Available online at www.centmapress.org

INTERNATIONAL JOURNAL ON FOOD SYSTEM DYNAMICS

Int. J. Food System Dynamics 13 (3), 2022, 321-334

DOI: https://dx.doi.org/10.18461/ijfsd.v13i3.C6

Halal Blockchain Application for a Chicken Slaughtering Factory

Adam Akbar[a], Nur Aini Rakhmawati[a], and Iwan Vanany[b]

[a]*Department of Information Systems, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia.*

[b]*Department of Industrial and Systems Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia.*

adamakbar.id@gmail.com; nur.aini@is.its.ac.id; vanany@ie.its.ac.id

Received June 2021, accepted February 2022, available online June 2022

ABSTRACT

The vast majority of traded products in Indonesia must be halal certified. This means that Indonesian consumers rely heavily on certificates issued by the Indonesian halal authority. Halal certification validates individual products and after four years, a certificate expires and must be renewed. One of the requirements in the certification process is that all the ingredients of the product must be halal. For this reason, we attempted to enhance the information on halal products, production, processes and delivery processes, using blockchain to address the validity issue. This paper proposes a blockchain-based system to support traceability in the chicken slaughterhouse industry. The design was tested using a black-box approach with 46 test cases that were based on a general scenario in a chicken slaughterhouse factory. The test result showed that all test cases produced the desired output. Hence, the proposed system fulfills the goal of enhancing the traceability of halal products. Ultimately, the buyers of halal chicken products will be assured that the chicken slaughterhouse followed the halal assurance system in its halal-critical processes.

Keywords: *halal food; supply chain; blockchain; trace system.*

1 Introduction

Every Muslim is bound to the Islamic law that defines what they are allowed to do and not allowed to do, commonly known as *halal* and *haram*, respectively (Jallad, 2008). 'Halal' in Arabic means permissible, and the opposite is 'haram', which means forbidden (Oxford University Press, 2009). The halal and haram status are also applied to food. Food is considered halal if it satisfies a list of requirements, such as being toxin-free, safe to consume, not containing any haram substances, and many more (Khattak et al., 2011). Any food products derived from carnivores are considered haram (Latif et al., 2014). Any animal products that were processed not according to Islamic law are haram, e.g., slaughtered without reciting the name of Allah, or processed in the same place as haram meat products (Yani et al., 2020). The halal status of food needs to be ensured not only in terms of the production process but also in terms of the raw materials from suppliers, delivery, storage, and other processes before it reaches the end consumer. The assurance of halal food along the supply chain using an information system and digital technology requires a blockchain application (Tan et al., 2017).

As the largest Muslim country in the world, Indonesia has made the standard for halal food into a national regulation (Nurjaya et al., 2021), which means that all products that are traded in Indonesia must follow halal regulations and be halal certified (Presiden Republik Indonesia, 2021). The Institute of Food, Drugs and Cosmetics Study of the Indonesian Ulama Council (LPPOM MUI) as one of the official halal auditing agencies, audits the halal status of foods, drugs, and cosmetics (Ratanamaneichat and Rakkarn, 2013). Halal certificates are issued by BPJPH (*Badan Penanggulangan Jaminan Produk Halal* – Halal Product Assurance Agency), as the halal authority in Indonesia. Vanany et al. (2020) point out that a blockchain application is not only needed between halal authorities and halal food companies but also by halal food companies to strengthen their halal business system.

Blockchain technology has been implemented in many sectors. The most well-known application is the Bitcoin cryptocurrency in the financial sector (Chatterjee and Chatterjee, 2017). Blockchain technology is also used for electoral voting systems in the government sector, because of its high availability, verifiability, integrity, and ease of determination (Hanifatunnisa and Rahardjo, 2017). Zhang et al. (2017) have implemented blockchain to deliver patient data securely to various organizations and devices in the healthcare sector. They concluded that blockchain technology allows a more interoperable environment, which cannot easily be achieved using traditional approaches. Meanwhile, in the supply chain field, Jamil et al. (2019) used blockchain to prevent the spread of counterfeit drugs by implementing a secure drug supply chain record system. The scope of blockchain application has also been expanded to the food industry (Bumblauskas et al., 2020), including in the halal context (Queiroz et al., 2019).

Blockchain applications in the food industry enhance product traceability, authenticity, and legality among supply chain actors (Wang et al., 2019) through an open ledger to record all supply chain transactions (Hew et al., 2020). Generally, the objectives of blockchain application are enhancing traceability (Tan et al., 2020) and increasing halal supply chain performance (Surjandari et al., 2021). The existence of transparency in a halal traceability system is believed to enhance halal integrity for end consumers, because it allows them to access supply chain information from the first process until it reaches them (Hew et al., 2020). Currently, Halal Digital Chain in Malaysia, HalalChain in UAE (Hew et al., 2020), Korean Telecom, Arab-Brazilian Chamber of Commerce, and PT Sierad Produce Tbk (Vanany et al., 2021b) have adopted blockchain technology for a halal traceability system to support halal food integrity.

Concerning the halal chicken industry in Indonesia, Wahyuni et al. (2018) surveyed two Indonesian companies to measure the safety and halal risk in the product manufacturing process of both companies using the Risk Exposure concept. They found that the highest risk in food safety was in the delivery and cooking process. In the first company, the halal risk was the highest in the slaughter process. In the second company, it was in the manufacturing process. In another study, Vanany et al. (2019) developed a multi-phased Quality Function Deployment (QFD) model based on HAS 23000 requirements to identify the key processes and their priority for improvement programs of manufacturing halal chicken meat. The multi-phased QFD model was applied in an Indonesian company processing approximately 8000 chickens per day. Based on the case study, the multi-phased QFD model identified that the slaughtering and delivery processes were key. The equipment, the procedures, and the workers involved in the processes were determined as the most critical halal factors. Improvement via training and education was suggested. Vanany et al. (2020) proposed a conceptual framework for Indonesian halal food integrity based on the case of Indonesian halal chicken meat. The study resulted in a blueprint for a blockchain system to achieve integrity of halal food, including transparency and real-time tracking. Therefore, we practically developed the design from Vanany et al. (2020).

As an upstream part of the supply chain, slaughterhouses have an impact on the halal status of the end product. It is helpful to have all records of slaughterhouse activities, for its customers and for LPPOM MUI. To ensure that all records that have been stored are valid, blockchain technology can be used as the core system. Several previous studies closely related to the Indonesian halal food situation were used as a reference in constructing the ideal scenario in this study, as a requirement to be accommodated by the halal blockchain system (Wahyuni et al., 2018; Vanany et al., 2019; 2021a).

The main objective of this study was to practically develop and test the halal food blockchain-based tracing system design from Vanany et al. (2020) study.

The design of this system is proposed to extend the advantages of blockchain for the halal food supply chain based on the case of a chicken slaughtering factory in Indonesia. The proposed system design can be considered to make the halal certification process in Indonesia more efficient for all parties involved. This paper is structured as follows: 'Proposed System Design' explains the proposed system design, 'Experimental Result' explains the experiment and the result, and Conclusion' concludes the paper.

2 Proposed System Design

The proposed system was designed in five steps. First, we defined a scenario based on several previous studies (Wahyuni et al., 2018; Vanany et al., 2019; 2021a). Based on the scenario, we defined the roles involved who produce or consume information that is in the system. Second, we modeled the information to be stored or retrieved to fit with the system development. In the third step, we designed the system's architecture to support the system's availability. Fourth, we mapped the actions of each role to the system. In the last step, we defined a way for users to access the system. These steps will be explained in the following subsection.

2.1 Scenario

There are eight roles involved according to Vanany et al. (2020): 1) buyers, 2) suppliers, 3) logistics, 4) halal auditing agency, 5) halal government authority, 6) small micro business, 7) halal supervisor, and 8) MUI. In our case, we identified only two roles who produce primary data for the system and one more role that supports these two roles. Hence, the scope of the proposed system is limited to these three roles. The first role is the supplier, who produces poultry meat-based products like raw meat, sausages, or nuggets. The second role is the customer (buyer), who purchases the products of the supplier. A company may have overlapping roles, which means that the supplier can be a customer of another supplier. The last role is the courier (logistics), who delivers products from suppliers to customers. Figure 1 illustrates the positions of the three roles in the proposed system. The supplier can manage product details and item details. The courier can update shipment details like the location of items that they deliver to customers. The customer can get details of all items that they buy. The term 'product' is used for the thing that a supplier produces and offers to customers. The term 'item' refers to a physical product that has been manufactured by a supplier and is delivered to the customer.

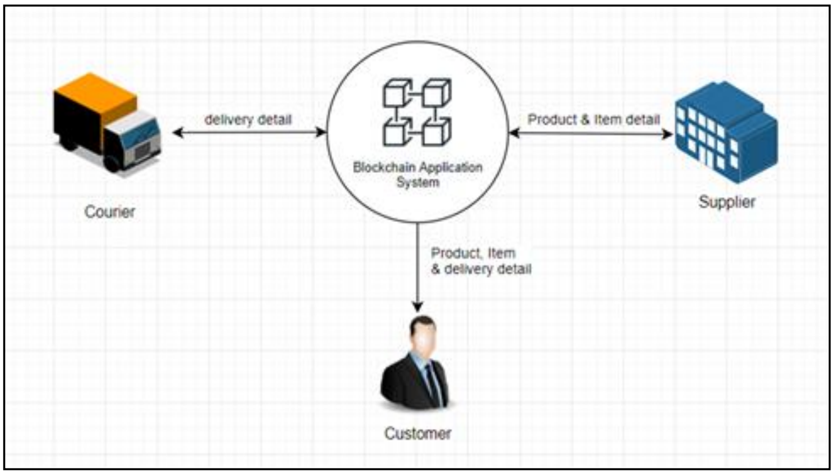

Figure 1. System Illustration.

The five main processes in the proposed system are shown in Figure 2, namely: 1) managing product details; 2) managing manufacturing details; 3) issuing invoices; 4) managing shipment; and 5) tracking and tracing items. These processes are an expansion of the process overview in Figure 1. The first three processes are actions from the supplier's end, the fourth process is an action from the courier's end, and the fifth process is an action from the customer's end. In an ideal case, the supplier manages all product details before the customer places an order. The customer places an order based on the product list that they have received from the system. After that, the supplier should fulfill the order by manufacturing the items if the ordered items have not been manufactured yet. While manufacturing the items, the supplier must store information about the manufacturing activities in the system. After the ordered items are ready, the supplier can issue an invoice containing all ordered items and store that information in the system. After the invoice has been issued, the courier can pick up the ordered items and deliver them to the customer. While delivering the items, the courier must submit their location to the system when reaching any checkpoints in the real world.

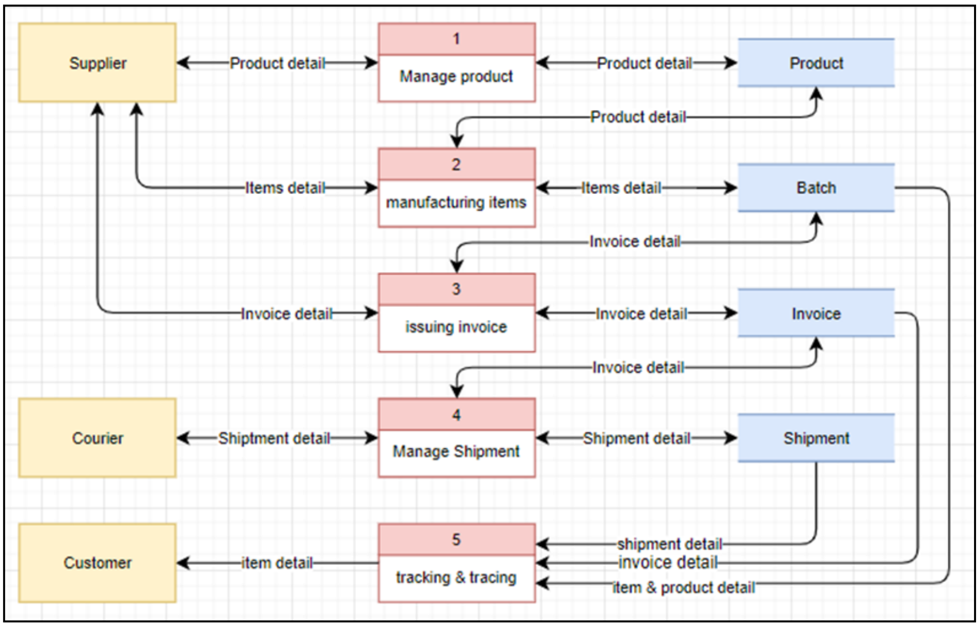

Figure 2. Data Flow Diagram of Proposed System.

In our scenario, there are seven parties: three suppliers *(supp1*, *supp2*, and *supp3*), three customers *(cust1*, *cust2*, and *cust3*), and one courier. *Supp1* is a supplier for *cust1*; *supp2* supplier for *cust2*; and *supp3* is a supplier for *cust3*. The courier handles all shipments from these companies. To give a broader context, *supp1* and *supp2* are slaughterhouses that produce raw chicken meat. *Supp3* is a frozen food company that produces sausages and minced chicken. *Cust1* is a fine dining restaurant that serves various processed chicken foods. *Cust2* is a fast-food franchise restaurant that makes fried chicken. *Cust3* is a company that has a dozen food trucks to sell hotdogs and hamburgers. An overview of this scenario is shown in Figure 3, along with the scope of the system. The scope is limited to seven companies. The chicken farm and products produced by the customers are not recorded in the system.

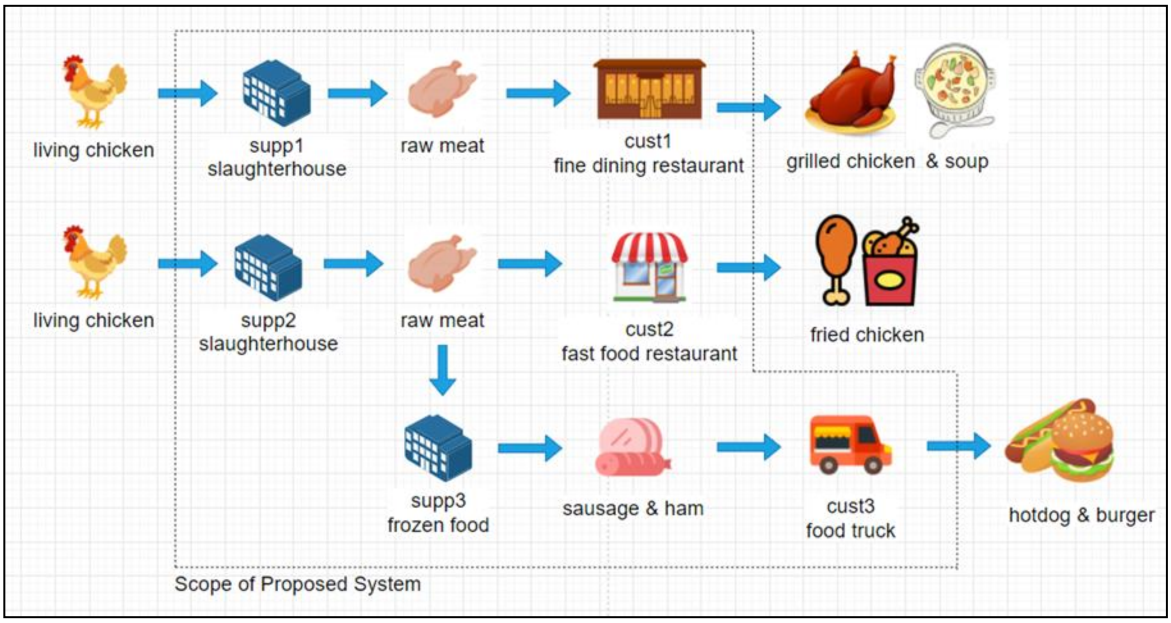

Figure 3**.** Overview of the Main Scenario.

2.2 Data Model

As shown in Figure 2, the proposed system models the information into four tables to fulfil the traceability needs: 1) product; 2) batch; 3) invoice; and 4) shipment. All product details are stored in the product table; item details are stored in the batch table; purchased item details from the customer are stored in the invoice table, and shipment details are stored in the shipment table. Based on the scenario from the previous subsection, one product can be an ingredient of another product, so all ingredients of the product must be listed in the system. We also need an identification number (ID) for every entity in the system. For product details, we need to store the product's name, the halal certificate of the product, and the product manufacturer's name. Hence, the properties listed in the product table are: 1) *ID*; 2) *name*; 3) *ingredients*; 4) *halalCertificateID*; and 5) *manufacturerName*. As item details, we store the production code as the item's ID, the product base of that batch, the quantity of the item in that batch, and the batch ID of the ingredients that were used in manufacturing that batch. The properties listed in the batch table are: 1) *ID*; 2) *product*; 3) *qty*; and 4) *ingredients*. As invoice details, we store the invoice number as the ID, the ID of the buyer, and the details of the purchased items. The properties listed in the invoice table are: 1) *ID*; 2) *buyerID*; and 3) *items*. As shipment details, we store the shipment code as the ID, the list of invoices related to that shipment with their details, and the shipment history. The properties listed in the shipment table are: 1) *ID*; 2) *invoices*; and 3) *history*. A summary of the data model is presented in **Fehler! Ungültiger Eigenverweis auf Textmarke.**.

Table 1.
Summary of the Data Model.

Table	Property	Description
Product	ID	Product identification number
	Name	Product name
	Ingredients	List of product ingredients
	halalCertificateID	The halal certificate number
	manufacturerName	The name of the manufacturer
Batch	ID	Batch identification number
	Product	The state of the product being manufactured in this batch
	Qty	Quantity of items produced in this batch
	Ingredients	List of ingredients in this batch
Invoice	ID	Invoice number
	buyerID	Organization identification number of the buyer
	Items	List of items purchased by the buyer
Shipment	ID	Shipment code
	Invoices	List of invoice ID with their items and quantity
	History	List of locations and dates

2.3 System Architecture

Before getting into the architecture of the proposed system, it is necessary to explain the fundamental technicalities of the blockchain. The commonly accepted definition of a blockchain is that it is a set of nodes that validate each other to increase system integrity since it has no central control. In Hyperledger Fabric, there are three types of nodes: 1) the client, who wants to submit the transaction; 2) the peer, who conducts the transaction; and 3) the orderer, who communicates with all the nodes through a broadcast system (Hyperledger, 2020a). Hyperledger Fabric implements a channel system, which is a subnetwork of a blockchain network (Hyperledger, 2018). This channel system makes the ledger accessible only to channel members. For example, in the scenario, we have *supp1* carrying out a transaction with *cust1*; and *supp2* carrying out a transaction with *cust2*. If we deliver them to the supplier-customer-only network, *cust1* cannot access the transaction data of *cust2*, and vice versa. Besides nodes and channels, Hyperledger Fabric uses PKI (Public Key Infrastructure) (Chokhani et al., 2003) to authorize every node in the blockchain network. The CA (Certificate Authority) node authorizes clients through the PKI method (Hyperledger, 2021).

In the proposed system, one channel is dedicated for three companies only – the supplier, the customer, and the courier – to keep their company confidential to other companies. An exception is made for *channel0*, which is dedicated to holding all product and batch details stored by all suppliers and accessed by all customers. *Channel0* does not have an invoice and shipment table because those tables are restricted to the companies involved in transactions. The invoice and shipment tables are available on another channel.

Every company has a peer, and every peer has a connection to other peers based on the scenario that was described earlier. In detail, the *channel0* members are: 1) *supp1*; 2) *supp2*; 3) *supp3*; 4) *cust1*; 5) *cust2*; 6) *cust3.* The *channel1* members are: 1) *supp1*; 2) *cust1*; and 3) *courier*. The *channel2* members are: 1) *supp2*; 2) *cust2*; and 3) *courier*. The *channel3* members are: 1) *supp2*; 2) *supp3*; and 3) *courier*. The *channel4* members are: 1) *supp3*; 2) *cust3*; and 3) *courier*. There is one orderer node for every channel to handle every dataset broadcast to each channel's members. Every peer has a CA node to authorize every user that enters the blockchain network. The REST API application is connected to the network through CA; this allows client applications to establish connections with the blockchain network via REST API. This architecture is illustrated in Figure 4 and a summary of the channel members is given in Table 2.

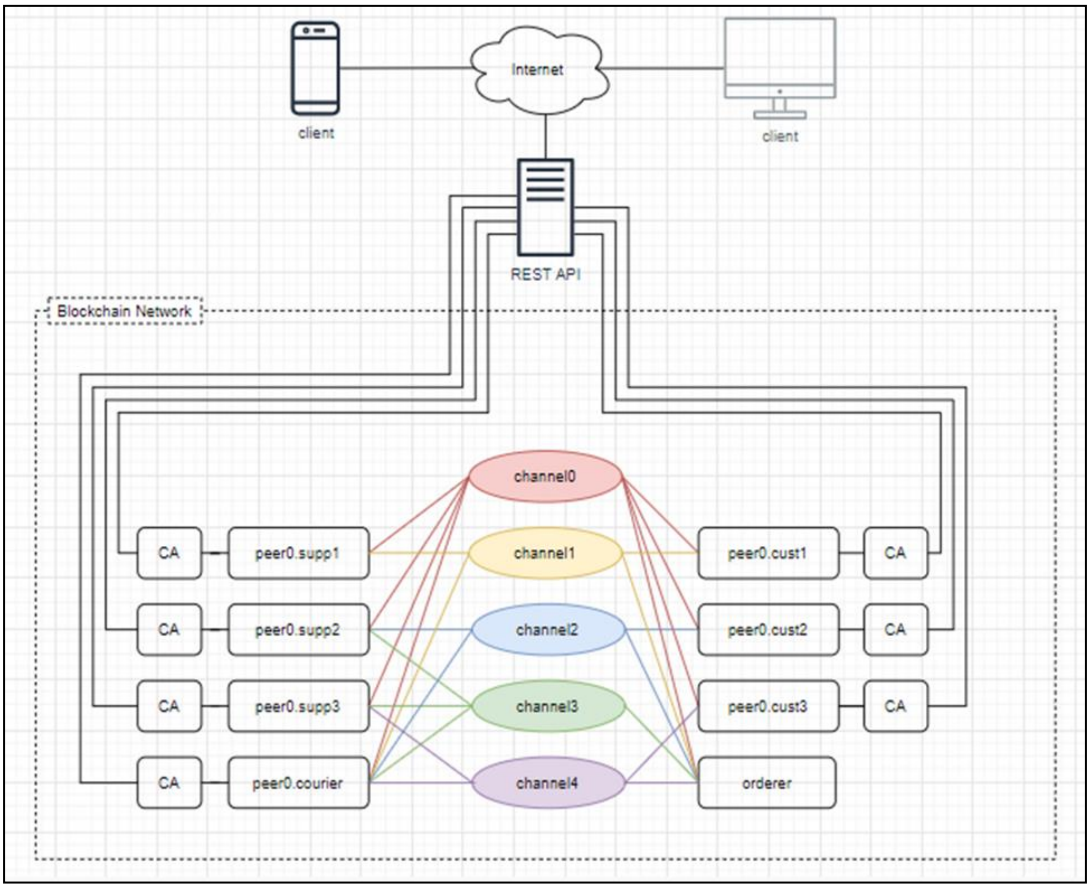

Figure 4. the Architecture of the Proposed System.

Table 2.
Summary of Channel Members.

Channel name	Channel members
Channel0	Supp1, supp2, supp3, cust1, cust2, cust3, and orderer
Channel1	Supp1, cust1, courier, and orderer
Channel2	Supp2, cust2, courier, and orderer
Channel3	Supp2, supp3, courier, and orderer
Channel4	Supp3, cust3, courier, and orderer

2.4 Smart Contract

In the real world, a contract is used for locking an agreement between parties, which includes what the parties can do or cannot do. A contract is a point of reference for each party to take action. Like a real-world contract, a smart contract is used in a blockchain to reference peers when taking action, especially peer action to the ledger. The consensus of all blockchain members applies to this contract. A smart contract is a computer program defining actions that peers can take. Since a blockchain is a decentralized system, there is no central control to enforce rules to all members. In Hyperledger Fabric 2, a smart contract can be written in several programming languages, such as GO, Java, or Typescript. In the proposed system, Typescript was used for developing a smart contract to get a richer development experience using its annotations and decorators (Hyperledger, 2020b).

Based on the five main processes that are described in the previous subsection, a smart contract must have all functions to fulfill the purpose of the processes. The *Managing product* process requires storing, retrieving, and manipulating product details in the ledger. The *Manufacturing item* process requires storing, retrieving, and manipulating batch details in the ledger. The *Issuing invoice* process includes storing, retrieving, and manipulating invoice details in the ledger.

The *Managing shipment* process consists of storing, retrieving, and manipulating shipment details in the ledger. The last process, the *Tracking and tracing* process, requires retrieving and manipulating all details that have been stored in the system.

We named all required functions using CRUD and Camel Case naming conventions to make the terminology more understandable for other developers. 'Create' for storing, 'read' for retrieving, and 'update' for manipulating and the table names follow suit. The result of function naming and a summary of all actions are presented in Table 3. By removing redundant actions, twelve functions for the smart contract are gathered. The list of all functions is: 1) *createProduct*; 2) *readProduct*; 3) *updateProduct*; 4) *createBatch*; 5) *readBatch*; 6) *updateBatch*; 7) *createInvoice*; 8) *readInvoice*; 9) *updateInvoice*; 10) *createShipment*; 11) *readShipment*; and 12) *updateShipment*.

Table 3.

Summary of Smart Contract Functions.

PROCESS	ACTION	FUNCTION NAME
Managing product	Store product details	createProduct
	Retrieve product details	readProduct
	Manipulate product details	updateProduct
Manufacturing item	Store shipment details	createBatch
	Retrieve shipment details	readBatch
	Manipulate shipment details	updateBatch
Issuing invoice	Store invoice details	createInvoice
	Retrieve invoice details	readInvoice
	Manipulate invoice details	updateInvoice
Managing shipment	Store shipment details	createShipment
	Retrieve shipment details	readShipment
	Manipulate shipment details	updateShipment
Tracking and tracing	Retrieve product details	readProduct
	Retrieve batch details	readBatch
	Retrieve invoice details	readInvoice
	Retrieve shipment details	readShipment

2.5 REST API

REST API is used as a tunnel from the client application to execute the smart contract function through a URL to communicate with the blockchain network. Since the application development trend has shifted to microservices, modern web or mobile applications can quickly implement this communication approach. We use Express.js (OpenJS Foundation, n.d.) as a REST API server with JSON format for data exchanges between clients and servers in the proposed system. The URL format must include the channel name, peer name, and function name separated by a slash ("/") to identify which channel, which peer, and what function to execute. Based on RFC 7231 about the hypertext transfer protocol, the HTTP methods for storing, retrieving, and manipulating data are POST, GET, and PUT (Fielding and Reschke, 2014). Following the URL format that was set, an example of URL usage is shown in

Figure 5.

```
URL Format:
http://[domain]/[channel-name]/[peer-name]/[function-name]

Examples:
-    http://[domain]/channel0/cust1/ReadProduct
-    http://[domain]/channel3/supp1/CreateProduct
-    http://[domain]/channel3/courier/UpdateShipmentLocation
```

Figure 5. Example of Proposed REST API URL Format.

3 Experimental Results and Discussion

To measure the scenario fulfilment of the proposed system, we conducted the black box testing approach. In software development, black box testing measures the satisfaction of requirements based on the software input and output without considering the software mechanism or code (Nidhra and Dondeti, 2012). However, there are many ways to evaluate a software, e.g., based on performance, cost, interface, or attack test. This paper focused on delivering halal information through a blockchain-based system. System optimization can be a topic for future research. Two previous studies that we found also conducted black box testing by simulating a few cases for test halal blockchain-based systems (Surjandari et al., 2021; Chandra et al., 2019).

The test was conducted in two phases. In the first phase, thirty cases were determined based on the ideal scenario that is described in 0. Using the dummy data in Appendix, a summary of cases and results from the first phase is shown in Table 4. Thirty out of thirty cases were successfully executed.

In the current state, all dummy data are stored in the system. For the second phase, we tested the restriction channel of the proposed system 0) by retrieving other organization data. We switched data for retrieval between organizations based on the first phase; sixteen cases were determined. Seven out of sixteen returned a failure response since they were not a member of the channel where the data belonged. Nine out of sixteen cases returned a success response. Case numbers 9b-16b returned a success response because the product details and batch details were on *channel0*, where the members are all organizations listed. Case number 7b also returned a success response because *supp3* is a member of *channel4*, as listed in Table 2.

To conduct the test, we deployed the proposed system on one machine using virtualization software called Docker (Docker, 2021) to imitate the nodes required by the scenario. We simulated all cases using a Phyton script. The program's tasks were to send HTTP requests to the REST API server and to record the responses from those requests. A sample of the REST API response is shown in Figure A2-1. Source code, dataset, and complete results of the tests are available at (Akbar, 2021).

Table 4.

The List of Test Cases of First Phase and Their Results.

#	Description	Is Success
1a	***Supp1*** store **product** details with ID: **111111 000001** on ***channel0***	True
2a	***Supp2*** store **product** details with ID: **222222 000001** on ***channel0***	True
3a	***Supp3*** store **product** details with ID: **333333 000001** on ***channel0***	True
4a	***Supp1*** store **batch** details with ID: **111111 000001 21010301** on ***channel0***	True
5a	***Supp2*** store **batch** details with ID: **111111 000001 21010301** on ***channel0***	True
6a	***Supp3*** store **batch** details with ID: **111111 000001 21010301** on ***channel0***	True
7a	***Supp1*** store **invoice** details with ID: **111111 210105001** on ***channel1***	True
8a	***Supp2*** store **invoice** details with ID: **222222 210105001** on ***channel2***	True
9a	***Supp2*** store **invoice** details with ID: **222222 210105002** on ***channel3***	True
10a	***Supp3*** store **invoice** details with ID: **333333 210107001** on ***channel4***	True
11a	***Courier*** store **shipment** details with ID: **777777 2101050001** on ***channel1***	True
12a	***Courier*** store **shipment** details with ID: **777777 2101050002** on ***channel2***	True
13a	***Courier*** store **shipment** details with ID: **777777 2101050002** on ***channel3***	True
14a	***Courier*** store **shipment** details with ID: **777777 2101070001** on ***channel4***	True
15a	***Cust1*** retrieve **shipment** details with ID: **777777 2101050001** on ***channel1***	True
16a	***Cust2*** retrieve **shipment** details with ID: **777777 2101050002** on ***channel2***	True
17a	***Supp3*** retrieve **shipment** details with ID: **777777 2101050002** on ***channel3***	True
18a	***Cust3*** retrieve **shipment** details with ID: **777777 2101070001** on ***channel4***	True
19a	***Cust1*** retrieve **invoice** details with ID: **111111 210105001** on ***channel1***	True
20a	***Cust2*** retrieve **invoice** details with ID: **222222 210105001** on ***channel2***	True
21a	***Supp3*** retrieve **invoice** details with ID: **222222 210105002** on ***channel3***	True
22a	***Cust3*** retrieve **invoice** details with ID: **333333 210107001** on ***channel4***	True
23a	***Cust1*** retrieve **batch** details with ID: **111111 000001 21010301** on ***channel0***	True
24a	***Cust2*** retrieve **batch** details with ID: **222222 000001 21010301** on ***channel0***	True
25a	***Supp3*** retrieve **batch** details with ID: **222222 000001 21010301** on ***channel0***	True
26a	***Cust3*** retrieve **batch** details with ID: **333333 000001 21010501** on ***channel0***	True
27a	***Cust1*** retrieve **product** details with ID: **111111 000001** on ***channel0***	True
28a	***Cust2*** retrieve **product** details with ID: **222222 000001** on ***channel0***	True
29a	***Supp3*** retrieve **product** details with ID: **222222 000001** on ***channel0***	True
30a	***Cust3*** retrieve **product** details with ID: **333333 000001** on ***channel0***	True

Table 5.
The List of Test Cases in Second Phase

#	Description	Is Success
1b	***Cust1*** retrieve **shipment** details with ID: **777777 2101050002** on ***channel2***	False
2b	***Cust2*** retrieve **shipment** details with ID: **777777 2101050002** on ***channel3***	False
3b	***Supp3*** retrieve **shipment** details with ID: **777777 2101070001** on ***channel4***	False
4b	***Cust3*** retrieve **shipment** details with ID: **777777 2101050001** on ***channel1***	False
5b	***Cust1*** retrieve **invoice** details with ID: **222222 210105001** on ***channel2***	False
6b	***Cust2*** retrieve **invoice** details with ID: **222222 210105002** on ***channel3***	False
7b	***Supp3*** retrieve **invoice** details with ID: **333333 210107001** on ***channel4***	True
8b	***Cust3*** retrieve **invoice** details with ID: **111111 210105001** on ***channel1***	False
9b	***Cust1*** retrieve **batch** details with ID: **222222 000001 21010301** on ***channel0***	True
10b	***Cust2*** retrieve **batch** details with ID: **222222 000001 21010301** on ***channel0***	True
11b	***Supp3*** retrieve **batch** details with ID: **333333 000001 21010501** on ***channel0***	True
12b	***Cust3*** retrieve **batch** details with ID: **111111 000001 21010301** on ***channel0***	True
13b	***Cust1*** retrieve **product** details with ID: **222222 000001** on ***channel0***	True
14b	***Cust2*** retrieve **product** details with ID: **222222 000001** on ***channel0***	True
15b	***Supp3*** retrieve **product** details with ID: **333333 000001** on ***channel0***	True
16b	***Cust3*** retrieve **product** details with ID: **111111 000001** on ***channel0***	True

4 Conclusion

With its unique method of storing data, the blockchain system offers safety and transparency at the same time in an auditable way because it is immutable, append-only, ordered, time-stamped, open, transparent, secure, and consistent (Drescher, 2017). In this paper, we have shown how using blockchain can enhance the traceability of the halal food supply chain through the system design described in 'Proposed System Design'. The proposed system was designed based on halal food implementation in Indonesia, with a chicken slaughtering factory as a case study (Vanany et al., 2020). We also conducted a test on the proposed system using 46 test cases, as described in 'Experimental Result'. Based on the test results, the proposed system fulfilled the needs of tracing the halal food supply chain. Customers can precisely trace the halal information of the items that they have bought from the supplier. In contrast to a paper-based system, the data in the proposed system is processed faster and more transparent because of the blockchain system. However, the halal information can still be made more comprehensive by adding more information about item treatments. Ideally, implementing the proposed system in the real world can increase consumer trust and satisfaction. Future research needs to implement this system on other halal food cases such as beef, lamb, and duck meat, which are commonly consumed in Indonesia, to measure the fitness of the system. The proposed system may also be applied in other supply chain fields like drugs and cosmetics that fall under LPPOM MUI's jurisdiction.

References

Akbar, A. (2021). *sensasi-delight/csbn-test: Rev Final 1.5 (*September 13). doi:10.5281/zenodo.5502144.

Bumblauskas, D., Mann, A., Dugan, B., Rittmer, J. (2020). A blockchain use case in food distribution: Do you know where your food has been? *International Journal of Information Management,* **52**: 1-10.

Chandra, G.R., Liaqat, I.A., Sharma, B. (2019). Blockchain redefining: The halal food sector. *2019 Amity International Conference on Artificial Intelligence (AICAI),* IEEE, pp. 349-354.

Chatterjee, R., Chatterjee, R. (2017). An Overview of the Emerging Technology: Blockchain. *3rd International Conference on Computational Intelligence and Networks (CINE),* pp. 126-127.

Chokhani, S., Ford, W., Sabett, R., Merrill, C., Wu, S. (2003). *RFC3647: Internet X. 509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*. (IETF Datatracker, November). doi:10.17487/RFC3647.

Docker (2021). *Docker Desktop for Mac and Windows (May 15).* Retrieved from Docker: https://www.docker.com/products/docker-desktop.

Drescher, D. (2017). *Blockchain Basics: A Non-Technical Introduction in 25 Steps.* Frankfurt am Main, Germany: Apress.

Fielding, R.T., Reschke, J. (2014). *RFC 7231 - Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content (June).* Retrieved April 30, 2021, from https://tools.ietf.org/html/rfc7231.

Hanifatunnisa, R., Rahardjo, B. (2017). Blockchain Based E-Voting Recording System Design. *11th International Conference on Telecommunication Systems Services and Applications (TSSA), pp. 1-6.*

Hew, J.-J., Wong, L.-W., Tan, G.W.-H., Ooi, K.-B., Lin, B. (2020). The blockchain-based Halal traceability systems: a hype or reality? *Supply Chain Management: An International Journal,* **25**(6): 863–879. doi:10.1108/SCM-01-2020-0044.

Hyperledger (2018). *Channels — hyperledger-fabricdocs master documentation.* September. Retrieved May 16, 2021, from https://hyperledger-fabric.readthedocs.io/en/release-2.2/channels.html.

Hyperledger (2020a). *Blockchain network — hyperledger-fabricdocs master documentation*. June. Retrieved May 12, 2021, from https://hyperledger-fabric.readthedocs.io/en/release-2.2/network/network.html.

Hyperledger (2020b). *Smart Contract Processing — hyperledger-fabricdocs master documentation*. August. Retrieved April 29, 2021, from https://hyperledger-fabric.readthedocs.io/en/release-2.2/developapps/smartcontract.html.

Hyperledger (2021). *Fabric CA User's Guide — hyperledger-fabric-cadocs main documentation*. March 14. Retrieved May 12, 2021, from https://hyperledger-fabric-ca.readthedocs.io/en/latest/users-guide.html.

Jallad, N.A. (2008). The concepts of al-halal and al-haram in the Arab-Muslim culture: a translational and lexicographical study. *Language design: journal of theoretical and experimental linguistics,* **10**: 77-86.

Jamil, F., Hang, L., Kim, K., Kim, D. (2019). A Novel Medical Blockchain Model for Drug Supply Chain Integrity Management in a Smart Hospital. *Electronics,* **8**(5): 505.

Khattak, J.Z., Mir, A., Anwar, Z., Abbas, G., Khattak, H.Z., Ismatullah, H., and others. (2011). Concept of Halal Food and Biotechnology. *Advance Journal of Food Science and Technology*, 3(5): 385-389.

Latif, I.A., Mohamed, Z., Sharifuddin, J., Abdullah, A.M., Ismail, M.M. (2014). A comparative analysis of global halal certification requirements. *Journal of Food Products Marketing,* **20**(sup1): 85-101.

Nidhra, S., Dondeti, J. (2012, June). Black Box and White Box Testing Techniques-A Literature Review. *International Journal of Embedded Systems and Applications,* **2**(2): 29-50.

Nurjaya, N., Paramarta, V., Dewi, R.R., Kusworo, K., Surasni, S., Rahmanita, F., Sunarsi, D. (2021). Halal Tourism in Indonesia: Regional Regulation and Indonesian Ulama Council Perspective. *International Journal of Criminology and Sociology,* **10**: 497-505.

OpenJS Foundation (n.d.). *Express - Node.js web application framework*. Retrieved May 12, 2021, from https://expressjs.com/.

Oxford University Press (2009). *Ḥalāl - Oxford Reference*. Retrieved May 12, 2021, from https://www.oxfordreference.com/view/10.1093/acref/9780195305135.001.0001/acref-9780195305135-e-0292.

Presiden Republik Indonesia (2021). Peraturan Pemerintah Republik Indonesia Nomor 39 Tahun 2021 (February 2). Jakarta, Indonesia.

Queiroz, M.M., Telles, R., Bonilla, S.H. (2019). Blockchain and supply chain management integration: a systematic review of the literature. *Supply Chain Management: An International Journal,* **25**(2): 241–254. doi:10.1108/SCM-03-2018-014.

Ratanamaneichat, C., Rakkarn, S. (2013). Quality assurance development of halal food products for export to Indonesia. *Procedia-Social and Behavioral Sciences,* **88**: 134-141.

Surjandari, I., Yusuf, H., Laoh, E., Maulida, R. (2021). Designing a Permissioned Blockchain Network for the Halal Industry using Hyperledger Fabric with multiple channels and the raft consensus mechanism. *Journal of Big Data,* **8**(1): 1-16.

Tan, A., Gligor, D., Ngah, A. (2020). Applying Blockchain for Halal food traceability. *International Journal of Logistics Research and Applications*, **25**(6): 947-964. doi:10.1080/13675567.2020.1825653.

Tan, K.H., Ali, M.H., Makhbul, Z. M., Ismail, A. (2017). The impact of external integration on halal food integrity. *Supply Chain Management: An International Journal,* **22**(2): 186-199.

Vanany, I., Hua Tan, K., Siswanto, N., Arvitrida, N.I., Pahlawan, F. M. (2021a). Halal six sigma framework for defects reduction. *Journal of Islamic Marketing,* **12**(4): 776-793. doi:10.1108/JIMA-11-2019-0232.

Vanany, I., Maarif, G.A., Soon, J.M. (2019). Application of multi-based quality function deployment (QFD) model to improve halal meat industry. *Journal of Islamic Marketing,* **10**(1): 97-124. Retrieved from http://clok.uclan.ac.uk/22478/1/.

Vanany, I., Pramesti, N.S., Rakhmawati, N.A., Soon, J.M. (2021b). Blockchain Adoption for Halal Food Integrity in Indonesian Food Industry. *Proceedings of the Second Asia Pacific International Conference on Industrial Engineering and Operations Management.* Surakarta, Indonesia, September 14-16, 2021.

Vanany, I., Rakhmawati, N.A., Sukoso, S., Soon, J. (2020). Indonesian Halal Food Integrity: Blockchain Platform. *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, pp. 297-302.

Wahyuni, H.C., Vanany, I., Ciptomulyono, U. (2018). Food Safety and Halal Food Risks in Indonesian Chicken Meat Products: An Exploratory Study. *2018 IEEE International Conference On Industrial Engineering And Engineering Management (IEEM)*, pp. 1011-1015.

Wang, Y., Han, J.H., Beynon-Davies, P. (2019). Understanding blockchain technology for future supply chains: a systematic literature review and research agenda. *Supply Chain Management: An International Journal,* **24**(1): 62–84. doi:10.1108/SCM-03-2018-0148.

Yani, A., Suryadi, R.A., Nurrohman, N. (2020). Stunning on Animals Slaughter on Islamic Law Perspective. *Jurnal Ilmiah Al-Syir'ah,* **18**(1): 77-90. doi:10.30984/jis.v18i1.1103.

Zhang, P., White, J., Schmidt, D. C., Lenz, G. (2017). Applying Software Patterns to Address Interoperability in Blockchain-based Healthcare Apps. *arXiv preprint arXiv:1706.03700*.

Appendix 1

Table A1-1.
The List of Dummy Positions in Company.

#	ID	Name	Location
1	111111	Supplier 1	City A
2	222222	Supplier 2	City A
3	333333	Supplier 3	City A
4	444444	Customer 1	City B
5	555555	Customer 2	City C
6	666666	Customer 3	City C
7	777777	Courier	City A, B, and C

Table A1-2.
The List of Dummy Product Details.

#	ID	Name	Ingredients	Halal Certificate ID	Manufacturer Name
1	111111 000001	Carcass Chicken	Chicken	00111111000001	Supp1
2	222222 000001	Carcass Chicken	Chicken	00222222000001	Supp2
3	333332 000001	Chicken Sausage	Chicken, Salt, Garlic	00333333000001	Supp3

Table A1-3.
The List of Dummy Batches.

#	ID	Product	QTY	Ingredients
1	111111 000001 21010301	ID: **111111 000001** Name: **Carcass Chicken** Ingredients: Halal Certificate ID: **00011111000001** Manufacturer Name: **Supp1**	1000	
2	222222 000001 21010301	ID: **222222 000001** Name: **Carcass Chicken** Ingredients: Halal Certificate ID: **00022222000001** Manufacturer Name: **Supp2**	1500	
3	333333 000001 21010501	ID: **333333 000001** Name: **Chicken Sausage** Ingredients: **Chicken, Salt, Garlic** Halal Certificate ID: **00033333000001** Manufacturer Name: **Supp3**	2000	222222 000001 21010301

Table A1-4.
List of Dummy Invoices.

#	ID	Buyer ID	Items
1	111111 210105001	44444	Batch ID: **111111 000001 21010301,** QTY: **100**
2	222222 210105001	55555	Batch ID: **222222 000001 21010301,** QTY: **1000**
3	222222 210105002	33333	Batch ID: **222222 000001 21010301,** QTY: **500**
4	333333 210107001	66666	Batch ID: **333333 000001 21010501,** QTY: **800**

Table A1-5.
The List of Dummy Shipments.

#	ID	Invoices	History
1	777777 2101050001	ID: **111111 210105001** QTY: **100**	Date: **2021-05-01,** Time: **10:00** Location: **Pick up at origin** Date: **2021-05-01,** Time: **11:00** Location: **Arrived at City A Facility** Date: **2021-05-01,** Time: **12:00** Location: **Departed from City A Facility** Date: **2021-05-02,** Time: **03:00** Location: **Arrived at City B Facility** Date: **2021-05-02,** Time: **08:00** Location: **Departed from City B Facility** Date: **2021-05-02,** Time: **10:00** Location: **Arrived at destination**
2	777777 2101050002	ID: **222222 210105001** QTY: **1000**	Date: **2021-05-01,** Time: **16:00** Location: **Pick up at origin** Date: **2021-05-01,** Time: **17:00** Location: **Arrived at City A Facility** Date: **2021-05-01,** Time: **18:00** Location: **Departed from City A Facility** Date: **2021-05-01,** Time: **22:00** Location: **Arrived at City C Facility** Date: **2021-05-01,** Time: **22:00** Location: **Departed from City C Facility** Date: **2021-05-02,** Time: **03:00** Location: **Arrived at destination**
		ID: **222222 210105002** QTY: **500**	Date: **2021-05-01,** Time: **16:00** Location: **Pick up at origin** Date: **2021-05-01,** Time: **17:00** Location: **Arrived at City A Facility** Date: **2021-05-01,** Time: **18:00** Location: **Departed from City A Facility** Date: **2021-05-01,** Time: **19:00** Location: **Arrived at destination**
3	777777 2101070001	ID: **111111210107001** Batch ID: **333333 000001 21010501** QTY: **800**	Date: **2021-07-01,** Time: **10:00** Location: **Pick up at origin** Date: **2021-07-01,** Time: **11:00** Location: **Arrived at City A Facility** Date: **2021-07-01,** Time: **12:00** Location: **Departed from City A Facility** Date: **2021-07-01,** Time: **18:00** Location: **Arrived at City C Facility** Date: **2021-07-01,** Time: **19:00** Location: **Departed from City C Facility** Date: **2021-07-01,** Time: **20:00** Location: **Arrived at destination**

Appendix 2


```
{
  "success": true,
  "message": “read Product is
    success”
}
```

(a)

```
{
  "success": true,
  "message": "read Product is
    success",
  "data": {
    "id": "111111000001",
    "name":
      "Carcass Chicken",
    "ingredients": [
      "Chicken"
    ],
    "halalCertificateId":
      "00111111000001",
    "manufacturerName":
      "Supp1"
  }
}
```

(b)

```
{
  "success": false,
  "message":
    "DiscoveryService:
    channel2 error: access
    denied"
}
```

(c)

Figure A2-1. REST API Responses; (a) Case No. 1a; (b) Case No. 27a; (c) Case No. 7b.