A Novel IoT Authorization Architecture on Hyperledger Fabric with Optimal Consensus using Genetic Algorithm

Nuttapong Klaokliang, Padungpol Teawtim,
Phet Aimtongkham and Chakchai So–In, *SM*, *IEEE*
Applied Network Technology, Department of Computer Science,
Faculty of Science, Khon Kaen University
123 Mitaparb Rd., Maung, Khon Kaen, Thailand, 40002
{knuttapong, t_padungpol and phet}@kkumail.com
and chakso@kku.ac.th

Aimaschana Niruntasukrat
NECTEC, National Science and Technology Development Agency
(NSTDA) 112 Thailand Science Park,
Thanon Phahonyothin Tambon Klong Nueng, Amphoe Klong
Luang, Pathum Thani 12120, Thailand
aimaschana.niruntasukrat@nectec.or.th

***Abstract*—This paper proposes a distributed authorization architecture for Internet of Things adopting a hyperledger fabric framework. This paper also applies Genetic Algorithm to enhance the consensus component, traditionally used for Kafka, in order to determine the best configuration over different parameters, such as input transactions and their success rates. The experimental results confirm our superior performance in that the transaction transfer rate of our optimization, called GA Kafka, outperforms the traditional Kafka by 69.43%, and 89.97% for transaction transfer success rate.**

Keywords—Internet of Thing; Authorization architecture; Hyperledger Fabric; Consensus optimization

I. INTRODUCTION

Internet of Things (IoT) is a recent technology in which electronic devices can communicate and interconnect with each other via Internet. Based on Cisco white paper, D. Evans reported that by 2020, there will be more than 50–billions devices connected to the Internet [1]. Several issues have been investigated including communication, computing, and, in particular, security [2]. One of the most important security aspects also focuses authorization while a variety of IoT devices require in order to access the IoT resources and services [2].

There are many crucial steps for authorization process; for example, the device must send the data to prove the right toward a central system. With Internet connection, the device, which largely sends requests to the center, is huge in workload [3]. Thus, the authorization process will be delayed, in particular, with a centralized concept. This research, then, investigates the probable use of distributed architecture for IoT authorization adopting Blockchain on a hyperledger fabric framework.

Traditionally, Hyperledger Fabric [4] is based on a principle of Blockchain, i.e., a distributed ledger architecture, originally designed for Smart Contract [5]. Note that this architecture is open and be able to modify the composition of the consensus as appropriate for better performance [6]. There are two well–known types of Hyperledger Fabric's consensus: 1) Solo, a centralized architecture, and 2) Kafka, a decentralized architecture.

Although there is a debate over the centralized vs. distributed, in particular, in IoT, the latter is promising and that leads to the prevalent use of Kafka [7] with its key advantage on that can prevent a single point of failure for scalability and fault tolerant purposes [8]. To complete the authorization operation, Kafka requires some specific policies, such as time and size of the block before making a decision on the input transaction, such as determining the number of replicas to make a copy of data before export them to peers [7].

However, these configurations in Kafka only perform once at startup and be able to use for an entire operation, and therefore, the optimal policy configuration cannot be real–time determined. Thus, this research first investigates the probable use of a hyperledger fabric framework using Blockchain for IoT authorization architecture design and then applies Genetic Algorithm (GA) based on various parameters including transaction rates used to improve the performance of Kafka consensus

This remaining of the paper is organized as follows: we provide a brief survey on related work in Section 2. Then, Sections 3 discusses our research methodology and we provide details of our performance evaluation and analysis in Section 4. Finally, Section 5 illustrates the conclusion and point out our future research direction.

II. RELATED WORK

A. IoT Authorization Architecture

Although there are several authorization techniques used in IoT, a commonly well–known use is based on OAuth, traditionally used for Web authorization. One of the early investigations was proposed by A. Niruntasukrat *et al*. [9] who studied a possibility to apply OAuth 1.0 for IoT authorization. The key advantage is on no requirement of stored access token but only device key and secret at the actual device instead; and these will be used to request the token from the authorization server, and so, avoiding the outsider to steal the information.

In addition, P. Fremantle *et al*., further investigated the application of OAuth 2.0 [10] used for IoT authorization. Here, unlike OAuth 1.0, the device would rather not connecting directly to the authorization server excluding the

978-1-5386-7805-3/18/$31.00 ©2018 IEEE

request token process in order to reduce the system complexity. Note that, however, both schemes are still considered as centralized schemes, and so, not properly used in IoT for scalability purpose.

B. Blockchain

The concept of Blockchain was first applied to Bitcoin for digital currency [11–12] used for money exchange based on a distributed ledger architecture. There are two types of Blockchain: Permissionless and Permissioned [13–14]. The former Blockchain typically uses the consensus approach [11–13] applying a specific hardware to verify the block; however, its main limitation is time–complexity, i.e., high cost for energy consumption. In opposite, the latter is a group of trusted nodes used to create new block [13–16] with its key advantage on being able to handle more transactions (low complexity) [13] including the easy–handling for participant control. Our proposal also adapted the concept of this approach for IoT authorization using Hyperledger Fabric.

C. Consensus

Recently, there are two main consensus algorithms used in Hyperledger Fabric, i.e., Solo and Kafka [4]. Traditionally, Solo is the centralized consensus algorithm. There is only one node which performs decision making to export the block to all peers. The key limitation is over availability and scalability; and with centralized concept makes Solo not wide–spread use, i.e., currently under development [18].

Kafka is a promising decentralized consensus algorithm with its key feature to prevent a single point of failure, potentially occurs in the case of Solo (for scalability purpose) [6]. Kafka uses the specific technique to make a copy of the topic information to its members within a cluster. The system has a common agreement to export the block to add the chain in Blockchain that ensure Kafka exported the same set of data (for redundancy approach) [7].

Note that recently, there is another consensus scheme – Byzantine Fault Tolerant (BFT) based on the principle of majority vote [19]. Here, the fault nodes do not exceed the determination given Byzantine equation. In comparative with Kafka, one of the key advantages is over the customized configuration but with some specific requirements including an optimal configuration determination. Thus, in this research, we propose an optimized derivation using Genetic Algorithm (GA) on Kafka, called GA Kafka suitable for IoT authorization architecture using Hyperledger Fabric.

III. RESEARCH METHODOLOGY

A. Authorization Architecture

Our proposal is implemented based on OAuth 2.0 authorization architecture [20] to improve the authorization process. Figure 1 shows an overall IoT authorization architecture as follows: a user first requests a token from an authorization server and uses it for his own device. After that, in case the device aims to access publish/subscribe system, it connects to the broker using the granted token.

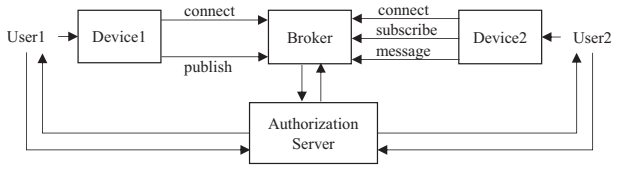

Figure 1. IoT Authorization Architecture

B. Authorization Flow

Figure 2 shows a flow of authorization process as follows: first, the user must authenticate to its peer with username and password to request the access token. When the authentication process is completed, the access token will be created. Then, the peer creates the transaction and sends it with the access token to an ordering service using consensus. The service, then, generates a block which will be sent to all peers which, after that, append to its own Ledger.

After that, a specific peer will send the access token back to the requested user. Then, the user embeds the token to the device for further access. For promptly access service, say, publish/subscribe, there requires broker, here, we adopted MQTT since it takes a minimum computational consumption in additional to well–design properly for device–to–device communication in IoT [9]. When the device connects to MQTT broker, it uses the token for authorization purpose. The broker will send the request through Hyperledger Fabric as an authorization service. After the authorization process is completed, the device can finally use the broker service.

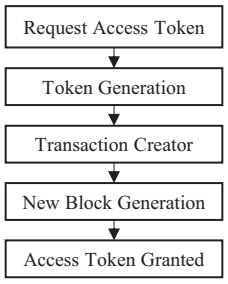

Figure 2. Authorization Flow

C. Authorization Server

The authorization server in a traditional architecture is centralized. This research tempts to modify into a decentralized one using Hyperledger Fabric. The components are as follows (See also Figure 3):

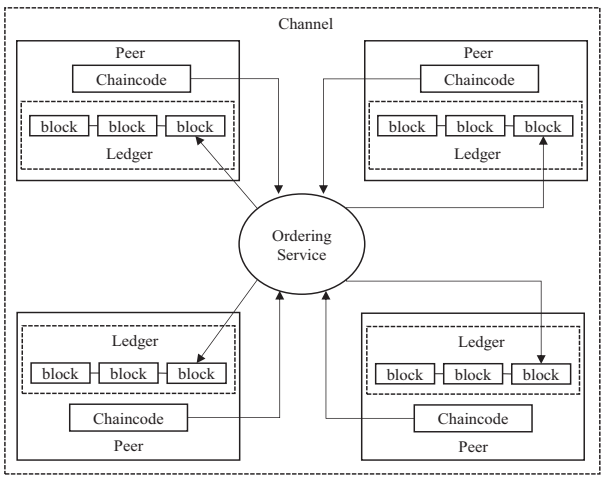

Figure 3. Architecture design of an enhanced authorization server with Hyperledger Fabric

- **Ledger:** Line of Blockchain that records transactions.
- **Peer:** Ledger storage and Smart Contract processor.
- **Chaincode:** i.e., Smart Contract that can define the transaction.
- **Channel:** A definition of peer communication with

installed Chaincode.

- **Endorsement Policy:** i.e., the agreement of each Chaincode installed that needs to be processed by peers before the consensus stage starts.
- **Ordering Service:** A specific service consisted of Ordering Service Node (OSN) is used for receiving the transaction that sends from peers to determine the block (e.g., bash of transactions). The consensus is used for that purpose, and here, we adopt Kafka as consensus. Note that a specific policy is required to complete the consensus operation such as maximum time, total block size, and the number of transactions in the block before exporting the block to peers in the channel that the block is related [6].

Figure 4 also shows the operation flow of Hyperledger Fabric as follows: first a client joins the channel to create transactions by sending a request for transaction to every peer defined in Endorsement Policy. Peers that receive the request process the transaction with Chaincode and verify whether the transaction is in the correct format. They, then, send the verification back to the client.

When a client receives all verifications, it then sends the transaction with verifications to Ordering Service. The transaction will be sorted and put into a specific block. Then, the service will send that transaction to a new block to all relevant peers in the channel. Every peer that receives the block will update their Ledger by appending a new block at the end of the Ledger.

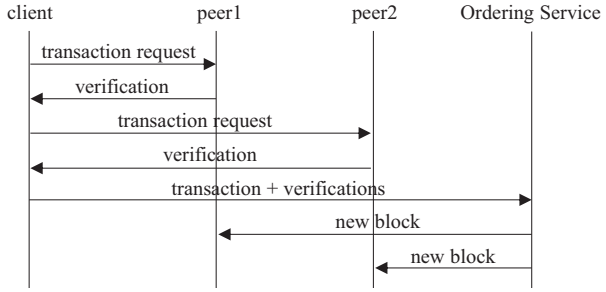

Figure 4. Hyperledger Fabric: operation flow

D. Ordering Service

Figure 5 shows an ordering service component as follows: each transaction (e.g., tx1, tx2, …, txn) will be sent toward OSN into Kafka Cluster. When the configuration policy is met, the transactions will be grouped into a specific block and after that, that block will be sent to all peers via OSNs.

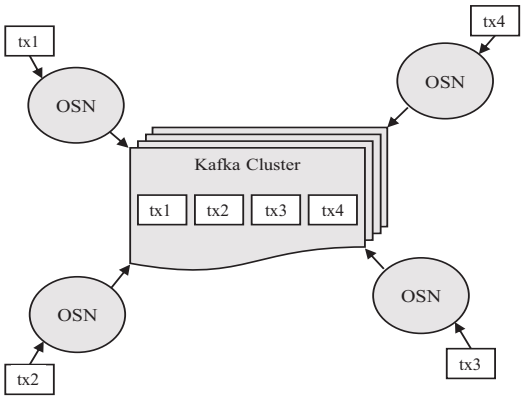

Figure 5. The Architecture of the Ordering Service that uses Kafka as the consensus algorithm

E. Consensus

Based on the principle of Blockchain, every peer must record the same set of data [12]. The most important component in this process is a consensus. Thus, this research aims to optimize the consensus to improve the authorization process for IoT architecture. The proposed algorithm is based on Hyperledger Fabric using GA to aid the optimal configuration determination process of Kafka consensus.

GA is one of the artificial intelligence techniques inspired by evolutionary and natural selection theory to determine an optimal solution given computational constraint [21]. The process begins with the definition of genes and chromosomes; then, sets the initial population by random and evolution processes including reproduction, crossover, and mutation [22]; then, evaluates fitness function; all details are as follows:

- **Genes Definition:** In this step, GA parameters are set. Three main parameters are selected to the gene variables including block size, time to waits before create the new block, and transactions limit per block.
- **Operator:** There are two primary operators, i.e., two–point crossover and random resetting mutation.
- **Fitness Function**: In each generation, every child must have a fitness value in order to determine which one is best for parenting into the next generation as illustrated in Equation (1) below.

$$Fitness = \left(\frac{r}{d}\right) \times s \quad (1)$$

where r denotes a transaction transfer per second; and d is a default transaction transfer per second. Here, s is a transaction transfer success rate.

The equation for fitness value is derived from the relationship where r is related to variable d by dividing r by d representing more efficiency of the proposed algorithm than the default configuration in terms of transaction transfer per second. Note that the multiplication factor s is used for considering the transaction transfer success rate.

Two–point crossover: Figure 6 shows an example of the two–point crossover. Here, the first chromosome, i.e., A1 (first gene), A2 (second gene), and A3 (third gene), represents the three parameters of parent A, and similar to B. Here, the two–point crossover method is used; we swap the second gene from both chromosome to generate two new chromosomes, i.e., A1, B2, and A3; and B1, A2, and B3.

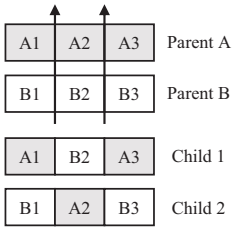

Figure 6. Two–point crossover

Resetting Mutation: Figure 7 shows the mutation process. Here, there are two randomized processes. First, we randomly select the order of the gene in each chromosome, i.e., C2 (the second gene). Then, we randomly select the new gene, i.e., $C2_{(new)}$, in the range of 10 and 1000.

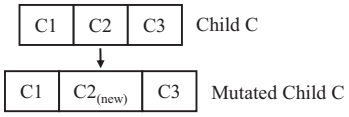

Figure 7. Random resetting mutation

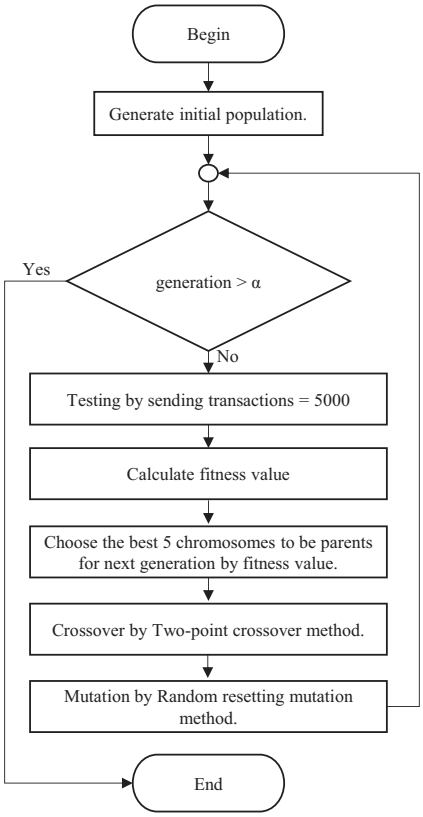

Figure 8. The consensus optimization process with the genetic algorithm

In addition, Figure 8 shows an overall process of consensus optimization as follows:

We first generate an initial population in terms of configuration (i.e., the size of the block, time to waits before creating the new block, and transaction limit per block). Note that we limit the size to 25 based on the actual experiment given the optimal performance. After that, there is a condition to limit the generation over threshold α (here is 6 based on the intensive experiment against the optimal performance).

Next, the testing process is performed by sending a number of transactions (here is 5000) to compute the fitness value. We, then, select the top five chromosomes to be added in the parent selection process. Two GA operations, i.e., crossover and mutation, will be performed on the parents to create the next children to be added in the current population. This process is iteratively continued.

IV. Performance Evaluation

A. Simulation Configuration

To confirm our proposed scheme performance, i.e., GA Kafka, here, the simulation testbed was implemented using Node–RED, a Node.js programming language as a flow–based programming typically used to simulate IoT devices [23]. In this architecture, the communication is based on MQTT protocol and we performed an authorization using Hyperledger Fabric, as shown in Figure 9 as follows:

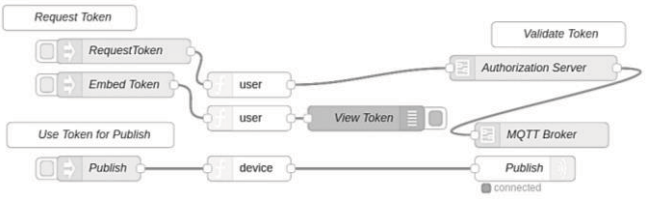

Figure 9. The simulation of IoT authorization using Node–RED

The user who needs access token makes a request (Request Token) to an authorization server (Validate Token). The token will be embedded to the user for device usage (Embed Token). Note that there is also a function to verify the token (View Token). The device with access token will connect to MQTT broker (with granted authorization) to request the service (Publish). The performance measurement focuses on the consensus component tested on Ubuntu 16.04 LST 64–bits 4 GB RAM 1TB Hard Disk that installed Hyperledger fabric version 1.0 via docker, Hyperledger Caliper, and Apache Kafka version 0.10.2 [24–27].

Considering GA configurations, there are 6 generations used for consensus optimization process, each of which has the population of 25 configurations, i.e., in total of 150 configurations. There are three main metrics including fitness values, transaction transfer rate per second, and transaction transfer success rate. The simulation is over 10 trials and provide the averages as results.

B. Simulation Result

Figure 10 shows the fitness values over round (all possible configurations = 150; each generation is 25 configurations) on GA over Kafka consensus. Over rounds, the maximum fitness value is in increasing step over the generation. For example, the maximums of 0.27, 0.63, 0.88, and 1.72 are for the 1st, 2nd, 3rd, and 4th generation, respectively. Then, the maximum one will be finally used to determine the optimal configuration.

In addition, Figure 11 shows the second metric – transaction transfer rate per second which the trend follows that of the fitness value (See also Figure 10), i.e., over rounds, the maximum rate keeps increasing; but in different values, e.g., 22, 34, 42, 62, 64, and 84, respectively. This trend also applies to transaction transfer success rate with different values, i.e., 33.84, 52.14, 58.54, 78.04, 81.12, and 99.98, respectively (See also Figure 12).

Furthermore, with the derived optimal configuration, Table 1 shows the comparison between traditional Kafka and its optimization (GA Kafka) over the latter two metrics. With 10 trails as average, GA Kafka can boost the transaction transfer per second to 32.7 compared to just 19.3 for the traditional one, and similarly, 45.22% and 23.93% for the transaction transfer success rate.

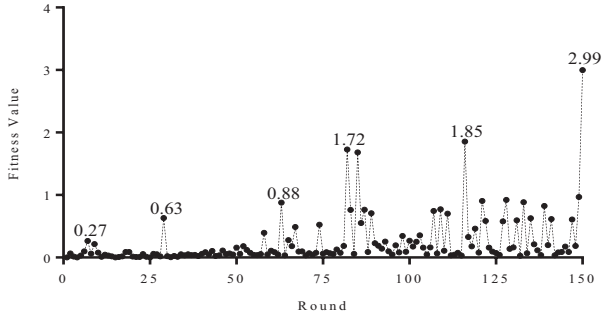

Figure 10. Fitness values for GA Kafka

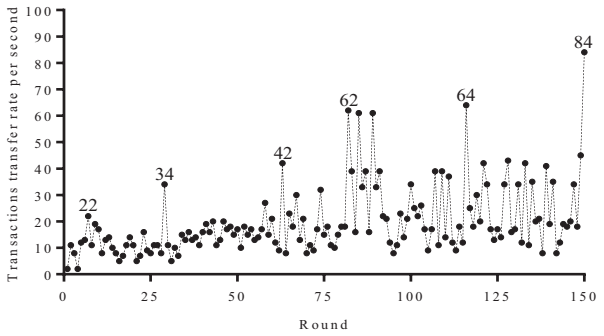

Figure 11. Transaction transfer rate per second for GA Kafka

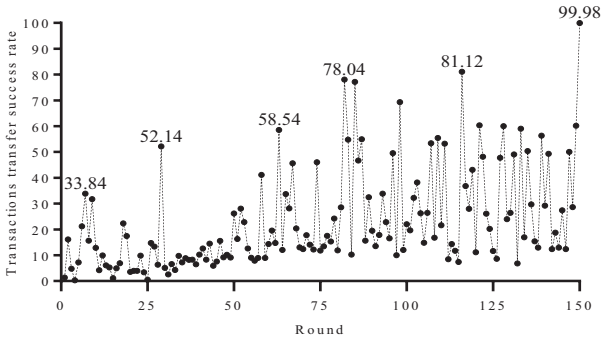

Figure 12. Transaction transfer success rate for GA Kafka

TABLE I. PERFORMANCE COMPARISON OF CONSENSUS ALGORITHMS

Performance Measurement	***Algorithm***	
	Kafka	***GA Kafka***
Transaction transfer per second	19.3	32.7
Transaction transfer success rate	23.93 %	45.22 %

V. CONCLUSIONS

A traditional IoT authorization architecture has a key limitation on a centralized service. This research investigated the integration of a hyperledger fabric framework (in use of Kafka) for the distributed authorization. The consensus is the key component for performance enhancement purpose, and therefore, we applied GA to improve the configuration selection criteria. Based on our intensive evaluation, the performance of our proposal, GA Kafka, outperforms the traditional Kafka in terms of transaction transfer per second and transaction transfer success rate, in average, by 69.43% and 89.97%, respectively.

Although the evaluation confirms our superior performance, the optimal configuration only performed once, and so over time, various real–time factors may be changed probably requiring another set of optimal configuration. Thus, an approach of self–adaptive consensus should be further studied. Also, different configurations including communication and computation overheads affecting the algorithm performance should be well–investigated, and these considerations and advancements are for future research direction.

ACKNOWLEDGMENT

This work was supported by Department of Computer Science, Faculty of Science, Khon Kaen University and NECTEC, National Science and Technology Development Agency (NSTDA).

REFERENCES

[1] D. Evans, “The Internet of things: how the next evolution of the Internet is changing everything,” Cisco Internet Business Solution Group White Paper, 2011.

[2] Z. K. Zhang, M. C. Y. Cho, C. W. Wang, C. W. Hsu, C. K. Chen, and S. Shieh, “IoT security: ongoing challenges and research opportunities,” in Proc. 2014 IEEE 7th International Conference on Service–Oriented Computing and Applications, pp. 230–234, 2014.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. “Internet of Things (IoT): A vision, architectural elements, and future directionsm,” Future Generation Computer Systems, vol. 29, no. 7, 1645–1660, 2013.

[4] Hyperledger.org, “Welcome to Hyperledger Fabric,” 2017. Available at [http://hyperledger–fabric.readthedocs.io/en/release–1.0/]

[5] C. Cachin, “Architecture of the Hyperledger blockchain fabric,” in Proc. Workshop on Distributed Cryptocurrencies and Consensus Ledgers, 2016.

[6] Hyperledger.org, “Ordering Service FAQ,” 2017 Available at [https://hyperledger–fabric.readthedocs.io/en/release–1.1/ordering–service–faq.html].

[7] K. Christidis, “A Kafka–based Ordering Service for Fabric,” 2016. Available at [https://docs.google.com/document/d/1vNMaM7XhOlu9tB_10dKnlrhy5d7b1u8lSY8a–kVjCO4].

[8] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, “An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends,” in Proc. IEEE International Congress on Big Data, pp. 557–564, 2017.

[9] A. Niruntasukrat, C. Issariyapat, P. Pongpaibool, K. Meesublak, P. Aiumsupucgul, and A. Panya, “Authorization mechanism for mqtt–based internet of things,” in Proc. 2016 IEEE International Conference on Communications Workshops, pp. 290–295, 2016.

[10] P. Fremantle, B. Aziz, J. Kopecký, and P. Scott, “Federated identity and accessmanagement for the internet of things,” in Proc. 2014 International Workshop on Secure Internet of Things, pp. 10–17, 2014.

[11] S. Nakamoto, “Bitcoin: A peer–to–peer electronic cash system,” 2008. Available at [https://bitcoin.org/bitcoin.pdf].

[12] J. Garay, A. Kiayias, and N. Leonardos, “The bitcoin backbone protocol: Analysis and applications,” in Proc. International Conference on the Theory and Applications of Crytographic Techniques, pp. 281–310, 2015.

[13] M. Vuklić, “The quest for scalable blockchain fabric: Proof–of–work vs. BFT replication,” in Proc. International Workshop on Open Problems in Network Security, pp. 112–125, 2015.

[14] J. Kwon, “Tendermint: Consensus without Mining,” 2014. Available at [https://tendermint.com/static/docs/tendermint.pdf].

[15] W. Martino, “Kadena: The first scalable, high performance private blockchain,” 2016. Available at [http://kadena.io/docs/Kadena–ConsensusWhitePaper–Aug2016.pdf].

[16] M. Vuklić, “Rethinking permissioned blockchains,” in Proc. ACM Workshop on Blockchain, Cryptocurrencies and Contract, pp. 3–7, 2017.

[17] C. Cachin and M. Vuklić. “Blockchains Consensus Protocols in the Wild,” arXiv preprint arXiv:1707.01873, 2017.

[18] F. christopher, “Hyperledger Fabric Ordering Service,” 2017. Available at [https://github.com/hyperledger/fabric/blob/master/orderer/README.md].

[19] J. Sousa, A. Bessani, and M. Vuklić, “A byzantine fault–tolerant ordering service for the hyperledger fabric blockchain platform,” arXiv preprint arXiv:1709.06921, 2017.

[20] D. Hardt, “The OAuth 2.0 authorization framework,” 2012.

[21] D. E. Goldberg and J. H. Holland, “ Genetic algorithms and machine learning,” Machine Learning, vol. 3, no. 2, pp. 95–99, 1988.

[22] J. Carr, “An introduction to genetic algorithms,” Senior Project, 1, 40, 2014. Available at [https://www.whitman.edu/Documents/Academics/Mathematics/2014/carrjk.pdf]

[23] R. Lea, “Node–RED: Lecture 1 – A brief introduction to Node–RED,” 2016. Available at [http://noderedguide.com/nr–lecture–1/].

[24] Ubuntu.com, “Ubuntu,” 2018. Available at [https://www.ubuntu.com].

[25] Y. Baohua, “Hyperledger Fabric All–in–one Docker image,” 2016. Available at [https://github.com/yeasy/docker–hyperledger–fabric].

[26] Z. Haojun, “A blockchain benchmark framework to measure performance of multiple blockchain solutions,” 2018. Available at [https://github.com/hyperledger/caliper].

[27] Kafka.apacha.org, “APACHE kafka A distributed streaming platform,” 2017. Available at [https://kafka.apache.org/].