecture also limits points of access to crucial information such as encryption keys and users' wallets. Thus, vulnerabilities are limited to the application installed on the client-side (customers). The central server serves as a gateway between the application and the blockchain, ensuring only eligible Oricon users and defined operations to be executed. The blockchain network is private and requires permission to access, which limits security threats to it.

The application interacts with the Node.js server using REST API to communicate with the Hyperledger Fabric via HTTP protocols. The REST API is implemented using Express.js, with JSON requests and responses. Hyperledger Fabric v1.4.8 is used in this study and established in a virtual machine with specifications presented in Table 4. The CouchDB is used as the database on the Node.js server to store information on user wallets (each containing a public key, private key, and digital certificate) and account credentials (each containing username and password).

The designed network consists of one channel that connects applications (Node.js server), endorsers, and orderers, as shown in Figure 3. The number of endorsers and orderers

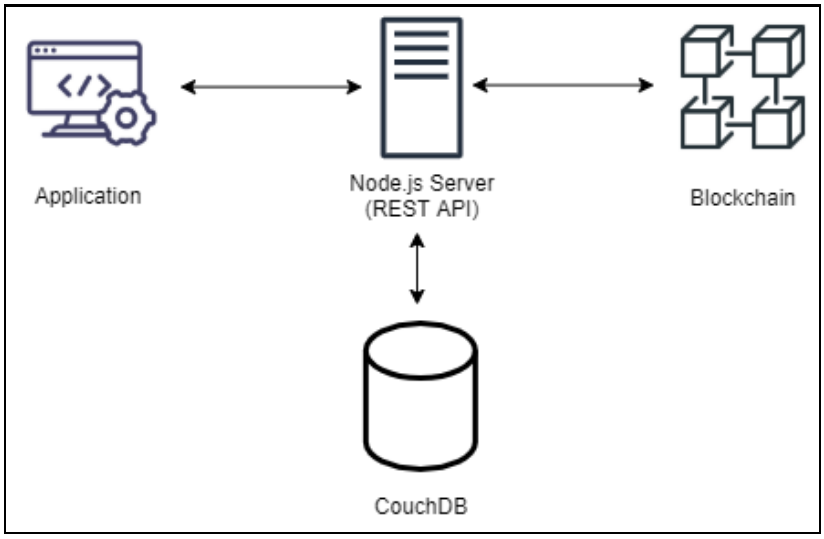

FIGURE 2. System architecture

TABLE 4. Virtual machine specification

Operating system	Linux Ubuntu 20.4 64-bit
Central processing unit	1.80 GHz (4 CPUs) @ 2.0 GHz
Random access memory	4 GB
Solid state drive	50 GB

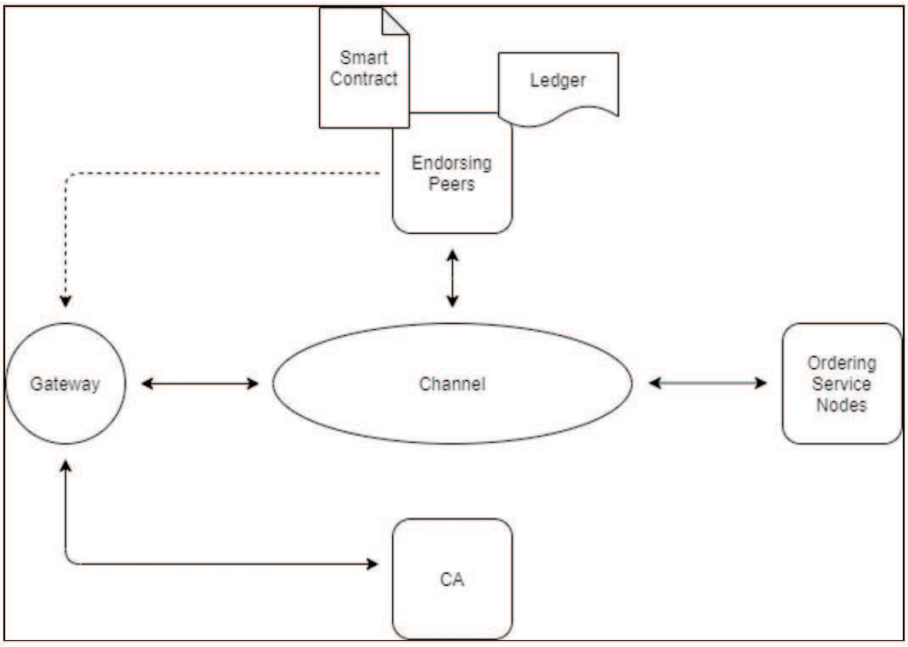

FIGURE 3. Network design

used is two endorsers and five orderers; authorized users from a particular company control this private network. The identity of each user is registered to a certificate authority (CA) by the company administrator. In order to connect to the channel, the application uses an identity required by Fabric Gateway to communicate with the network through a gRPC connection.

Each endorser holds copies of the smart contract and ledger to perform endorsement and validation. The ledger world state used is CouchDB, which allows smart contracts to perform complex queries. The application will connect to an endorser using service discovery and run the smart contract to create a transaction proposal. If the transaction proposal performs queries, a transaction response is returned with the results of the smart contract execution. Suppose the transaction proposal performs updates to the ledger states. In that case, the application collects all required endorsements from endorsers and puts them into a transaction message to be sent to an orderer. The endorsement policy defined in the smart contract is Org1.peer, which needs endorsement from any peer of the Org1 MSP.

All transactions received by orderers are strictly ordered through consensus and packaged into blocks to be sent to a leader peer. Then, the leader peer propagates the block to other peers using the peer-to-peer gossip protocol. All transactions in the block are validated and committed to the ledger on each peer. The peer connected to the event hub will send Chaincode events to the application to notify committed transactions.

3.3. **Implementation.** The user interface is implemented for cross-platform mobile applications using Ionic Angular. To enforce access controls at the client-side application, route guards prevent access to specific routes if the required user role is not met. On Login Page, users can input their account credentials to log in to the application. Users (companies and manufacturers) can enroll their accounts to import the identities on the Enrollment Page. These pages are shown in Figure 4.

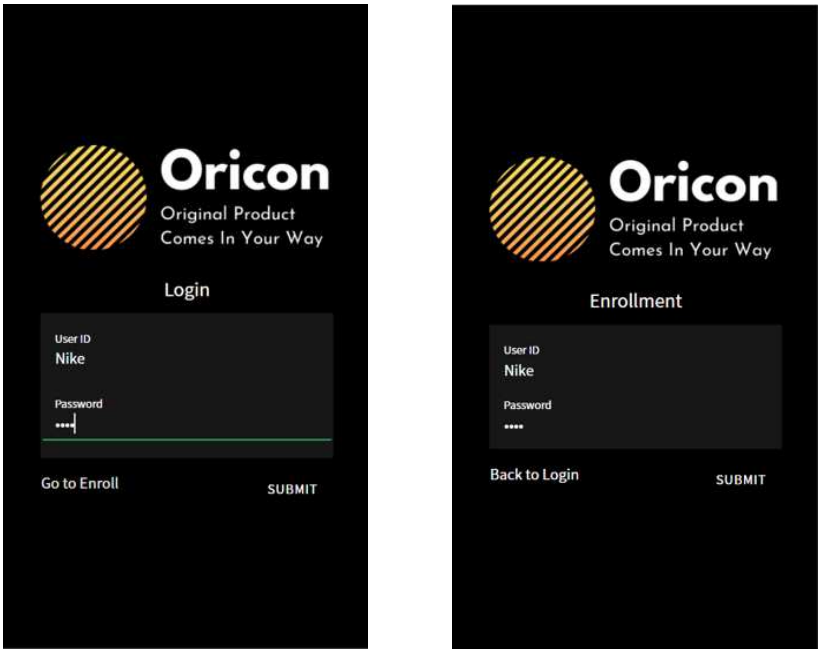

FIGURE 4. Login and enrollment page

Users can view all stored physical goods on the Product Catalog Page, and the selected product routes the application to each Product Details Page. These pages are shown in Figure 5. Users can scan a product using the QR code to display the product details on the Search Product Page. This page is shown in Figure 6.

Users can view a list of all registered users on the User Management Page and register new users. This page is shown in Figure 7. Users can manage their ORC tokens on the Token Wallet Page to view the allowances given, transfer tokens, transfer from tokens,

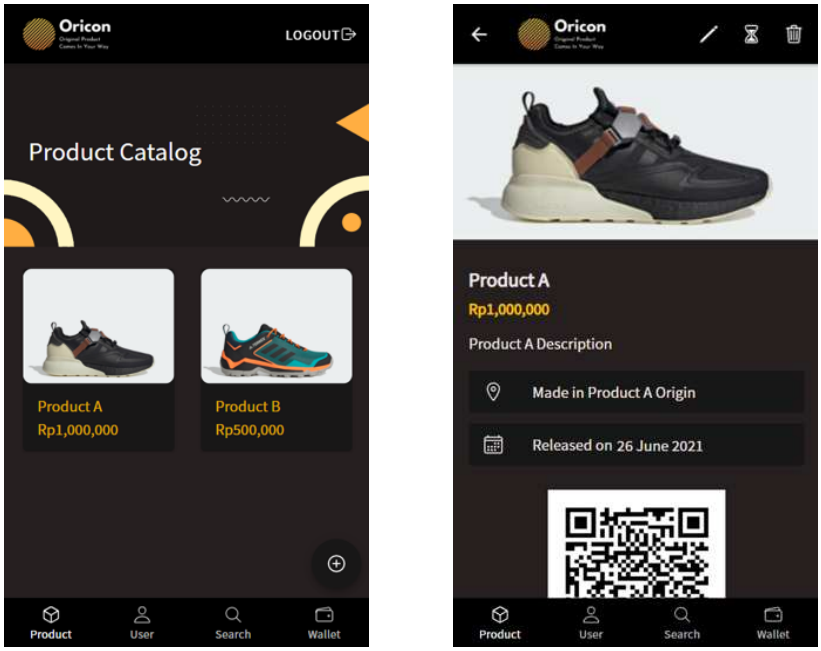

FIGURE 5. Product catalog and details

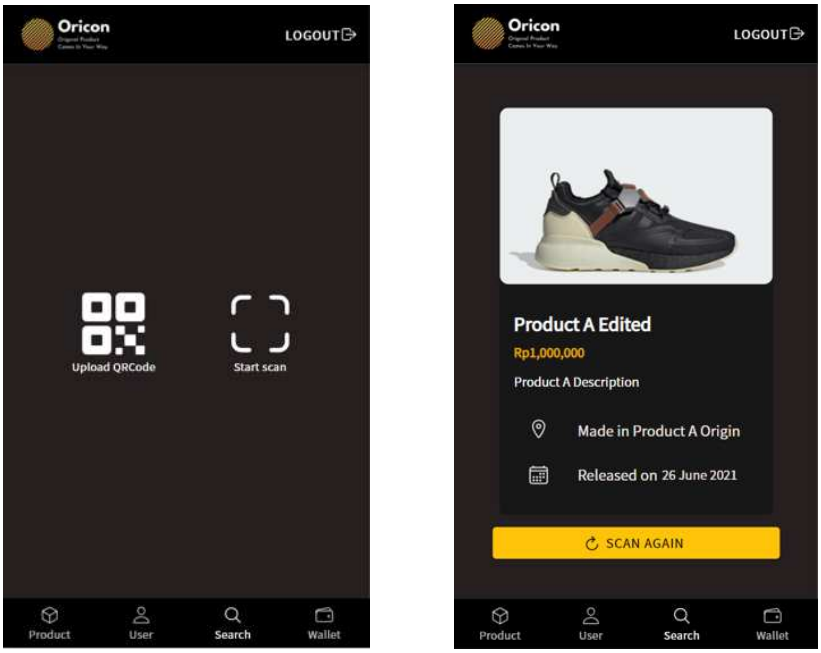

FIGURE 6. Search item

and mint tokens. This page is shown in Figure 8. The left picture is the Token Wallet Page when logged in as admin, while the right picture is the Token Wallet Page when logged in as token admin.

The final implementation expands related works mentioned earlier in Section 1 on anti-counterfeiting. The QR code is preferred due to its low-cost characteristics compared to radio frequency identification (RFID) and NFC. The idea is for the company or manufacturer to print the QR code on the physical goods directly, making it more challenging to replicate and modify the QR code. Along with the mobile application for smartphones, all smartphones today carry cameras and naturally serve as the tool to read the QR code.

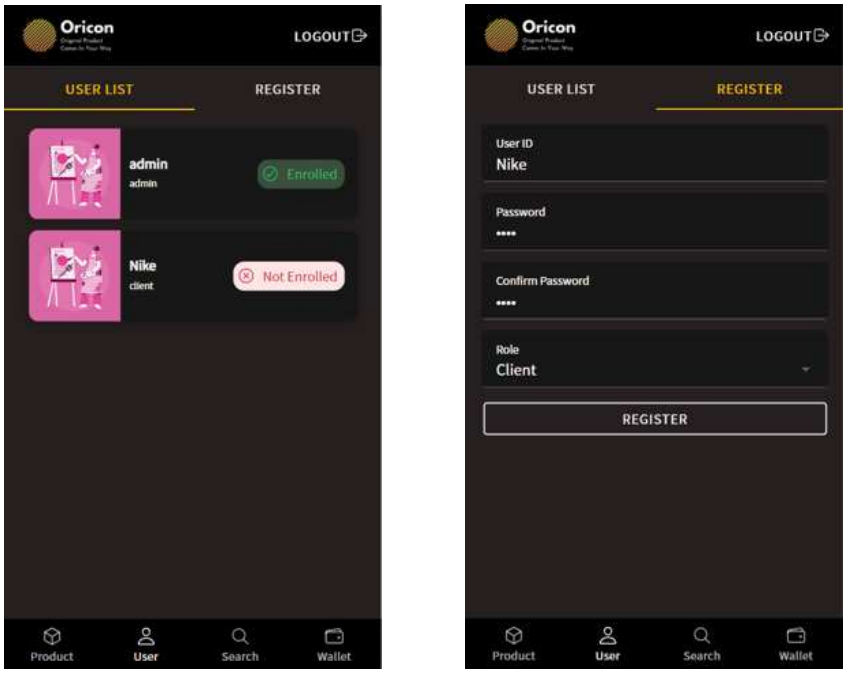

FIGURE 7. User profile

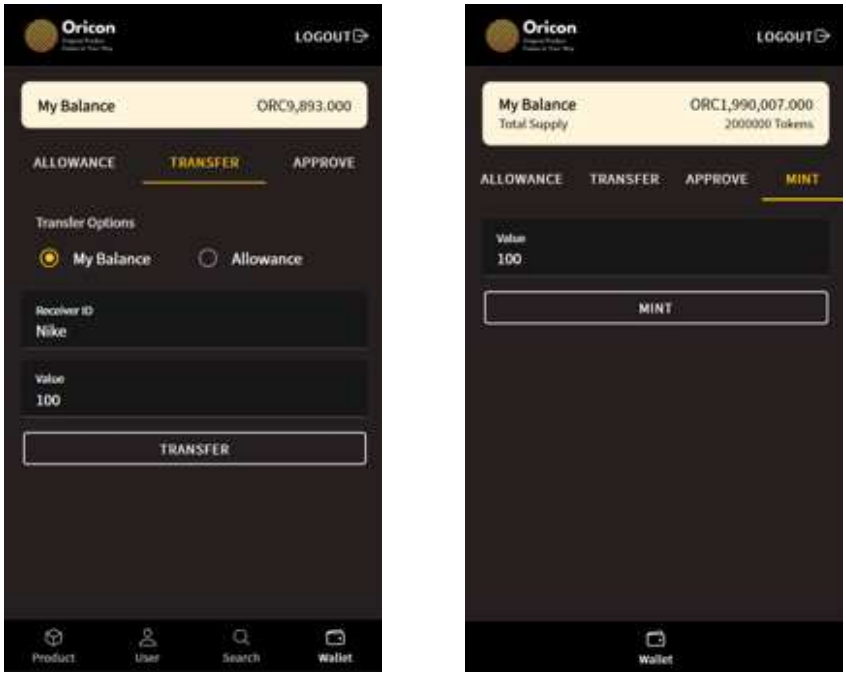

FIGURE 8. User wallet

Thus, no additional hardware devices are required by the users in the process (neither companies nor customers).

The semi-centralized architecture also benefits Oricon's users by maximizing performance and scalability while maintaining control of crucial aspects of the system, such as the central server and blockchain network containing important data from malicious attacks. The central server and blockchain network are owned and run entirely by Oricon, and they are not part of the companies. This ensures transparency and reliability in the integrity and honesty of the stored digital assets and related information from companies

and customers. Thus, the users (companies and customers) are only required to own a smartphone to run Oricon's client application for the proposed anti-counterfeiting solution. This whole design and architecture allow lightweight operation on the client side, which is suitable for field actors in this anti-counterfeiting activity.

The introduction of the ORC token allows Oricon to dynamically set the market price for the token, following the services' supply and demand. This is an advantage compared to anti-counterfeiting works using public blockchains such as Ethereum, which suffers from high transaction fees. Companies and customers willing to use the service could purchase ORCs directly from Oricon at a given price.

3.4. **Testing.** Performance evaluation is carried out to measure throughput and latency of Hyperledger Fabric implemented using Hyperledger Caliper v0.3.2. The testing is done to all transaction functions defined in smart contracts by dividing their execution into different test rounds. For each test round, except for deleting physical goods, txDuration is set to 30s to submit transactions continuously. For deleting physical good test round, txNumber is used for 100 transactions to submit transactions in total with a fixed amount. This is done to avoid deleting unknown digital assets which have never been created in the first place.

Several dummy physical goods are initialized with a size of approximately 26KB. The rate controller used is a fixed backlog, where the peak throughput per second (TPS) will be found by modifying the TPS to maintain the specified transaction backlog. The backlog is targeted at two transactions for each worker, and workers represent client nodes that trigger transactions. There are 20 dummy user accounts made to be used randomly.

There are six tests performed by configuring the number of BatchTimeout and MaxMessageCount, which is shown in Table 5. The numbers of AbsoluteMaxBytes and PreferredBytesSize used are 99MB and 512KB. For each test, 2 and 4 parallel transactions are used to observe the scalability by configuring the number of workers used in Hyperledger Caliper. In Tests 1 to 3, the number of MaxMessageCount is set to 20 to make it difficult for a block to contain 20 transactions before BatchTimeout is expired. BatchTimeout is tested at 2, 4, and 8 seconds. In Tests 4-6, the number of BatchTimeout is set to 20 to make it difficult for a block to wait for 20 seconds before MaxMessageCount is reached. MaxMessageCount is tested at 2, 4, and 8 transactions.

Table 5. Test parameter

Test No.	Parameter	BatchTimeout	MaxMessageCount
1	BatchTimeout	2	20
2	BatchTimeout	4	20
3	BatchTimeout	8	20
4	MaxMessageCount	20	2
5	MaxMessageCount	20	4
6	MaxMessageCount	20	8

4. **Results and Discussion.** The number of successful transactions, failed transactions, latency, and throughput of all transaction functions in smart contract are obtained and averaged in each test. The number of successful transactions with 4 workers is higher than 2 workers. The highest number of successful transactions occurs when the number of BatchTimeout is 2s. The number of failed transactions on 4 workers is also higher than 2 workers, when BatchTimeout is 4s and 8s. The number of failed transactions on

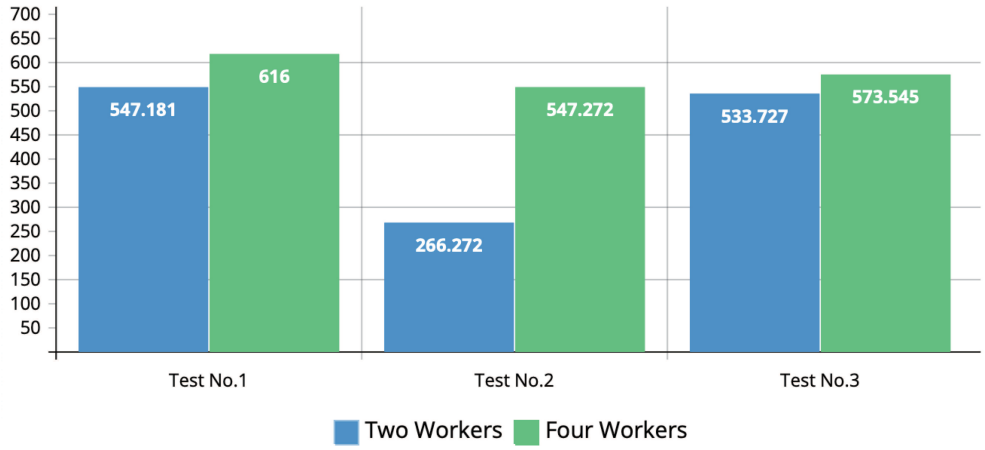

FIGURE 9. Average number of successful transactions for each BatchTimeout

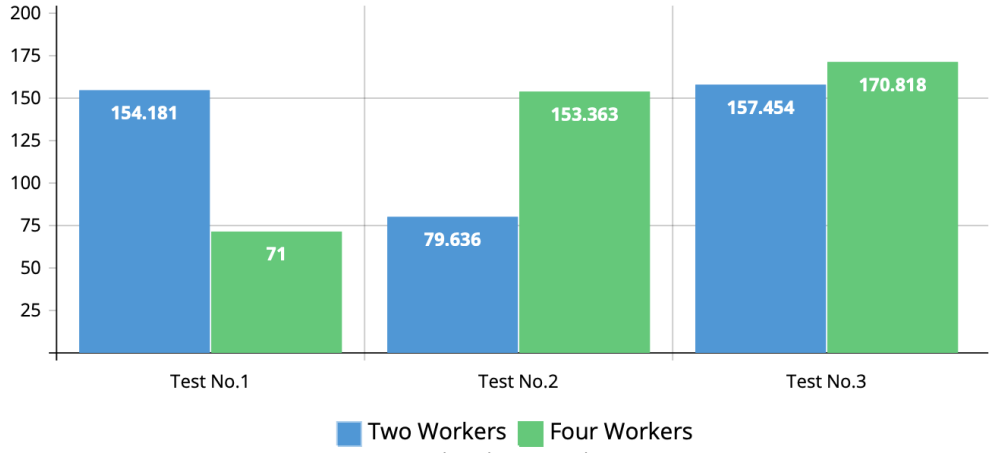

FIGURE 10. Average number of failed transactions for each BatchTimeout

4 workers tends to increase as the BatchTimeout gets longer. The graphics are shown in Figure 9 and Figure 10.

The number of latencies increases as the BatchTimeout gets longer on both 2 and 4 workers. The latency is not significantly affected by the number of workers, which can be seen that the values between two workers are not much different. The longer BatchTimeout increases the waiting time for a block to be committed. The throughput is not significantly affected when the BatchTimeout gets longer on both 2 and 4 workers. Besides, throughput is significantly affected by the number of successful transactions, which can be seen that both values have the same pattern. The graphics are shown in Figures 11 and 12.

The number of successful transactions with 4 workers is higher than 2 workers. The highest number of successful transactions occurs when the numbers of MaxMessageCount are 4 and 8 transactions. The number of failed transactions on 4 workers is also higher than 2 workers. The number of failed transactions on 4 workers tends to increase when the numbers of MaxMessageCount are 4 and 8 transactions. The number of successful and failed transactions decreases from 2 to 4 workers when the number of MaxMessageCount is 2. The graphics are shown in Figure 13 and Figure 14.

The number of latencies tends to increase as the MaxMessageCount increases on both 2 and 4 workers. The higher MaxMessageCount affects the number of transactions should

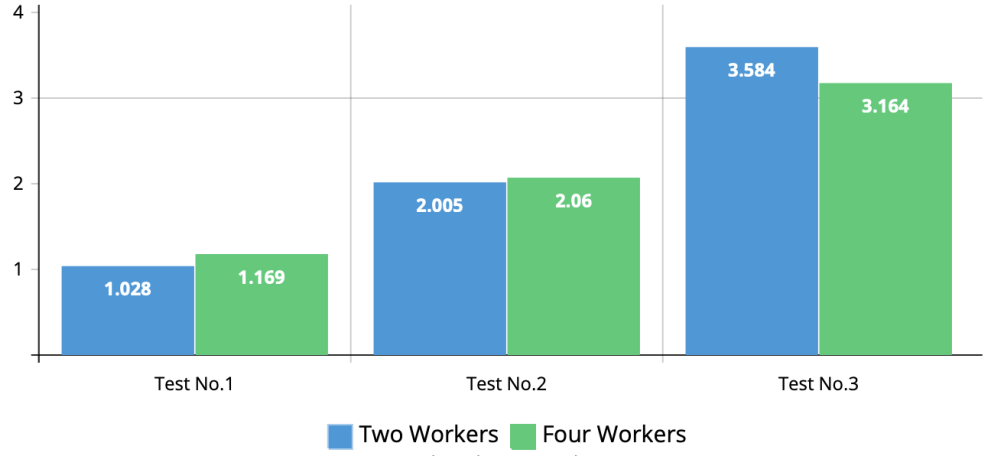

FIGURE 11. Average latency (s) for each BatchTimeout

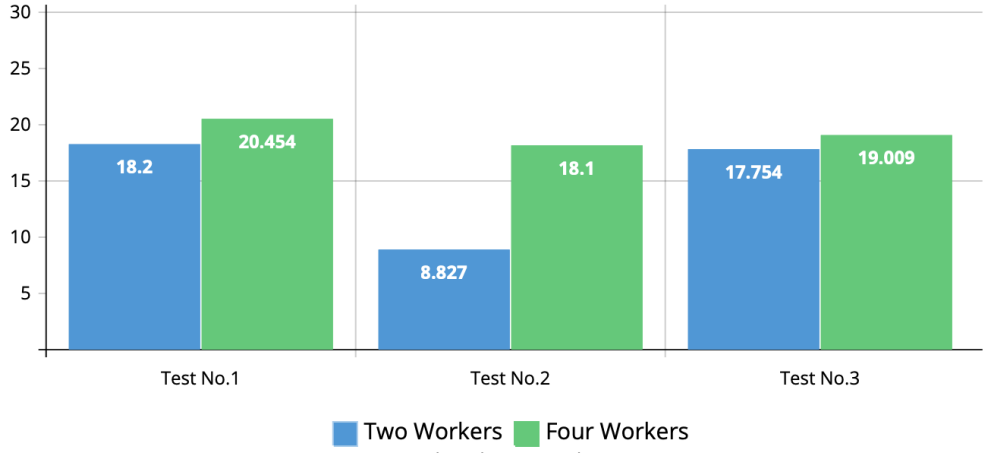

FIGURE 12. Average throughput (transactions/second) for each BatchTimeout

be reached in a block. The number of throughput tends to increase on both 2 and 4 workers when the numbers of MaxMessageCount are 4 and 8 transactions. The number of throughput tends to decrease from 2 to 4 workers when the number of MaxMessageCount is 2 transactions. This result is an anomaly, where the number of BatchTimeout is 2 seconds, transactions are commited more often. This causes read-write conflicts during the validation phase. The number of throughput increases if many successful transactions are committed in a batch. The graphics are shown in Figure 15 and Figure 16.

Overall, BatchTimeout and MaxMessageCount affect latency and throughput. The larger the BatchTimeout, the higher the latency. The higher MaxMessageCount requires many parallel transactions to maintain low latency. The higher latency can lower the throughput, but not so significantly. The number of parallel transactions does not affect the latency significantly.

Throughput is affected significantly by the number of committed transactions that are valid. The larger MaxMessageCount can increase the throughput if many transactions in one batch do not conflict. Submitted transactions are prone to conflict if they are trying to update to the same object. The higher number of parallel transactions causes the MaxMessageCount to be reached quickly.

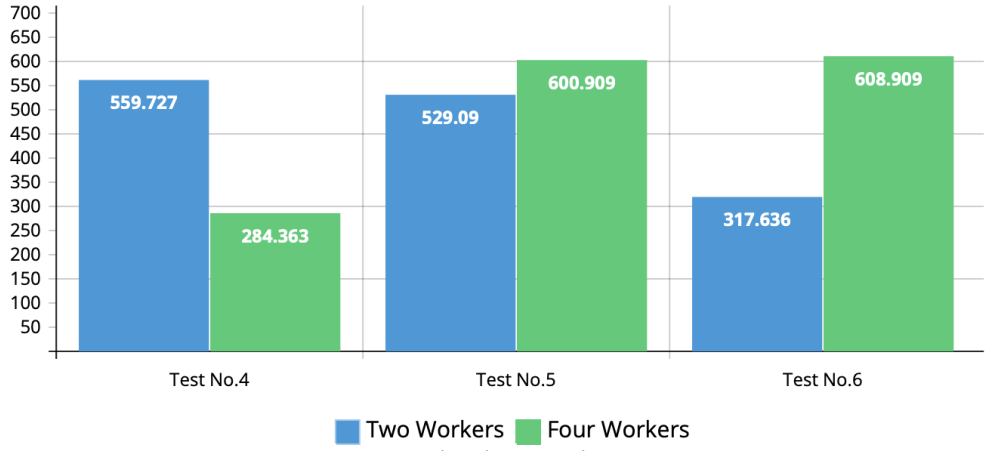

FIGURE 13. Average number of successful transactions for each MaxMessageCount

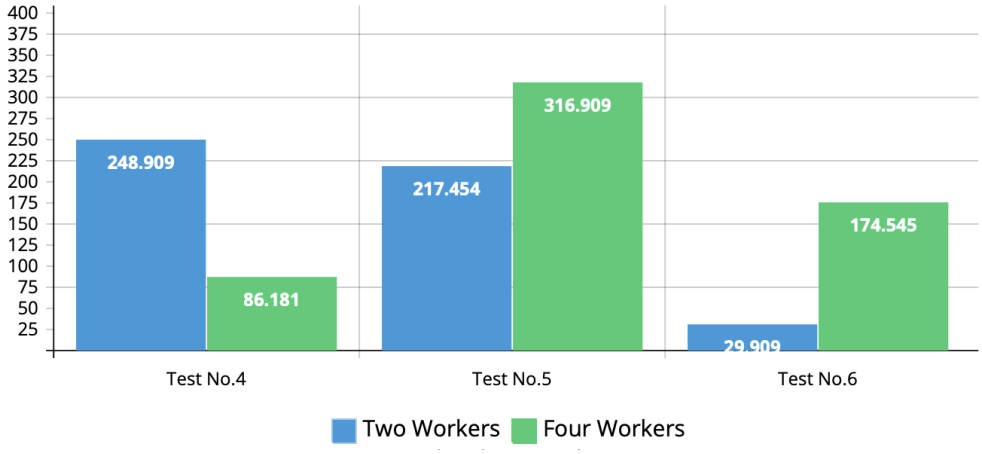

FIGURE 14. Average number of failed transactions for each MaxMessageCount

The urgency of Oricon's transactions depends on how often parallel transactions are submitted. The more parallel transactions submitted, the higher the urgency of the transaction to be committed quickly. Therefore, the BatchTimeout and MaxMessageCount are configured to 2s and 20 transactions, respectively, as default parameters. The performances obtained are latency of 1.028s and throughput of 18.2 TPS for every two parallel transactions, and latency of 1.169s and throughput of 20.454 TPS for every four parallel transactions.

5. **Conclusions.** The use of Hyperledger Fabric for applications tailored to this work is completed successfully. The Oricon can be used for companies to create, store, and manage the digital assets of the physical goods and for their customers to identify and verify the authenticity and origin of the physical goods. In order to execute the smart contract and submit transactions, Oricon consumed ORC tokens as fungible tokens, which were compliant with the ERC-20 token standard. ERC-20 token standard has the advantage of being more reliable for interoperability. However, this token standard limits the scalability of the network. When parallel transactions were submitted, a few transactions failed to be committed because their read-write set values were conflicted.

Performance evaluation based on latency and throughput has also been obtained. The default parameter is set to 2s of BatchTimeout and 20 transactions of MaxMessageCount.

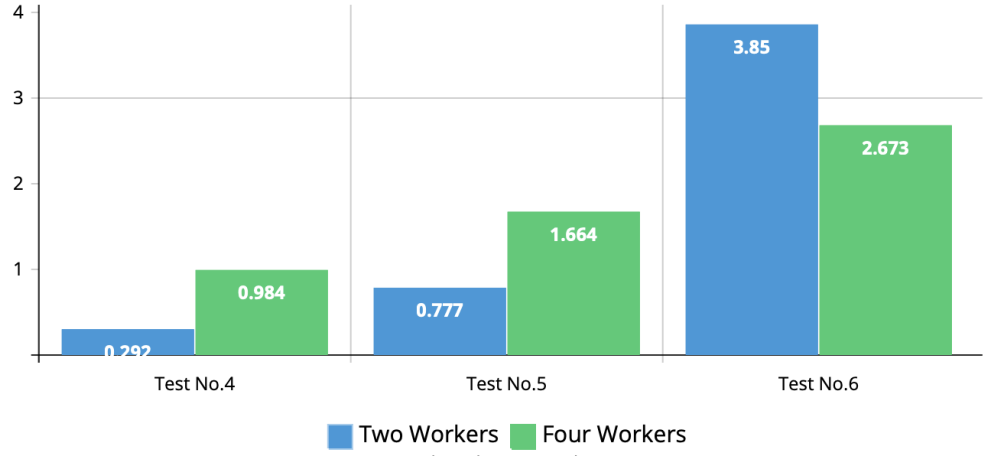

FIGURE 15. Average latency (s) for each MaxMessageCount

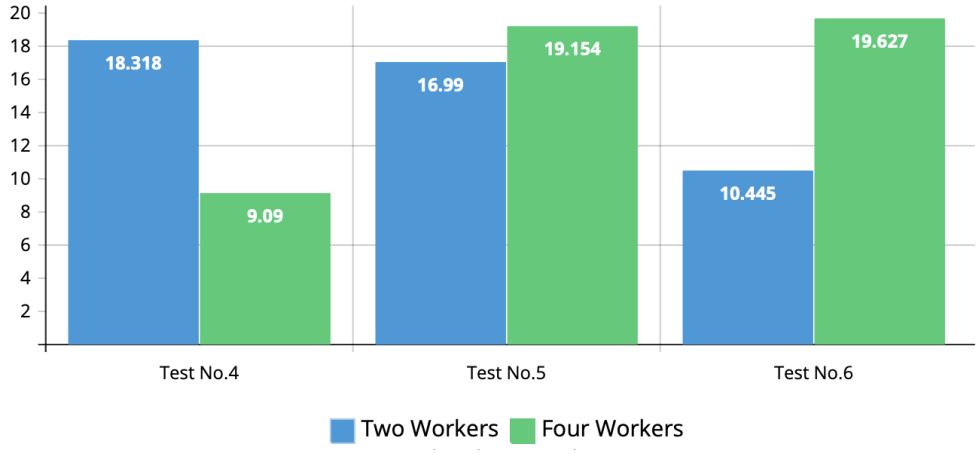

FIGURE 16. Average throughput (transactions/second) for each MaxMessageCount

This parameter is chosen because the token transactions need to be committed quickly yet can contain as many transactions as possible. The performances obtained are latency of 1.028s and throughput of 18.2 TPS for every two parallel transactions, and latency of 1.169s and throughput of 20.454 TPS for four parallel transactions. There are also other findings related to performance evaluation that can be concluded as follows.

- Latency is affected by the numbers of BatchTimeout and MaxMessageCount. The higher the BatchTimeout and MaxMessageCount, the higher the latency obtained. BatchTimeout affects the latency because the waiting time for a block committed in each batch becomes longer. Meanwhile, MaxMessageCount affects the latency because more transactions must be submitted in each batch.
- Throughput is affected by the number of parallel transactions and latencies. The higher latency can lower the throughput, but the effect is insignificant if the number of parallel transactions and MaxMessageCount is high. The higher MaxMessage-Count allows many parallel transactions to be committed in each batch. In order to increase the throughput, the number of committed transactions must be valid.
- BatchTimeout and MaxMessageCount can be configured based on the number of parallel transactions and dependency among transactions. Suppose the number of parallel transactions is high, and each transaction does not tend to update the same object. In that case, the number of BatchTimeout and MaxMessageCount can be

increased to maintain high throughput values. Suppose the number of parallel transactions is high, and each transaction tends to update the same object. In that case, the number of BatchTimeout and MaxMessageCount can be lowered to avoid conflicts during the validation phase.

Acknowledgment. The authors thank Universitas Multimedia Nusantara for the support of this research work.

REFERENCES

[1] D. Weinberger, How blockchain, smart tags are tackling counterfeit goods, *Supplychainbrain*, https://www.supplychainbrain.com/blogs/1-think-tank/post/34693-tackling-product-counterfeiting-with-block chain, Accessed on Jul. 04, 2022.
[2] International Chamber of Commerce, *Global Impacts of Counterfeiting and Piracy to Reach US$4.2 Trillion by 2022*, 2017.
[3] B. Berman, *Damage to the Global Economy from Counterfeit Goods Exceeds $300 Billion; Risk of Injury, too*, Ipcloseup, 2021.
[4] OECD and EUIPO, *Trends in Trade in Counterfeit and Pirated Goods*, 2019.
[5] U.S. Customs and Border Protection, *Counterfeit Shipment with 550 Pounds of Fake Apple, Samsung Products Seized by Cincinnati CBP*, 2020.
[6] B. B. A. Christyono, M. Widjaja and A. Wicaksana, Go-Ethereum for electronic voting system using clique as proof-of-authority, *TELKOMNIKA (Telecommunication Comput. Electron. Control)*, vol.19, no.5, p.1565, 2021.
[7] L. Mark, V. Ponnusamy, A. Wicaksana, B. B. Christyono and M. Widjaja, A secured online voting system by using blockchain as the medium, in *The Smart Cyber Ecosystem for Sustainable Development*, P. Kumar, V. Jain and V. Ponnusamy (eds.), Wiley, 2021.
[8] M. El Khatib, F. Beshwari, M. Beshwari and A. Beshwari, The impact of blockchain on project management, *ICIC Express Letters*, vol.15, no.5, pp.467-474, 2021.
[9] I. Singhal, Anti-counterfeit product system using blockchain technology, *Int. J. Res. Appl. Sci. Eng. Technol.*, vol.9, no.12, pp.291-295, 2021.
[10] P. Danese, R. Mocellin and P. Romano, Designing blockchain systems to prevent counterfeiting in wine supply chains: A multiple-case study, *Int. J. Oper. Prod. Manag.*, vol.41, no.13, pp.1-33, 2021.
[11] N. Alzahrani and N. Bulusu, A new product anti-counterfeiting blockchain using a truly decentralized dynamic consensus protocol, *Concurr. Comput. Pract. Exp.*, vol.32, no.12, 2020.
[12] N. C. K. Yiu, An empirical analysis of implementing enterprise blockchain protocols in supply chain anti-counterfeiting and traceability, *arXiv.org*, arXiv: 2102.02601, 2021.
[13] N. C. K. Yiu, Toward Blockchain-enabled supply chain anti-counterfeiting and traceability, *arXiv.org*, arXiv: 2102.00459, 2021.
[14] N. C. K. Yiu, An NFC-enabled anti-counterfeiting system for wine industry, *arXiv.org*, arXiv: 1601.06372, 2021.
[15] Editorial Team, Ethereum presents London hard fork to fix high transaction fees on its network, *VOI*, https://voi.id/en/technology/67336/ethereum-presents-london-hard-fork-to-fix-high-transaction-fees-on-its-network, Accessed on Jul. 04, 2022.
[16] Hyperledger Fabric Documentation, *Introduction*, 2020.
[17] P. Cluchet, M. Koscina and M. Lombard-Platet, PlasticCoin: An ERC20 implementation on hyperledger fabric for circular economy and plastic reuse, *Proc. of 2019 IEEE/WIC/ACM Int. Conf. Web Intell.*, no.10, pp.223-230, DOI: 10.1145/3358695.3361107, 2019.
[18] E. Androulaki et al., Hyperledger fabric: A distributed operating system for permissioned blockchains, *Proc. of the 13th EuroSys Conf. (EuroSys2018)*, vol.2018-Janua, no.1, DOI: 10.1145/3190508.3190538, 2018.
[19] D. Ongaro and J. Ousterhout, In search of an understandable consensus algorithm, *Proc. of 2014 USENIX Annu. Tech. Conf. (USENIX ATC 2014)*, pp.305-319, 2019.
[20] Hyperledger Performance and Scale Working Group, Hyperledger Blockchain performance metrics, *Hyperledger.org*, pp.1-17, 2018.
[21] Hyperledger Caliper Documentation, *Getting Started*, 2020.

Author Biography

Wilson Philips received the B.Sc. degree in Informatics from Universitas Multimedia Nusantara, Indonesia, in 2021. His research interests are blockchain applications and applied computing.

Arya Wicaksana is a lecturer at the Department of Informatics at UMN. He received Master Degree in VLSI Engineering from Universitas Tunku Abdul Rahman. He successfully demonstrated the UTAR first-time success ASIC design methodology on a multi-processor system-on-chip project using 0.18 μm processing technology in 2015. His main research interests are blockchain applications and computational intelligence. He recently worked on decentralized autonomous social media. He is affiliated with ACM and IEEE as a professional member. He has served as an invited reviewer in IEEE ACCESS, IJNMT, and IFERP and an invited author in IntechOpen and other scientific publications.