Time Estimation for a New Block Generation in Blockchain-Enabled Internet of Things

Halgamuge, Malka N.; Munasinghe, Geetha K.; Zukerman, Moshe

Published in:
IEEE Transactions on Network and Service Management

Published: 01/02/2024

Document Version:
Post-print, also known as Accepted Author Manuscript, Peer-reviewed or Author Final version

Publication record in CityU Scholars:
Go to record

Published version (DOI):
10.1109/TNSM.2023.3316394

Publication details:

Citing this paper
Please note that where the full-text provided on CityU Scholars is the Post-print version (also known as Accepted Author Manuscript, Peer-reviewed or Author Final version), it may differ from the Final Published version. When citing, ensure that you check and use the publisher's definitive version for pagination and other details.

General rights
Copyright for the publications made accessible via the CityU Scholars portal is retained by the author(s) and/or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights. Users may not further distribute the material or use it for any profit-making activity or commercial gain.

Publisher permission
Permission for previously published items are in accordance with publisher's copyright policies sourced from the SHERPA RoMEO database. Links to full text versions (either Published or Post-print) are only available if corresponding publishers allow open access.

Take down policy
Contact lbscholars@cityu.edu.hk if you believe that this document breaches copyright and provide us with details. We will remove access to the work immediately and investigate your claim.

Download date: 18/03/2024
© 2023 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Time Estimation for a New Block Generation in Blockchain-Enabled Internet of Things

Malka N. Halgamuge, Senior Member, IEEE, Geetha K. Munasinghe, and Moshe Zukerman, Life Fellow, IEEE

Abstract— The Internet of Things (IoT) has emerged with Distributed Ledger Technology (DLT) to address existing scalability challenges and improve the trustworthiness of machine-to-machine communication. Among the numerous potential benefits of combining IoT and DLT, Blockchain, a subset of DLT, is a crucial enabler to accelerate secure IoT adoption. Appending a new block to a blockchain, especially in a blockchain-based IoT ecosystem, requires more delay than expected. This delay is one of several issues limiting the broader adoption of blockchain within the IoT domain. To assess this delay, we develop a new comprehensive model to estimate the time required to generate a new block in a blockchain-enabled IoT system. To this end, we develop sub-computation models and compare time consumption associated with the block generation process by conducting an extensive analysis of the following selected IoT layers: device layer, cluster head layer, fog/edge layer, and cloud layer. Our study identifies potential time-consuming steps in adding a new block to a network. Our results demonstrate that the type of blockchain framework and data encryption algorithms could affect the block generation time and that Avalanche, Conflux, Algorand, Polkadot Hyperledger Fabric outperforms Ethereum in terms of block generation time in IoT networks. On the other hand, the blockchain framework does not play a significant role in block generation time for smaller data packets. We also observed the benefit of using 256-bit ECC (elliptic curve cryptography) encryption and the fog layer in IoT networks to enhance the scalability of the block generation process. All in all, our results indicate that the total block generation time varies depending on the selected IoT framework, data encryption algorithm, blockchain type, and key functions of the layers.

Index Terms—Blockchain, Distributed Ledger Technology (DLT), new block, Internet of Things (IoT), fog computing, cloud computing, encryption algorithm, computation time.

I. INTRODUCTION

With the advanced technologies in the Internet era, people’s lifestyles have been enhanced in different ways. The Internet of Things (IoT) is one of the key technologies facilitating smart life opportunities with daily substances connected to the Internet. Furthermore, it aids consumers in delivering customized solutions based on specific user requirements across multiple platforms. The main reason for IoT adoption is to derive intelligent information from the available data to make real-time decisions, analyze performance, and predict parameters. However, the full potential of the IoT has not been achieved yet due to factors such as security loopholes, lack of standards, heterogeneous behavior of devices, and varied communication protocols. As forecast by Holst 2021 [1], the number of connected devices (IoT) worldwide will almost triple from 8.74 billion in 2020 to more than 25.4 billion IoT devices in 2030. Correspondingly, scalability and the current centralized architecture in the IoT network to handle authentication, authorization, and device connections will be a bottleneck in the future world. Therefore, blockchain technology has been used to mitigate weaknesses in security [2], [3], privacy, scalability, and transaction transparency within the IoT framework.

A blockchain is a Distributed Ledger Technology (DLT) that allows storing a linked list of records as a chain with additional information for transactions. It is an alternative computer model with key strengths such as tracking and recording each transaction information and unmodifiable transaction records. Thus, blockchain technology provides a secure and cost-effective mechanism for transactions compared to traditional methods. Blockchains are available in different types, such as private (permissioned) and public (permission-less). Bitcoin, Ethereum, Cardano, Avalanche, Conflux, Algorand, and Polkadot are considered public blockchains, and Hyperledger Fabric is an example of a private blockchain. Depending on the application requirements, users have to adopt a suitable blockchain type. Estimation of the time for generating a new block is especially important because blockchains are computationally expensive and require significant bandwidth overhead and long latency. The PoW-based blockchain framework, in particular, requires a significant time duration to mine blocks. For example, a typical PoW, such as Bitcoin, takes 10 minutes to mine each block [4]. Estimation of the generation time of a new block can indicate whether or not a particular PoW is suitable to meet the Quality of Service (QoS) requirements of specific IoT applications.

Using blockchain technology in the IoT framework delivers several advantages [5], such as (i) the ability to track and analyze any activity of the chain for authorized officers, (ii) improvements of the overall security, and (iii) creating smart contracts to execute agreements if specific conditions are available. It is essential to make real-time decisions in the IoT
by analyzing the data transmitted from millions of connected devices.

A. Motivation

In a blockchain, every transaction in a decentralized system must undergo several processes requiring substantial processing power and time [6]. Each data transaction must go through various processes, including acceptance, mining, dissemination, and validation by a network of nodes. A new block generation time is straightforward and independent of the delay in IoT layers (fog/edge/cloud) when IoT applications are not considered. In blockchain-enabled IoT systems, however, the delay on the various IoT layers directly impacts the block generation time, affecting the time required for data to be added to the blockchain.

To add IoT transactions to the blockchain, the block generation time across the whole IoT system should be less than the block interval (time) of the chosen blockchain framework. Generating a model to estimate the total block generation time will assist the professionals in discovering the areas that need to be considered to reduce the total response time. So far, however, there has been little discussion about the total computation time in the IoT framework. Even though some studies [7]–[9] have focused on generating models for time consumption in different areas in the sensor network and the IoT framework, a complete processing time estimation model for the entire framework has not been generated.

Therefore, our model analyzes all functions in each IoT layer and proposes a new, comprehensive, time estimation model for understanding the impact of Blockchain-Enabled IoT on the block generation process and evaluating the block generation time in the IoT environment (device layer, cluster head layer, fog/edge layer, and cloud layer). In addition, in our proposed model, transaction queuing time and blockchain network latency are taken into account.

Block time or block processing time is the required time duration to create a new block or the time it takes to mine a block or file in a blockchain. Block processing time is the actual time within a network to validate transactions for one block and add a new block to the blockchain. Based on the blockchain type, the different blockchain frameworks require different time durations. For example, the estimated block time in Bitcoin is 10 minutes, whereas Ethereum’s is between 10 and 19 seconds.

The effectiveness of encryption algorithms outside the scope of blockchain applications has been extensively explored, e.g., [10]–[12], as encryption helps protect information and sensitive data and can improve the security of communication between client apps and servers. For blockchain applications, the encryption time will account for a significant portion of the total time spent on the block creation process. No existing work provided an evaluation of the time to estimate generating a new block in a blockchain-enabled IoT network. In this study, we explore and analyze encryption methods that are crucial when selecting a blockchain framework.

B. Main Contributions of this Paper

The main contributions of this paper include the following.

- We develop a new comprehensive time-computation model of appending a new block to a blockchain to estimate the new block generation time in a blockchain-based IoT ecosystem. This requires the development of sub-time computation models for each IoT layer (device layer, CH layer, fog/edge layer, and cloud layer).
- We identify each IoT layer’s extreme time consumption scenarios where we simulate our new time computation model under different circumstances (varying data packet size, blockchain type, and encryption algorithm).
- Using the model, we also identify the significance of the fog layer for the block validation process by which we achieve improvement of the efficiency of the entire block generation process.
- We discover using our proposed model that the existing Hyperledger Fabric, Ethereum, Avalanche, Conflux, Algorand, Polkadot blockchain frameworks, and 256-bit ECC (elliptic curve cryptography) encryption algorithm enhances the performance and scalability of the block generation process in the blockchain-enabled IoT platform.
- We observe based on our new model that the blockchain framework does not play a significant role in block generation time for smaller data packets.

C. Structure of Paper

The remainder of the paper is organized as follows. Section II explains the background and related work. Section III highlights key layers of the IoT architecture, their main functions, and the total computation time required to generate a new block in the blockchain-based IoT architecture. Section IV presents the simulation setup. Section V demonstrates results and discussion in Section VI. Section VIII concludes the paper and outlines possible future improvements (Section VII) for the IoT and the blockchain technology.

II. BACKGROUND AND RELATED WORK

Most blockchain implementations that integrate into IoT systems, such as Ethereum (Geth or Parity implementations) and Hyperledger Fabric, generate blocks regardless of whether sensing data arrives at the blockchain network. This block time is generally stable and primarily determined by the hardware infrastructure and a blockchain network’s consensus protocol configurations. For instance, the PoW consensus protocol of Ethereum has a mechanism to auto-adjust the mining difficulty to ensure that the block time is stable around the desired figure, regardless of the total hash rate of the network.

Previous literature has noted the importance of improving the scalability of the IoT framework without slowing down the transaction processing speed. Different studies have been conducted to improve the efficiency of the data processing in the IoT layers using different mechanisms, such as (i) introducing energy optimization techniques [9], [17], [18], different network architectures [10], [19], [20] for sensor network, (ii) integrating edge and fog layers into the IoT framework with enhancements [8], [21]–[24], (iii) introducing various validation processes for the blockchain [25], and (iv)
DLT-based IoT data trading over the narrowband Internet-of-Things system [26], [27]. Table I shows a summary of some of the prior studies that used blockchain-based IoT networks. However, in reviewing the prior studies, not much evidence was found to assess the importance of the time consumption of each significant function when processing a block in the IoT.

Another significant aspect is the overall delay in the block generation process due to different factors. Numerous studies have attempted to explain why the slowness of generating blocks such as network delay, consensus process in the blockchain, service delivery delay, response time, limited capabilities of the IoT devices, and data processing time within the different IoT layers. Damiano et al. [13] use novel and innovative hybrid blockchain architecture to increase the overall performance of block generation time by decreasing the memory capacity. Lei et al. [15] consider the cryptographic processing time of the intelligent transportation system (ITS) and propose a security key management module to improve block generation by enhancing the security of the process.

Moreover, a strong relationship between block generation time and the blockchain type has been reported in the literature. Table VII and Table VIII (Appendix) show a comprehensive comparison of different features of blockchain frameworks to determine the most efficient blockchain type. The comparison between, for example, block interval, median block propagation time, block mining time, and cross-chain interoperability of the different blockchain types: Bitcoin, Litecoin, Ethereum, Hyperledger Fabric, NEO, Cardano, EOS, Algorand, Conflux, Binance (BNB), Polkadot (DOT), and Avalanche are presented. Several models have used Bitcoin and Ethereum for their studies. However, when considering the block interval time and processing time, the advantage of using those types for real-time decision-making systems is arguable.

Jang et al. [6], Zhou et al. [12], and Liang et al. [14] simulate sample modules to generate a block using the IoT architecture. However, the cluster head (CH) layer has not been taken into consideration in all models. In the model, developed by Liang et al. [14] to secure collected data by drones, they utilize the Bitcoin blockchain technology and Public Key encryption as the cryptographic algorithm. Jang et al. [6] do not consider the data security in their model and do not mention the packet size for the simulation. Zhou et al. [12] have not included fog/edge layer in their simulation and used time-consuming data securing steps in the simulation model.
III. MODEL DESCRIPTION

Adopting blockchain technology is one of the practical ways of improving scalability in the IoT framework. We develop a new comprehensive time-computation model by using the process employed in the basic IoT framework to generate a new block to a blockchain, as shown in Figure 1. We develop the following sub-computation models (corresponding to subsection A to F in this section) to compute the total time required to generate a new block in the IoT framework.

A: Layer 1, Child Sensors - Sensing, data logging, transient, data encryption, sensor delay, communication time to CH;
B: Layer 2, Cluster Head (CH) - Data retrieval time from child sensors, switching, actuation, cluster data cryptographic, data processing, data communication time to fog server;
C: Layer 3, Fog/Edge Computing - Data communication time from all connected CHs, data processing, blockchain data consensus process, fog layer cryptographic, data communication time to the cloud server;
D: Layer 4, Cloud Server - Data communication time from fog servers in the network, cryptographic operational time, block mining processing time;
E: Blockchain Network Delay - Impact of transactions queuing delay, the influence of block size on new block generation time, block mining, block propagation, and smart contract execution time.

For simplicity, we assume a linear relationship between the size of the data and the estimated time of a new block to be added to the blockchain. This linear relationship is realistic for cases when the data size is large which is the case in many IoT applications.

In this section, we propose our comprehensive model for estimating block generation time in the IoT framework. Block generation time in the IoT system should be smaller than the block interval (time) of a chosen blockchain framework in order to add IoT transactions to the blockchain. Therefore, such a comprehensive model is required to formulate the time consumption for key functions in the IoT layers (Figure 1) and next develop the total time computation model for the new block generation process in the IoT network.

A. Total Computation Time for Layer 1 (Child Sensors)

Sensor node operation is divided into several phases: (i) initializing, (ii) sensing, (iii) computing, (iv) transmitting data to the parent sensor, (v) sleeping, and (vi) waking up (for energy saving) [9]. However, encryption and delay time should be added to the existing functionality. The timing is divided into wake-up time, active time, sleep time, transient time, data encryption time, and delay. The sensor performs its functions as rounds. During a lifetime of a sensor, it can perform several rounds of functions that repeat the phases from (i) to (vi) in each round. The total time taken to sense and identify a single transaction is the sum of each phase in a single round (Figure 1).

Different sensors behave in different ways depending on the sensor type, where different time slots are required to conduct the same function based on the data type. For example, the sensing time for a humidity sensor is less than the time required for a visual sensor used for image processing.

Consider Layer 1 has \( n \) child sensors designated \( 1, 2, \ldots, n \), where \( L_1 = \{1, 2, \ldots, n\} \). Next, we identify the time required to complete each sensor function (Layer 1) in detail for sensing time, sensor data logging time, sensor data encryption time, sensor delay time, data communication time to CH, and total child sensor computation time.

a) Sensing time \( (t_s) \): The principal function of the sensor is to sense the conditions of an environmental phenomenon and transmit data to parent nodes. The time taken to sense 1 bit of data is considered as \( t_{s1} \). The total time taken to sense \( b_1 \) bits, \( t_{s1} \), is given by

\[
t_{s1} = b_1 t_{s1},
\]

(1)

b) Sensor Data logging time \( (t_{dl}) \): The sensed data is logged into the sensor memory. The time required to read and write \( b_1 \) bits of data, \( t_{dl} \), is given by

\[
t_{dl} = b_1 (t_{w} + t_{r})
\]

(2)

where \( t_r \) and \( t_w \) represent the time taken for reading and writing bit data.

c) Sensor Transient time \( (t_{tr}) \): To save battery power and as an energy-saving mechanism, sensors transit to sleep mode every milli/microsecond (depending on the application) after an active period. Therefore, they spend time to transit between operation modes such as active, sleep, and idle. For a single transaction, the transient time \( t_{tr} \) is a combination of sleep time and the active time given by

\[
t_{tr} = t_{off} + t_{on}
\]

(3)

where \( t_{off} \) is the sleep time, and \( t_{on} \) is the active time.

d) Sensor Encryption time \( (t_e) \): Data encryption \( t_e \) is performed before transmitting to ensure security during the communication via the network. Encryption time varies depending on the sensor voltage, as explained by Yuan and Qu [28], and the data packet size. For \( b_1 \) data, the sensor data encryption time is given by

\[
t_{e} = b_1 t_{e}
\]

(4)

where \( t_{e} \) is the time to encrypt 1 bit of data.

e) Sensor delay time \( (t_{sd}) \): The delay of the data communication between sensor nodes is derived as the sum of the few different factors named: (i) sender processing delay \( t_{sp} \), (ii) media access delay (contention delay) \( t_{ms} \), (iii) receiver processing time \( t_{rp} \), (iv) radio propagation time \( t_{ip} \), and (v) network traffic \( t_{tf} \) [29]. Therefore, we determine the maximum delay in transmitting a data packet \( t_{tc} \)

\[
t_{tc} = t_{ts} + t_{tm} + t_{tr} + t_{ip} + t_{tf}
\]

(5)

f) Data communication time to CH \( (t_{cm}) \): After receiving data, it is communicated/transmitted to the associated parent node for further analysis. We estimate the time required to transmit \( b_1 \) bits data, \( t_{cm} \)

\[
t_{cm} = b_1 t_{cl} + t_{tc} + \tau
\]

(6)
where $t_d$ is the time required to transmit 1 bit of data, $t_{lt}$ is the delay time, and $\tau_1$ is a random delay during the data communication time to CH.

g) Total child sensor computation time for Layer 1: ($t_s$) Sensors in a network have different behaviors and performances. For example, the time needed to process an image is more than the time required to process temperature data. Therefore, the maximum time taken for sensor functioning should be computed. We quantify the time to sense and complete a single transaction by a child sensor $t_s$:

$$t_s = t_{ts} + t_{dt} + t_{tr} + t_{e} + t_{lt} + t_{tm}.$$  

(7)

B. Total Computation Time for Layer 2 (Cluster Head (CH))

CH performs sensor functionality similar to the (low-powered) child sensors. It has sensing time, data logging time, transient time, and data communication time similar to other sensors mentioned in Section III-A. Additionally, the specific functions of the CHs are (i) retrieving data from child sensors, (ii) switching between microcontroller processing and events, (iii) performing data encryption and decryption, (iv) triggering physical events (actuating), (v) transmitting data to the next layer.

Consider Layer 2 with $m$ CHs designated $1, 2, \ldots, m$, where $L_2 = \{1, 2, \ldots, m\}$. Next, we explain the time required to complete each additional function illustrated in Figure 1 performed in the CH in detail.

a) Data retrieval time from child sensors ($t_{ctm}$): Due to limited hardware capabilities and the network traffic in the sensor network, each child sensor is assigned a separate time slot to transmit their packets to the parent sensor using Time Division Multiple Access (TDMA) method [30]. It is assumed that all child sensor nodes in the cluster (Figure 1) transmit their data to the CH asynchronously by utilizing the TDMA mechanism. Therefore, we compute the total time required to transmit data from child sensors, $t_{ctm}$ to CH

$$t_{ctm}(i) = \sum_{i=1}^{m} t_{tm(i)} + \tau_2$$  

(8)

where $t_{tm}$ is the data communication time to CH and $\tau_2$ is a random delay during the data retrieval time from child sensors.
b) Switching Time \( t_{\text{cas}} \): The time required for sensors to switch into different operational modes depends on the microcontroller processing time and the time between switching events [9]. We derive the switching time of a CH \( t_{\text{cas}} \), to switch between the \( \alpha \) number of microcontrollers and the \( \beta \) number of switching events (sensing different measurements based on the different environmental conditions such as object present or not, maximum and minimum level detection of the object)

\[
t_{\text{cas}}(k, j) = \alpha \sum_{j=1}^{\beta} t_{\text{cas}(j)} + \beta \sum_{k=1}^{\beta} t_{\text{se}(k)}
\]

(9)

where \( t_{\text{cas}} \) is the time required for micro-processing, and \( t_{\text{se}} \) is the time to switch between events.

c) Actuation time \( t_{\text{act}} \): The CHs provide mechanical responses based on the inputs they receive, such as turning on a light and activating motor sensors. Additionally, they perform as actuators to trigger physical events based on the received values from other sensors and themselves. We derive the total actuation time to trigger the \( \beta \) number of events \( t_{\text{act}} \)

\[
t_{\text{act}}(k) = \sum_{k=1}^{\beta} t_{\text{act}(k)}
\]

(10)

where \( t_{\text{act}} \) is the actuation time required to trigger a single event. The time varies with the event’s length, and this is used only if the sensor is an actuator.

d) Cluster data cryptographic time \( t_{\text{ck}} \): Encrypted data sent by the child nodes are decrypted and processed by the CH and encrypted again prior to the transmission to the fog layer. We estimate the total time taken to the encryption and decryption process in CH \( t_{\text{ck}} \), is defined by

\[
t_{\text{ck}} = (b_1 + b_2)(t_{\text{ck}} + t_{\text{dk}})
\]

(11)

where \( t_{\text{ck}} \) is the sensor encryption time and \( t_{\text{dk}} \) is the decryption time for 1 bit.

e) Data processing time \( t_{\text{dp}} \): Data for a single transaction is transmitted to a CH from multiple child sensors. CH uses data fusion mechanisms and rules to deliver reliable and accurate information to the next layer [31]. It is assumed that the time taken for data processing in CH is \( t_{\text{dp}} \). The time depends on the data type and the application requirements.

f) Data communication time to fog server \( t_{\text{tmm}} \): Time taken to transmit data to the fog layer from the CH is measured as \( t_{\text{tm}} \). The communication time depends on the data volume transferred to the next layer.

As illustrated in Figure 2, the study considers that the number of data packets transferred from the \( n \) number of child sensors is different from the sizes of data packets transferred from the respective CH. Since the CH is more powerful than child sensors, it can transmit more data. Similarly, different percentages of data are transferred to the next layer from each available cluster in the sensor network. Furthermore, the transmission of data packets between CH and the fog layer is delayed due to network delay. Therefore, we estimate the total time required to transmit CH data to the fog server

\[
t_{\text{tmm}} = \max_{n \in L_1} ((b_2 + nb_1q)P t_{\mu} + t_{\text{it}})
\]

(12)
where \( Q_1, Q_2, \ldots, Q_x \) are measures transferred from each cluster to their parent CH, \( t_v \) is the data communication time from the \( v^{th} \) child layer to its parent layer, \( q \) is the percentage of total data sending to the next CH layer from each cluster, and \( P \) is the total percentage of data transmitted to the fog layer. Depending on the sensor network (IoT) structure, we can use either Equation (14) or Equation (15). However, for simplicity, in this study, we use Equation (14).

C. Total Computation Time for Layer 3 (Fog/Edge Computing)

The fog/edge layer assists the IoT framework by conducting communication, routing, storage, and computation [32]. Data received from CHs are analyzed using fog computing before sending the final value to the cloud service. Furthermore, since some IoT sensors are unable to process raw data, the fog layer aids in improving the efficiency of the transaction [11]. In this study, five main functions of the fog server are considered: (i) data gathering from all associated CHs, (ii) data processing, (iii) blockchain validation process, (iv) data decryption and encryption, (v) data communication or transmission to cloud server.

Consider Layer 3 with \( r \) fog nodes designated 1, 2, \ldots, \( r \), where \( L_3 = 1, 2, \ldots, r \). Next, we describe the time required to complete each function in the fog layer.

a) Data communication time from all connected CHs (\( t_{f3m} \)):

To evaluate \( t_{f3m} \), we consider communication delays in all three layers as follows.

1) Sensor Layer - (i) sender processing delay \( t_{ls} \), (ii) media access delay (contention delay) \( t_{lm} \), (iii) receiver processing time \( t_{lr} \), (iv) radio propagation time \( t_{lrp} \), and (v) network traffic \( t_{lp} \) and random delay (\( r_1 \) and \( r_2 \)) during the data communication time.

2) Fog Layer - network delay, random delay

3) Cloud Layer - network delay, transactions queuing delay.

The fog server receives values from different CHs to perform analysis and other functionalities. Due to the synchronized data receiving capability of the fog layer, we estimate the total time required to transmit data from multiple CHs,

\[
t_{f3m} = \max_{m \in L_3} t_{f3m} \tag{16}
\]

where data \( t_{f3m} \) is the data communication time from the CH to the fog layer.

b) Data processing time (\( t_{fp} \)):

When the fog node successfully retrieves values, it starts processing the transaction data to generate information. For example, when multiple sensors sense the temperature from a large area and send values to different CHs, those CHs send finalized values to the fog nodes based on their received information. The time consumption for data processing time is measured as \( t_{fp} \) and it depends on the number of transactions and data amount to be processed.

c) Time for the blockchain data consensus process (\( t_{fc} \)):

It is essential to follow a consensus mechanism for data before adding a new transaction to the existing blockchain [10]. The different blockchain frameworks follow different mechanisms to validate the new transactions to mitigate different types of attacks. Different blockchain types have different mechanisms for performing the consensus process. For instance, bitcoin utilizes the Proof of Work (PoW) system, Ethereum uses PoW and PoS (Proof of Stake), and Hyperledger Fabric utilizes Byzantine Fault Tolerance (BFT) as their consensus algorithms [33]. The total consensus time depends on the selected blockchain type. Here we suggest performing a consensus process in the fog/edge layer to improve the total performance of the cloud layer. The advances in effective resource management in the fog layer can enhance the reliability of cloud facilities by minimizing process latency. The consensus time of a transaction to validate \( h \) bytes of data in the fog server \( t_f \) is measured by

\[
t_f = h(t_{fe} + t_{fd}) \tag{17}
\]

where \( t_f \) is the time required to process the consensus mechanism for the 1 byte of data (1 byte = 8 bits).

d) Fog layer cryptographic time (\( t_{fk} \)):

The fog layer performs the cryptographic process to decrypt data sent by sensors and encrypt them before sending it to cloud service. The time to encrypt and decrypt a file depends on the file size and the cipher algorithm [34]. Therefore, we determine the time required to encrypt and decrypt \( h \) bytes of data, \( t_f \),

\[
t_{fk} = h(t_{fe} + t_{fd}) \tag{18}
\]

where \( t_{fe} \) and \( t_{fd} \) represent the time taken to encrypt and decrypt 1 byte of data using desired ciphertext length.

e) Data communication time to the cloud server (\( t_{cj} \)):

If the transaction is accepted by the validation process of the fog layer, it is transmitted to the cloud server, which is the highest layer of the IoT framework. We evaluate the total time to transmit data to the cloud, \( t_c \),

\[
t_c = \max_{r \in L_3} (t_{ft} + t_{fj}) \tag{19}
\]

where \( t_f \) is the time to transmit \( h \) bytes from the fog layer to the cloud server, and \( t_{fj} \) is the network delay.

f) Total fog/edge server processing time (\( t_{FG} \)):

From the combinations of the equations in the section, we add together the total time consumption in the fog layer \( t_{FG} \),

\[
t_{FG} = t_{f3m} + t_{fp} + t_{fc} + t_{fk} + t_{cj} \tag{20}
\]

D. Total Computation Time for Layer 4 (Cloud Server)

The cloud server, the final layer of the IoT platform, is used as the main block management network in blockchain technology. It acts as the final layer with (i) final data validation with other cloud servers, (ii) block mining, and (iii) block propagation activities in the blockchain.

Consider Layer 4 with \( s \) cloud servers designated 1, 2, \ldots, \( s \), where \( L_4 = 1, 2, \ldots, s \). Next, we explain the time required to complete each sub-function in the cloud.

a) Data communication time from fog servers in the network (\( t_{ct} \)):

Since different fog networks are connected to a single cloud service, a cloud server receives data from multiple fog layers for a transaction. Therefore, we quantify the cloud has to wait until all fog servers finish transmitting data, \( t_{ct} \),

\[
t_{ct} = \max_{r \in L_3} t_{cj} \tag{21}
\]
where \( t_{ej} \) is the data communication time to the cloud.

b) Cryptographic operational time (\( t_{ec} \)): We estimate Time taken to encrypt and decrypt \( \delta \) MB data in the cloud server \( t_{ck} \)

\[
t_{ck} = \delta (t_{ce} + t_{cd})
\]  
where \( t_{ce} \) and \( t_{cd} \) represent the time required to encrypt and decrypt 1 MB data in the cloud.

c) Total cloud computation time (\( t_{CS} \)): From the combinations of the equations in the section, we derive the total time consumption in the cloud \( t_{CS} \) to generate a single block

\[
t_{CS} = t_{cd} + t_{ck}.
\]

E. Computation Time for Blockchain Network Layer (\( t_{bn} \))

The IoT data layer sends transactions to the blockchain network layer. The application layer is in charge of data processing and end-user service delivery. The Blockchain network layer maintains the blockchain while interacting with the application layer in both directions. We compute the time required for this process.

a) Impact of Transactions Queuing Delay in M/D/1 Queue Model for Blockchain-based IoT Network (\( t_{bD} \)): It is likely that, in some cases, new transactions arrive before the previous transactions are served and added to the blockchain. Therefore, a buffer is used to store transactions waiting to be connected to the blockchain. We model the waiting time in the queue (including the service time) of transactions \( t_{bD} \) to join the blockchain using queuing theory and select the M/D/1 queuing model.

The assumption that arriving transactions follow a Poisson process \( M \) is justified by the fact that these transactions are generated by many independent users. Because blockchain transactions are processed normally one at a time by a server, the single server assumption is justified. In our model, we assume for simplicity that the service time — specifically, the time taken to add a transaction to a given blockchain — is fixed (deterministic).

This assumption is grounded on several reasons: In many IoT applications, the size and complexity of transactions are approximately fixed, especially when devices are programmed to send standardized data packets required for adding transactions to a blockchain at regular intervals [35], [36]. While IoT networks can experience congestion, many real-time critical IoT applications are designed to operate under controlled network conditions, reducing the variability in transaction processing times [37]. For IoT applications requiring high real-time performance, the network and devices often use optimized protocols and infrastructures to ensure minimal required delay [38]. Therefore, while there might be slight fluctuations in service times, we can approximate them as deterministic for our analysis.

It is crucial to note that while our model adopts this deterministic assumption for simplicity, in practice, service times can vary based on numerous factors. Future work may explore stochastic models to account for such possible variability.

In addition, the assumptions of unlimited buffer and first-in-first-out (FIFO) scheduling are also consistent with blockchain practice. Although, in practice, the arrival rate over time may vary, our implied assumption of a constant arrival rate is overcome in this paper by considering different levels of queue utilization.

Let \( \lambda \) be the transaction arrival rate (transactions per second), \( \mu \) be the service rate (transactions per second), and \( \rho \) be the queue utilization factor, where \( \rho = \lambda / \mu \). As shown in [39], the mean transactions queuing delay in the system of the M/D/1 queue is given by

\[
t_{D} = \frac{1}{2\mu} \times \frac{2 - \rho}{1 - \rho}.
\]

b) Influence of Block Size on New Block Generation Time (\( t_{bs} \)): The capacity of a block or a block size is equal to the amount of data it can hold. A block, like any other container, may only hold a certain amount of data or transactions in the context of our work. Once a transaction is accepted after the mining process, the accepted transactions are ordered and packed into a block. Thus, the number of transactions that can fit into one block depends on the block size. Block sizes are limited on different blockchains. For example, the average block size is up to 4 MB for Avalanche and Polkadot, 2 MB for Bitcoin, Conflux, and Binance, 1 MB for Hyperledger Fabric, Litcoin, EOS, and Dogecoin, 20-30 kB for Ethereum. Additionally, some transactions are lightweight. Let \( s_b \) be the block size in MB, \( s_t \) the transaction size in MB, \( n_t \) the number of pending transactions to be added to a block, and \( \mu \) the transactions per second [tps]. As the ratio \( s_b / s_t \) is the number of transactions required per block and \( n_t / \mu \) is one transaction time, we evaluate the time to create one complete block \( t_{bs} \)

\[
t_{bs} = \frac{s_b}{s_t} \times \frac{n_t}{\mu}.
\]

c) Total block mining time: (\( t_{mn} \)) An existing blockchain is updated using two phases named block mining \( t_{mn} \) and block propagation. The time it takes for miners or validators in a blockchain network to validate transactions within a single block is known as mining time. Block generation (creation), validation, and integration to the transaction of the existing blockchain are termed block mining [40]. The time required for block mining in blockchain-based IoT systems can be influenced by various factors, including the specifics of the transaction and the role of centralized systems like the cloud layer. However, the mining process fundamentally relies on the consensus mechanism adopted by the blockchain and the collective efforts of participating nodes. Therefore time to mine a block is measured by \( t_{mn} \), which depends on the number of transactions \( x \) [15] in the blockchain.

d) Block propagation time (\( t_{pp} \)): Block propagation time is a well-known barrier in blockchains that prohibits any blockchain framework from scaling. Stale blocks are mine blocks that are excluded from the main chain due to simultaneous transactions, conflicts, or network propagation delays. The block propagation time is an essential factor influencing a blockchain framework’s scalability and stale block rate. When a block is added to an existing blockchain, it is broadcasted via the entire network to reach most network nodes using a suitable propagation mechanism. The average time it takes for
a new block to reach the majority of nodes in a network is called block propagation time.

The research revealed that extended propagation delay could lower a node’s ability to withstand 51% attacks and selfish mining attacks. Blockchain developers aim to minimize block propagation time to mitigate this security issue, typically targeting it to be less than 1% of the average block time [41]. Therefore, although for some blockchain frameworks, this target has not yet been achieved, in this study, we use this 1% target to calculate the block propagation time in cases where more reliable data on block propagation time is unavailable. Due to fluctuations in network conditions, the median block propagation time may not be consistently accurate.

c) Smart contract execution time (\(t_{sc}\)): We assume each blockchain has a designated target smart contract function which the submitter always signs. As a result of processing the transactions submitted by users, the blockchain node executes a smart contract. The transaction is submitted to any blockchain node responsible for distributing that to the entire blockchain network. After the transaction is distributed, it is processed by each node using the executable program in the target smart contract. We determine the time taken to execute the smart contract \(t_{sc}\).

\[ t_{bc} = t_{D} + t_{ba} + t_{mn} + t_{pp} + t_{sc}. \]  

(26)

F. Total Computation Time Required to Generate a New Block in the IoT Framework

From the combination of equations (7), (13), (20), (23), (24), and (26), we compute the total computation time to sense an environmental phenomenon until it generates as a new block to an existing blockchain in the IoT platform \(t_T\)

\[ t_T(i, j, k) = \max_{m \in L_1, L_2} t_{SN} + t_{ctm}(i) + t_{ctd}(j, k) + t_{tp}(k) + t_{ei} + t_{et} + t_{ec} + t_{D} + t_{ba} + t_{mn} + t_{pp} + t_{sc}. \]  

(27)

From the summations of equations (1) – (26), we further simplify the total computation time (27) as

\[ t_T(i, j, k) = \max_{m \in L_1, L_2} t_{SN} + t_{ctm}(i) + t_{ctd}(j, k) + t_{tp}(k) + t_{ei} + t_{et} + t_{ec} + t_{D} + t_{ba} + t_{mn} + t_{pp} + t_{sc}. \]  

(28)

Thus, this is the estimated time required to generate a new block in the blockchain-enabled internet of things (IoT).

G. Requirement for a New Block Generation Time for IoT

Once we obtain the total computation time to sense an environmental phenomenon until it generates a new block to an existing blockchain in the IoT platform, we need to observe which blockchain platform is suitable for the specific IoT applications. To add the IoT transactions to the blockchain, block generation time in the IoT should be less than the block interval (time) of a particular blockchain framework. Hence, we obtain the comparison

Block generation time in IoT\((t_T) < \) Block interval (time). \( \text{(29)} \)

Depending on the application requirements (such as smart agriculture, smart home, and smart transportation system), users could choose the suitable blockchain framework type using the proposed model.

IV. SIMULATION DESIGN

A simulation is carried out to evaluate the proposed model. The analysis is performed using MATLAB (MathWorks Inc., Natick, MA, USA) R2021b on a computer with macOS Monterey with Processor 2 GHz Quad-Core Intel Core i5 and RAM (Random Access Memory) 16 GB 3733 MHz LPDDR4X. All parameter values used for the simulation model are presented in Table II. The experimental values we use in this analysis are taken from peer-reviewed literature. Hence, we believe our results represent a general IoT setup.

A sensor network is used with 100 sensor nodes that are connected to their respective CHs, and the same simulation settings utilized in [9] are applied for the sensor network.
is assumed that all connected child sensors are homogeneous and take a similar time to transmit data to the respective CH. The sensor network we consider in our simulation consists of powerful CHs, fixed clusters, and single-hop transmission. Using the above simulation settings, we generate 1,000,000 random setups, enabling each simulation data point to obtain 1,000,000 random setups. Depending on the IoT application (or use case), we can determine how many hours/days the simulation will run. IoT in agriculture, collects data from sensors every hour, whereas IoT in smart cities collects data every two minutes.

The fog and cloud simulation setup used is similar to the simulation conducted by Naas et al. [8]. The size of data packets generated is assumed to be increased by 10 percent in each subsequent layer due to the added overheads (that is, if the sensor generates 1 kB data packet, the fog layer adds an overhead of 100 bytes). Moreover, we assume that the network is a permissioned network which increases the security of the block generation process. We use Ethereum, Hyperledger Fabric, Algorand, Conflus, Polkadot, and Avalanche blockchain for the simulation model. As the experiments performed by Gervais et al. [16] and Malik et al. [46] the respective median block propagation times in seconds are 1.02, 0.85, 0.5-0.75, and 0.075. Furthermore, to ensure data security, we consider the data encryption algorithms, such as Advanced Encryption Standard (AES), Data Encryption Standard (DES), Triple DES (DES3), Rivest Cipher 2 (RC2), Rivest Cipher 6 (RC6), Standard (AES), Data Encryption Standard (DES), Triple DES (DES3), Rivest Cipher 2 (RC2), Rivest Cipher 6 (RC6), Blowfish, 256-bit ECC (elliptic curve cryptography), 2048-bit RAS (Rivest-Shamir-Adleman) and RAS-3072 in the study.

V. RESULTS

This study illustrates the general taxonomy of a blockchain-enabled IoT system and its contribution to the overall data acquisition delay. We develop a model to determine key time-consuming areas of the IoT network to generate a new block in an existing blockchain. A new, comprehensive, time utilization model is essential for the IoT network to improve process efficiency and scalability. The applicability of the proposed model is validated through simulation experiments. We compare the total block generation time with the different blockchain frameworks and data encryption algorithms in the simulation process. Some blockchain frameworks do not provide block propagation time, hence they cannot be used to compare performance. Nevertheless, the IoT issues that affect delay are applicable to other blockchains, so we use blockchains for which data is available. In addition, we analyze the time consumption of each IoT layer to identify each layer’s contribution to the total block generation process. In the simulation process, we utilize the fog layer for the block validation process to improve the efficiency of the cloud server and the total block generation process in the IoT network. The results indicate that the total block generation time varies depending on the selected IoT framework, data encryption algorithm, blockchain type, and key functions of the layers. Finally, we compare the outcome of our model with similar time computation models developed by Jang et al. [6], Liang et al. [14], and Zhou et al. [12] to identify the efficiency of our model.

<table>
<thead>
<tr>
<th>Sym</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>b1</td>
<td>Packet size transmit from child sensor</td>
<td>1 kB</td>
</tr>
<tr>
<td>b2</td>
<td>CH communication or transmission packet size</td>
<td>2 kB</td>
</tr>
<tr>
<td>n</td>
<td>Number of child sensors in a cluster</td>
<td>100</td>
</tr>
<tr>
<td>β</td>
<td>Number of switching events in a sensor</td>
<td>1</td>
</tr>
<tr>
<td>α</td>
<td>Number of microcontrollers in a sensor</td>
<td>1</td>
</tr>
<tr>
<td>l</td>
<td>Number of validation cycles</td>
<td>1</td>
</tr>
<tr>
<td>q</td>
<td>Data percentage from sensors to fog</td>
<td>70%</td>
</tr>
<tr>
<td>P</td>
<td>Total data percentage to fog</td>
<td>80%</td>
</tr>
<tr>
<td>h</td>
<td>Data packet size in the fog layer</td>
<td>10 kB</td>
</tr>
<tr>
<td>δ</td>
<td>Data packet size in the cloud server</td>
<td>100 kB</td>
</tr>
<tr>
<td>s</td>
<td>Sensor sensing time</td>
<td>0.0002 ms [9]</td>
</tr>
<tr>
<td>sk</td>
<td>Sensor encryption time (AES)</td>
<td>0.7346 ms [34]</td>
</tr>
<tr>
<td>sb</td>
<td>Sensor decryption time (AES)</td>
<td>1.2857 ms [34]</td>
</tr>
<tr>
<td>w</td>
<td>Sensor data writing time</td>
<td>0.00645 ms [9]</td>
</tr>
<tr>
<td>f</td>
<td>Sensor data reading time</td>
<td>0.0003 ms [9]</td>
</tr>
<tr>
<td>τ1</td>
<td>Random delay (Layer 2 - CHs)</td>
<td>vary</td>
</tr>
<tr>
<td>τ2</td>
<td>Random delay (Layer 1 - Child sensors)</td>
<td>vary</td>
</tr>
<tr>
<td>t_p</td>
<td>Sensor propagation delay time</td>
<td>0.002 ms [17]</td>
</tr>
<tr>
<td>t_s</td>
<td>Sensor processing delay for communication</td>
<td>0.0006 ms [18]</td>
</tr>
<tr>
<td>t_d</td>
<td>Data transmission time to CH</td>
<td>0.0002 ms [29]</td>
</tr>
<tr>
<td>t_m</td>
<td>Media access delay</td>
<td>0.0001 ms</td>
</tr>
<tr>
<td>t_r</td>
<td>Receiver processing delay after data receiving</td>
<td>0.0006 ms [18]</td>
</tr>
<tr>
<td>t_f</td>
<td>Network traffic</td>
<td>0.0002 ms</td>
</tr>
<tr>
<td>t_s</td>
<td>Transient time to switch on</td>
<td>2.45 ms [9]</td>
</tr>
<tr>
<td>t_o</td>
<td>Transient time to switch off</td>
<td>0.25 ms [9]</td>
</tr>
<tr>
<td>t_c</td>
<td>Microcontroller processing time</td>
<td>0.0001 ms</td>
</tr>
<tr>
<td>t_s</td>
<td>Switching time between different events</td>
<td>0.002 ms [42]</td>
</tr>
<tr>
<td>t_a</td>
<td>Actuation time</td>
<td>0.0025 ms [19]</td>
</tr>
<tr>
<td>t_p</td>
<td>Data processing time of CH</td>
<td>0.14 ms [20]</td>
</tr>
<tr>
<td>t_s</td>
<td>Network delay in fog</td>
<td>1.41 ms [23]</td>
</tr>
<tr>
<td>t_d</td>
<td>Communication or transmission to fog</td>
<td>0.0006 ms [24]</td>
</tr>
<tr>
<td>t_f</td>
<td>Data processing time of fog</td>
<td>23.36 sec [7]</td>
</tr>
<tr>
<td>t_d</td>
<td>Data processing time of cloud</td>
<td>84.58 sec [7]</td>
</tr>
<tr>
<td>t_s</td>
<td>Data validation in own memory</td>
<td>0.0014 ms [25]</td>
</tr>
<tr>
<td>t_f</td>
<td>Fog layer encryption 100 kB (AES)</td>
<td>73.469 ms [34]</td>
</tr>
<tr>
<td>t_d</td>
<td>Fog layer decryption 100 kB (AES)</td>
<td>116.32 ms [34]</td>
</tr>
<tr>
<td>t_f</td>
<td>Data communication or transmission time from fog to cloud</td>
<td>9.01 ms [8]</td>
</tr>
<tr>
<td>t_v</td>
<td>Data validation time in cloud</td>
<td>0.17 ms [8]</td>
</tr>
<tr>
<td>t_c</td>
<td>Network delay of the cloud server</td>
<td>17.99 ms [23]</td>
</tr>
<tr>
<td>t_e</td>
<td>Encryption time in the cloud 1MB (AES)</td>
<td>187.12 ms [34]</td>
</tr>
<tr>
<td>t_d</td>
<td>Decryption time in the cloud 1 MB (AES)</td>
<td>120.82 ms [34]</td>
</tr>
<tr>
<td>μ</td>
<td>Service rate in the queue (transactions per second, tps)</td>
<td>20 (Ethereum) [43], 3500 (HyperLedger Fabric) [44], up to 4500 (Avalanche) [45]</td>
</tr>
<tr>
<td>n_t</td>
<td>Number of pending transactions</td>
<td>100 Tx</td>
</tr>
<tr>
<td>s_t</td>
<td>Average transaction size</td>
<td>380.04 bytes</td>
</tr>
<tr>
<td>s_b</td>
<td>Block size</td>
<td>≈ 20 KB - 2 MB</td>
</tr>
<tr>
<td>t_pp</td>
<td>Median block propagation time</td>
<td>Refer Table VIII</td>
</tr>
</tbody>
</table>
A. Overall New Block Generation Time Based on the Blockchain Type

Block time or block processing time is the time length it necessitates to create a new block or the time it carries to mine a block or file in a blockchain. Block processing time is the actual time within a network to validate transactions for one block and add a new block to the blockchain. Based on the blockchain type, the different blockchain frameworks take different times (see Table VII and Table VIII in Appendix). The blockchain framework decides block interval time; for example, the estimated block time in Bitcoin is 10 minutes, whereas Ethereum’s is between 10 and 19 seconds. Since the Bitcoin blockchain framework is computationally expensive and requires high bandwidth overhead and delays, it may not be suitable for most IoT applications. Further, it is well known that in Bitcoin, new blocks of size approximately 1 megabyte are mined every 10 minutes on average. Thus, the available data rate is about 6 megabytes/hour, 100 kilobytes/minute, or 1.67 kilobytes/second. This is well below the speed of most wireless communication technologies available today. Hence, it is not possible for an IoT system that would generate data at this rate. Thus, we remove the Bitcoin blockchain from our analysis.

The simulation is conducted to measure the overall block preparation time with the identified functions. The initial results show that cryptographic time and block mining time utilize the maximum time consumption of the IoT block generation process. Therefore, firstly, we observe the most appropriate blockchain type for the IoT network.

Few key blockchain frameworks, such as Ethereum, Hyperledger Fabric, Algorand Conflux, Polkadot, and Avalanche are associated with the simulation (Figure 3). We compare a small number of blockchain frameworks because the median block propagation time (Equation (26)) for all blockchain systems was not provided. In contrast, the blockchain framework does not play a significant role in block generation time for smaller data packets. The most evident result of the analysis is that the block processing time in the IoT network is less than the block interval times of the given blockchain types when the data packet size is below 110 kB.

Figure 3 indicates that Avalanche has the shortest block processing time. Due to the real-time decision-making requirements of the IoT network, it is essential to provide the information and decisions immediately to the end user. Therefore, in the results, it is observed that Avalanche, Hyperledger Fabric, Conflux, and Polkadot are the most time-efficient blockchain types for the IoT network.

B. Overall New Block Generation Time Based on Encryption Algorithm

Since encryption helps secure information and sensitive data and can improve the security of communication between client applications and servers, the efficiency of encryption algorithms has been extensively studied. Further, the encryption time will account for a significant portion of the total time spent on the block creation process. As a result, it is critical to thoroughly research and analyze encryption algorithms in order to choose the appropriate blockchain framework.

When considering the time taken to process data encryption, it is understandable that the selected algorithm affects the total block generation time (Figure 4). The results show that the total time is increased from 1.2 - 4.3 sec based on the selected algorithm type. However, as illustrated in Figures 4(a) HyperLedger Fabric (for symmetric encryption) and 4(b) (for asymmetric encryption), no significant differences are found between the seven algorithm types when the data packet size is below 5 kB. The most striking observation to emerge from the data comparison is that the processing time of each algorithm varies with the packet size after 35 kB. This observation could be due to the performance factors of each algorithm. Furthermore, in a typical IoT network, the fog and cloud layers process more data volume than the sensor layers. Therefore, the data packet size of each layer should be analyzed before incorporating an algorithm in a specific layer. These findings suggest that the block processing time AES and ECC are the time-efficient algorithms for real-time data security in the IoT framework when data packet size is up to 110 kilobytes.

The RSA (Rivest-Shamir-Adleman) and elliptic curve encryption are two of the world’s extensively utilized asymmetric algorithms. Because of the limitations of experimental data on the time taken to process data encryption for different data sizes, we limit the complete analysis of all algorithms to 200 kilobytes. The ECC-256 and RAS-2048 show the lowest block time. Hence, in addition to another popular encryption algorithm, AES, we further observe these three encryption algorithms with four different blockchain frameworks (Hyperledger Fabric, Algorand, Conflux, Polkadot, Avalanche) as shown in Figure 5 and Figure 6. Significant differences are observed between asymmetric (AES versus BLOWFISH) and asymmetric (ECC-256 versus RAS-2048) algorithms for block generation time.
(a) Different cryptographic algorithms (symmetric) for HyperLedger Fabric.

(b) Different cryptographic algorithms (asymmetric) for HyperLedger Fabric.

Fig. 4. Average block generation time versus data packet size for different cryptographic algorithms for HyperLedger Fabric.

(a) Different symmetric (AES, 3-DES, Blowfish) encryption algorithms.

(b) Different asymmetric (ECC-256, RSA-2048) encryption algorithms.

Fig. 5. Average block generation time using different symmetric (AES, 3-DES, Blowfish) and asymmetric encryption algorithms (ECC-256, RSA-2048) and blockchain frameworks (Ethereum, Algorand, HyperLedger, Conflux, Polkadot, Avalanche).

C. Overall New Block Generation Time Based on the IoT Layers (Sensor, Fog/Edge, Cloud)

In general, Layer 3 (fog/edge layer) assists in reducing the network latency of the IoT network by bearing the additional burden from the cloud server. Further, in our model, the fog layer has been utilized to perform the blockchain consensus process to minimize the block execution time of the cloud. Typically block the validation process is conducted in the cloud server, and it takes much time because of the high network latency in the cloud server. Therefore, we propose to utilize the fog layer for that to accelerate the block generation process. Figure 7 illustrates the average processing time versus the data packet size in each IoT layer. In addition to utilizing the fog layer to bear the additional burden from the cloud server, we use it to perform the blockchain consensus process to minimize the block execution time of the cloud. As a consequence, we observe lower block generation time in the cloud layer than in the fog layer.

It is essential to observe the block generation time comparison with/without the fog/edge layer and with/without the cloud layer for the blockchain-enabled IoT networks. Thus, Figure 8 shows a block generation time comparison for 100 kB data packets. We consider two scenarios: fog only and cloud only, and then compute the time using our proposed model of the block time required by each scenario. In the first scenario (fog only) block time takes 20-40 sec while in the second scenario (cloud only) it takes around 85-100 sec. Therefore, the results of our study confirm that if we use only cloud servers, the block generation time is doubled compared to using fog servers only.
Fig. 6. Average block generation time using different encryption algorithms (ECC-256, RSA-2048) and blockchain frameworks (HyperLedger, Algorand, Conflux, Polkadot, Avalanche).

Fig. 7. Block generation time comparison between different IoT layers.

D. Overall New Block Generation Time Based on the Transactions Queuing Delay

We apply the $M/D/1$ queue model to our blockchain-based IoT context. Figure 9 shows the impact of queuing delay of transactions ($t_D$) for different queue utilization factors ($\rho$) using Equation (24).

However, we observe that the queuing time is negligible relative to the block time in the IoT network (Figure 10). In addition, this demonstrates the impact of the transaction queuing delay ($t_D$) for different queue utilization factor values ($\rho$) is higher in Polkadot, while the overall block time is lower in Conflux and Avalanche.

E. New Block Generation Time Based on the Block Size

Figure 11 shows the block time versus block size when the device processing speed is 2 GHz using Equation (25). For all blockchain platforms, block time increases as block size grows. However, since block sizes are not likely to significantly exceed 2.5 MB, the maximum increase that we observe is below 0.06 seconds and therefore this increase due to block size is insignificant relative to the block time in the IoT network.

F. Overall New Block Generation Time: Performance Evaluation Comparison

Next, we compare our model with similar time computation models developed by Jang et al. [6], Liang et al. [14], and Zhou et al. [12]. Various attempts to measure block generation time have been presented in Table III and Figure 12 with a summary of their considerations, such as the IoT layers,
TABLE III

<table>
<thead>
<tr>
<th>Model</th>
<th>Packet Size</th>
<th>Total Time (sec)</th>
<th>Blockchain Type</th>
<th>Security</th>
<th>Device</th>
<th>CH</th>
<th>Fog/Edge</th>
<th>Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our model</td>
<td>1024 bytes</td>
<td>0.495</td>
<td>Hyperledger Fabric (BFT)</td>
<td>DES</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Jang et al. [6]</td>
<td>Not Mentioned</td>
<td>4.665</td>
<td>Hyperledger Fabric (BFT)</td>
<td>Not Given</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Zhou et al. [12]</td>
<td>1014 bytes</td>
<td>7.3</td>
<td>Ethereum (PoW)</td>
<td>ECIES, ECDSA</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Liang et al. [14]</td>
<td>1000 bytes</td>
<td>2.9</td>
<td>Bitcoin (PoW)</td>
<td>PK cryptography</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

approach with a total processing time of 0.495 sec. Finally, we verify that unnecessary time consumption can be eliminated by utilizing an appropriate blockchain framework, encryption algorithm, and fog layer.

G. Real Blockchain Data Comparison for the Different Blockchain Frameworks

The block size limit refers to the maximum amount of data that a blockchain block may carry. The maximum block size determines the number of transactions taken inside a block. This size consequently controls the throughput of the system. In addition, larger block sizes bring slow block propagation speeds; thus, expanding the old block rate or stale block rate, which impacts the blockchain network’s security. Observing the impact of block size on the scalability and security of the blockchain is, hence, essential. In addition, it is vital to understand the current market of blockchain technology and cryptocurrency. Thus, we compare actual blockchain data using [47] (accessed April 2022) for different blockchain frameworks. Since there is a vast range of values, we use the logarithmic scale (log scale) to demonstrate values.

Figure 13 shows block size in kilobytes and the number of transactions per day in mega ($10^6$) from June 2020 to May 2021 for Ethereum, Dogecoin, Ripple (XRP), and Litecoin. Ethereum and Litecoin show the largest block sizes. Thus, we further investigate the number of transactions of each framework. As block sizes are large in Ethereum and Litecoin, obviously, those can carry more transactions. On top of that, we compare block time in minutes and the number of transac-
Fig. 13. Real blockchain data comparison: Block size and the number of transactions for the different blockchain framework - Ethereum, Dogecoin, Ripple (XRP), and Litecoin for May 2021 to April 2022.

Fig. 14. Real blockchain data comparison: Block time and the number of transactions for the different blockchain framework - Ethereum, Dogecoin, Ripple (XRP), and Litecoin for May 2021 to April 2022.

tions per day in mega (Figure 14) Ethereum and Ripple show very low block time. We repeated the all above simulations for Ethereum, Hyperledger Fabric, Algorand, Conflux, Polkadot and Avalanche and found the results to be consistent.

VI. DISCUSSION

Combining blockchain and IoT/Edge technology can provide trusted access and control [3] to the network and to storage facilities. It also enables distributed computing at the edge, high-efficiency computation, large-scale network facilities, and data storage while preserving blockchain security.

A comprehensive understanding of the general functions and their time consumption is an immense advantage for any application to achieve maximum efficiency. The present study is designed to determine key time-consuming areas of the IoT network while adding a block to an existing blockchain. Our study compares the total block generation time with different blockchain frameworks and data encryption algorithms. In addition, the average time consumption comparison of each layer and fog layer utilization for the block validation process can be considered as another significant instance. The results of the study indicate that the total block generation time varies depending on the selected IoT framework, data encryption algorithm, blockchain type, and key functions of the layers.

In this study, first, we analyze all significant functions performed in each IoT layer, and next, we measure the time taken for each function to identify the respective data processing time of those. One of the key findings of the study is identifying the time taken for key activities in the sensor network (sensor layer and CH layer). The findings for sensor behavior are primarily similar to the prior study done by Halgamuge et al. [9]. They have conducted their study to estimate sensor energy consumption by identifying key sensor and CH functions. However, data cryptography and network latency functions are not included in their model. Furthermore, that study was mainly focused on energy consumption and optimization. In contrast, we mainly focus on the time consumption in our model to generate a new block in the IoT network.

As another key finding in this study, we propose that the fog layer can perform a block validation process. It will assist in reducing the heavy load of the cloud layer. Several prior studies are focused on using the fog/edge layer as an intermediate layer in the IoT platform to reduce the additional burden of the cloud layer [8], [23], [24]. However, using fog/edge in the blockchain-enabled IoT architecture is not adequately investigated. Therefore, the outcome of the simulation assists future studies in using fog in the blockchain paradigm.

This study indicates that a few appropriate blockchain types to generate a new block in the IoT platform are Avalanche, Conflux, Algorand, Polkadot and Hyperledger Fabric. The block generation time in the IoT is less than the block interval time of the blockchain frameworks such as Ethereum. Moreover, due to the high data generation rate and real-time decision-making requirements in the IoT network, they cannot be used in the IoT network.

Some distributed blockchains, such as Bitcoin, Ethereum, Hyperledger Fabric, Algorand, Conflux, Avalanche, Polkadot, Cardano, EOS, NEO, and Litecoin, are permissionless, allowing any node to participate in the consensus process. Hyperledger Fabric is a permissioned blockchain; thus, it is not vulnerable to diverging ledgers (“fork”). The Orderer is in charge of packaging transactions into blocks and distributing them around the network to anchor peers who are in charge of peer discovery. Additionally, Hyperledger Fabric uses an Orderer (also known as an “Ordering node”) that does this transaction ordering, which forms an ordering service along with the other Orderer nodes. Fabric’s design uses determin-
istic consensus techniques, ensuring that each block certified by a peer is final and correct.

In the computation (2nd, 3rd, and 4th generation) blockchain frameworks on the market may support IoT applications with smart contracts. NEO [48], Cardano [49], EOS [50], R3 Corda [51], Binance (BNB) [52], Polkadot [53], Avalanche [45], Algorand [54] and Conflux [55] are a few examples.

Compared to blockchain, platforms based on DAG utilize a different ledger structure and different methods for transaction confirmation. Thus, an extension of this work for all IOTA versions is considered as future work.

Data security is one of the essential and challenging constraints associated with the IoT platform. In contrast, most of the studies ignore adhering to security procedures in their experiments and mention it as a challenge [10]. Even though blockchain has an in-built security procedure, it has been vulnerable to several attacks in the past decade [56]. Data security should be a predominant factor of any reliable electronic system when managing sensitive consumer data (such as smart health monitoring applications). Therefore, the most appropriate mechanism to preserve data within a network is using an encryption algorithm to convert plain text to ciphertext during communication. Based on that fact, we simulate different encryption algorithms to identify the most efficient algorithm for the block generation process in the IoT network.

Additionally, we observe that 256-bit ECC (elliptic curve cryptography) has the potential to be the most efficient encryption algorithm for IoT networks to enhance the performance and scalability of the block generation process. However, the robustness of the encryption algorithm depends upon the type of cryptography, key management, number of keys, and number of bits used in a key. Longer key length and data length provides high security; on the other hand, they consume more power and result in more heat dissipation. This provides the trade-off between security level and power consumption. In addition, keys with more bits utilize more computation time to encrypt data. Table IV shows this comparison for different encryption algorithms.

In the computation of the encryption time, a standard assumption is made regarding uniform hardware and software parameters. Therefore, the most appropriate mechanism to preserve data within a network is using an encryption algorithm to convert plain text to ciphertext during communication. Based on that fact, we simulate different encryption algorithms to identify the most efficient algorithm for the block generation process in the IoT network.

Additionally, we observe that 256-bit ECC (elliptic curve cryptography) has the potential to be the most efficient encryption algorithm for IoT networks to enhance the performance and scalability of the block generation process. However, the robustness of the encryption algorithm depends upon the type of cryptography, key management, number of keys, and number of bits used in a key. Longer key length and data length provides high security; on the other hand, they consume more power and result in more heat dissipation. This provides the trade-off between security level and power consumption. In addition, keys with more bits utilize more computation time to encrypt data. Table IV shows this comparison for different encryption algorithms.
implementation, ensuring consistent processing power across different devices. Nevertheless, it is essential to acknowledge that real-world encryption speeds may differ due to discrepancies in hardware specifications, software configurations, and implementation difficulties. In our calculations, we adopt a processing speed of 2 GHz and allocate 1 GB of RAM as the basis for our estimations.

To estimate the encryption time (Table IV), we employ the following equations: Time per block calculates the number of rounds and operations per round: Time per block = Number of rounds × (Number of operations per round / Processing speed); Number of blocks: calculates the total number of blocks needed to encrypt the data: Number of blocks = Data size / Block size; Time taken to encrypt: combines the number of blocks and the time per block to determine the overall encryption time: Time taken to encrypt = (Number of blocks × Time per block). These equations provide a general framework for estimating the encryption time based on the given assumptions. However, it is essential to consider that actual encryption speeds may vary due to several factors, including algorithmic efficiency, hardware capabilities, and software optimizations. The comparison and the findings of the study will assist future studies in improving their models by considering security principles.

Since crypto-currencies process a few transactions per second, the theoretical limit is usually in the low two- or three-digit range, approximately for Ethereum, 15-20 tps, and for Bitcoin, 7 tps. The parameters primarily determine the average block time, maximum block size, and minimum transaction size [57]. In practice, we cannot increase the block size. When the block size is large, it takes a long time to propagate a new block to all nodes through the wireless blockchain network. This impacts the latency and security of the network. Besides, every device may not have high network bandwidth and sufficient hardware storage capacity on top of the computation power. Thus, high demands could lead to sacrificing blockchain decentralization, which is the blockchain trilemma. On the other hand, with the advancement of technology, the future device storage capacities (hard disks) and network speed remain to grow globally; thus, larger block sizes might be convincing in the future. If network speed and storage capacities continue to enhance, a noticeable increment in block sizes may be convincing in the future. That allows high transaction rates (tps) without a notable rise in energy consumption [58]; thus, the scalability of the network will be improved.

Jang et al. [6], Liang et al. [14], and Zhou et al. [12] simulate sample modules to generate a block using the IoT architecture. However, to obtain accurate decision-making in an IoT environment, the time taken for each function is essential. For example, Jang et al. [6] ignored CH layer, Zhou et al. [12] and Liang et al. [14] ignored CH and Fog/Edge layers. Therefore, it is essential to provide all required parameters used in any experiments or simulations. Jang et al. did not consider the data security in their model and did not mention the packet size they used for their experiment. Compared to the time of those models (Jang et al. [6] 4.66 sec, Zhou et al. [12] 7.3 sec and Liang et al. [14] 2.9 sec), our model has taken the best time consumption approach with a total processing time of 0.495 sec. These were the main limitations of their work. In addition, we validate that by utilizing an appropriate blockchain framework, encryption algorithm, and edge/fog layer, unnecessary time consumption can be eliminated. While we present one of the prevalent IoT architectures, it is essential to note that not all IoT implementations mandate every layer discussed. Indeed, in certain contexts, IoT services can be effectively deployed and utilized within local networks or intranets.

The most exciting finding was that we observed the block generation time comparison with/without the fog/edge layer and with/without the cloud layer for the blockchain-enabled IoT networks. The results of our study confirm that if we use only cloud servers, the block generation time is doubled compared to using fog servers only. These findings of the current study are consistent with those of Gill et al. (2019) [7] who found the vision of fog computing and experimentally found a complement to cloud computing and an essential ingredient of the IoT. Moreover, they compared by considering different configurations (CPU GHz, RAM size, and power). Thus, based on our analysis, we can suggest that if the network uses small data packets, it is recommended to use fog or edge computing because of the network’s efficiency (low latency, low data processing time).

In reviewing the recent literature, most of the studies have considered limited functional areas of the IoT platform to increase the data processing efficiency. So far, however, there has been little discussion about individual layer functions, and no study can be found with a comprehensive-time utilization model related to the block generation process. Without having a clear picture of process functions and respective time consumption, it is difficult to identify how those improvements impact the total block generation process. Therefore, the findings in this study have several important implications for future practices. In future investigations, it might be possible to use different data packet sizes to observe the evaluations of the estimated impact. Thus, it is essential to verify the proposed model in an industrial IoT cloud and fog computing environment for practical understanding. Fog computing does not substitute the cloud but rather complements it [59]. Fog computing can work with cloud computing for the blockchain-enabled IoT. Cloud computing will be the primary force to be considered with the IoT, Big Data, Artificial Intelligence (AI), and future technologies. Thus, we recommend combining fog and cloud for the blockchain-enabled IoT depending on the requirements, such as limited scalability, less data storage, and computation capabilities.

In recent years, various blockchain technologies have been explored for integration with IoT applications, aiming to strengthen security, transparency, and decentralized control. The popularity of blockchain-based IoT may vary over time and depend on specific use cases. In this study, we use a few blockchain types to generate a new block in the IoT platform: Avalanche, Conflux, Algorand, Polkadot, and Hyperledger Fabric. Table V provides an overview of the blockchain platforms utilized in IoT applications, detailing their unique features and relevance to the domain.
A. Possible Applications (Use Cases)

We use a few case studies to demonstrate the usefulness of the suggested model in this sub-section. A wide range of businesses, in addition to financial institutions, have planned to explore the blockchain’s capabilities. IoT devices may use blockchain to enhance security and transparency in their industries such as supply chain and healthcare [6], [33].

1) Energy Trading: blockchain plays an essential role in energy trade for IoT applications [26].

2) Supply Chain and Logistics: The effectiveness of a supply chain is dependent on trust among the various stakeholders. The combination of blockchain and IoT technologies can help to improve the traceability and dependability of data along the chain [64] and end-to-end visibility.

3) Smart Homes: Smart IoT-enabled devices are becoming increasingly vital in our daily lives. The IoT blockchain allows home security systems to be controlled remotely using a smartphone. Telstra, an Australian telecommunications corporation, has used blockchain and biometric security to protect data collected by smart devices [65].

4) Agriculture: Pavo marketplace [66] is a blockchain IoT use case that delivers transparency to farmers by offering a new and smart farming technique. Pavo’s IoT hardware device, which is put on farms, collects data that is kept on the blockchain. It allows farmers to improve their agricultural operations by analyzing the data collected.

B. Comparing Use Cases: Evaluating Model Applicability

To justify various design choices of our experiments, in this subsection, we present two different case studies that contrast the sensor data collection intervals in IoT applications for agriculture and smart cities. We assume a higher frequency of data collection in smart cities allows for more real-time monitoring and rapid response to changing conditions. At the same time, agricultural IoT applications generally require less frequent data collection due to the nature of their use cases.

Block time refers to the average time it takes to create a new block in a blockchain. To calculate the appropriate block time for each use case, we need to consider the data collection frequency, the number of sensors, and the blockchain’s capacity to store the data. We assume that each data point requires the same storage space. We determine the appropriate block time by assuming that the blockchain can accommodate the storage capacity required for each use case.

For each use case, we perform the activities below:

- Step 1: Get the total number of deployed sensors count.
- Step 2: Calculate the total data points per day for each sensor.
- Step 3: Calculate the data points generated per block time.
- Step 4: Calculate the storage capacity required.
- Step 5: Determine the appropriate block time, assuming the blockchain can accommodate the required storage capacity.
- Step 6: Calculate the maximum data points stored in a block for each blockchain framework.

To compare various design choices of our experiments, we compare two blockchain frameworks (Hyperledger Fabric, Avalanche) using two use cases (IoT in agriculture, IoT in smart cities) (Table VI). Note that in certain applications, as discussed (see Table VI), such as within the healthcare domain, data might be collected at frequencies as low as one data point per second.

C. Limitations of the Model

Like any model, the proposed model has some limitations. This information would help to estimate the time to generate a new block.

1) Constraints of the design - our research mainly focuses on the given IoT layer structure; however, the layers could be vary in a real-world context.

2) Sampling errors could occur - data we use from previous studies may not reflect the general population or appropriate population concerned.

3) Simulation data - data for simulation are collected from different previous studies. Thus, when considering all data in a single flow, the results could deviate from the actual.

4) Measured timing could be changed on other dependencies (performance of hardware, software).
### TABLE VI
**COMPARING USE CASES: CASE STUDY 1 - IoT IN AGRICULTURE AND CASE STUDY 2 - IoT IN SMART CITIES**

<table>
<thead>
<tr>
<th>Description</th>
<th>Case Study 1 - IoT in Agriculture</th>
<th>Case Study 2 - IoT in Smart Cities</th>
</tr>
</thead>
<tbody>
<tr>
<td>IoT environment context</td>
<td>A farm utilizes IoT sensors to monitor the temperature, humidity, and soil moisture across different sections of the field. These sensors collect hourly data, allowing the farm to respond to environmental changes and optimize crop management. A blockchain network is implemented on the farm to ensure data security and integrity of the data.</td>
<td>A smart city uses IoT sensors to monitor traffic flow, air quality, and energy consumption across different areas of the city. These sensors collect data every two minutes, allowing the city to respond quickly to traffic patterns, pollution levels, and power usage changes. This real-time data collection helps the city efficiently manage resources and improve the quality of life for its residents. A blockchain network is deployed on the IoT smart city to enhance data security and maintain the integrity of the collected information.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IoT environmental parameters - sensors</th>
<th>Temperature, humidity, light intensity, and soil moisture</th>
<th>Traffic management, air quality monitoring, noise level, and energy consumption tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of sensors deployed</td>
<td>50</td>
<td>200</td>
</tr>
<tr>
<td>Data collection frequency</td>
<td>Every 60 mins (hourly)</td>
<td>Every 2 mins</td>
</tr>
<tr>
<td>Total data points per day for each sensor</td>
<td>24 hours/day × 1 data point/hour = 24 data points</td>
<td>24 hours/day × 30 data points/hour = 720 data points</td>
</tr>
<tr>
<td>Total data points collected by all sensors in a day</td>
<td>50 sensors × 24 data points = 1200 data points</td>
<td>200 sensors × 720 data points = 144,000 data points</td>
</tr>
<tr>
<td>Number of data points generated within the desired block time.</td>
<td>50 data points</td>
<td>200 data points</td>
</tr>
<tr>
<td>Storage capacity required per block time by assuming that each data point requires 1 KB of storage</td>
<td>50 data points × 1 KB = 50 KB</td>
<td>6000 data points × 1 KB = 6,000 KB</td>
</tr>
<tr>
<td>Calculate the maximum data points that can be stored in a block for Hyperledger Fabric (1 MB block size = 1024 KB)</td>
<td>1024 KB / 1 KB = 1024 data points</td>
<td>1024 KB / 1 KB = 1024 data points</td>
</tr>
<tr>
<td>Calculate the maximum data points that can be stored in a block for Avalanche (4 MB block size = 4096 KB)</td>
<td>4096 KB / 1 KB = 4096 data points</td>
<td>4096 KB / 1 KB = 4096 data points</td>
</tr>
<tr>
<td>New block generation time using Hyperledger Fabric (1 MB block size)</td>
<td>Data points generated per 60 minutes: 60 minutes × (50 data points / 1024 max data points) = 2.929 minutes (approx.)</td>
<td>Data points generated per 2 minutes: 2 minutes × (6,000 data points / 1024 max data points) = 1.71 minutes (approx.)</td>
</tr>
<tr>
<td>New block generation time using Avalanche (4 MB block size)</td>
<td>Data points generated per 60 minutes: 60 minutes × (50 data points / 4096 max data points) = 0.73 minutes (approx.)</td>
<td>Data points generated per 2 minutes: 2 minutes × (6,000 data points / 4096 max data points) = 0.296 minutes (approx.)</td>
</tr>
</tbody>
</table>

5) Proposed model is generated using the common functions of the process; however, those could be slightly changed based on the devices used (e.g., different sensors and hardware).

6) This study does not differentiate the communication approach devices use to transmit data (e.g., Bluetooth or WiFi). However, communication may be interfered with or depends on this in the real world.

7) The median block propagation time for all blockchain frameworks was not given; thus, we use a limited number of blockchains for the comparison.

VII. FUTURE DIRECTIONS

Current security technologies are unable to keep up with the exponential expansion in the number of Internet-connected devices and their CPU and memory constraints, resulting in vulnerabilities for hackers to exploit. IoT/Edge computing can increase the amount of resources and services available at the network's edge. However, distributed nodes pose a security concern. Despite its widespread application in various fields, AI remains a single source for building black boxes for consumers, limiting their trust in its outcome. In government and industry, federated databases are becoming more widespread. Optimizing federated-AI model adaptation...
for on-the-fly IoT/Edge data and leveraging Federated-AI for blockchain scalability and interoperability is an exciting potential future research avenue.

In the future, researchers could explore the feasibility and implications of executing all described operations on a local blockchain, where a single device carries out all processes. This exploration could open new avenues for efficient IoT implementations and further promote advancements in this direction.

In this work, quantum cryptography’s effects on the future of encryption were not examined, again, because of the unavailability of data. However, as with any other new technology, as data becomes available, it can be incorporated into our model.

The Digital twin is a representation of a physical asset, process, or service in a digital form. Directly modeling the physics of Edge/IoT networks is challenging. Automating the construction of the digital twin of Edge/IoT networks for cybersecurity monitoring and cyber threat modeling is an important future research direction. Furthermore, exploring how to combine different emerging technologies such as federated learning, artificial intelligence, and digital-twin technology with blockchain technology will be important to improve security in IoT networks.

Our model examines all functionalities in each IoT layer and presents a novel and comprehensive time estimation framework for understanding the influence of blockchain-enabled IoT on the block generation process and assessing block generation time in the IoT ecosystem. Further validation of our delay estimations of blockchain new block generation time in the IoT ecosystem. Further validation enabled IoT on the block generation process and assessing and presents a novel and comprehensive time estimation security in IoT networks.

This exploration could open new avenues for efficient IoT implementations and further promote advancements in this direction.

This study develops a new comprehensive-time computation model for block generation in blockchain-enabled IoT and provides insights that can lead to improved scalability of the architecture. It analyzes the main functions of four IoT layers (device layer, CH layer, fog/edge layer, and cloud layer) to adapt fast, real-time, machine-to-machine communications. The evidence from this study suggests that the block generation time in IoT networks mainly depends on the blockchain type, encryption algorithm, and the efficiency of each layer. The significant findings to emerge from this study are that the block generation time can be reduced by utilizing an efficient encryption algorithm (such as 256-bit ECC, elliptic curve cryptography) and the blockchain frameworks (such as Avalanche, Conflux, Algorand, Polkadot, and Hyperledger Fabric). Nevertheless, the blockchain framework does not play a significant role in block generation time for smaller data packets. Additionally, the results of this study indicate that utilizing the fog layer for the block validation process improves the efficiency of the entire block generation process. This study could be replicated using more experimental data (encryption time for different data packet sizes) to clarify the estimated impact using a real industrial IoT environment. By employing a suitable blockchain framework, encryption algorithm, and fog layer, unnecessary time consumption can be eliminated. During the design process, the proposed tool will assist in determining the most appropriate blockchain technology for a specific use case. However, it is recommended to conduct further studies on optimizing key functions of each IoT layer to improve blockchain scalability. In particular, such optimization can be achieved through utilizing Big data analysis, Artificial Intelligence, deep learning, machine learning, federated learning techniques, and the IoT service orchestration with the block generation process.

IX. APPENDIX

Here, we provide Table VII and Table VIII, which were referred to in Section II, where we provide a detailed comparison of different features of blockchain frameworks to select the most efficient blockchain type.

REFERENCES

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main Website</strong></td>
<td>bitcoin.org</td>
<td>litecoin.org</td>
<td>ethereum.org</td>
<td>hyperledger.org</td>
<td>neo.org</td>
<td>cardano.org</td>
<td>eos.io</td>
<td>algorand.com</td>
<td>conflux.org</td>
<td>binance.com</td>
<td>polkadot.network</td>
<td>avalancex.org</td>
</tr>
<tr>
<td><strong>Blockchain Generation</strong></td>
<td>1st Gen</td>
<td>1st Gen</td>
<td>2nd Gen</td>
<td>2nd Gen</td>
<td>1st Gen</td>
<td>2nd Gen</td>
<td>3rd Gen</td>
<td>4th Gen</td>
<td>3rd Gen</td>
<td>4th Gen</td>
<td>4th Gen</td>
<td>4th Gen</td>
</tr>
<tr>
<td><strong>Organisation / Developers</strong></td>
<td>Bitcoin Foundation</td>
<td>Lincoln Cons. Development Team</td>
<td>Ethereum Foundation</td>
<td>Linux Foundation (China)</td>
<td>Chain Foundation</td>
<td>IOHK</td>
<td>Extrasol (BlockOne)</td>
<td>Algorand, Inc.</td>
<td>Conflux Foundation</td>
<td>Binance Exchanges</td>
<td>W3C Foundation</td>
<td>Ava Labs</td>
</tr>
<tr>
<td><strong>Governance</strong></td>
<td>Bitcoin Foundation Developers Community</td>
<td>Lincoln Cons. Development Team</td>
<td>Ethereum Foundation (China)</td>
<td>Ethereum Foundation</td>
<td>EOSIO Foundation</td>
<td>Core Consensus</td>
<td>Core Consensus</td>
<td>Conflux Foundation</td>
<td>Binance Exchanges</td>
<td>W3C Foundation</td>
<td>Avalanche-X Council</td>
<td></td>
</tr>
<tr>
<td><strong>Blockchain Network Type</strong></td>
<td>Permissionless &amp; Public</td>
<td>Permissionless &amp; Public</td>
<td>Permissionless &amp; Public</td>
<td>Permissionless &amp; Private</td>
<td>Permissionless &amp; Public</td>
<td>Permissionless &amp; Public</td>
<td>Permissionless &amp; Public</td>
<td>Permissionless &amp; Public</td>
<td>Permissionless &amp; Public</td>
<td>Permissionless &amp; Public</td>
<td>Permissionless &amp; Public</td>
<td></td>
</tr>
<tr>
<td><strong>Platform Description</strong></td>
<td>Generic Blockchain platforms</td>
<td>Generic Blockchain platform</td>
<td>Generic Blockchain platform</td>
<td>Modular platform</td>
<td>Generic Blockchain platform</td>
<td>Modular platform</td>
<td>Modular platform</td>
<td>Generic Blockchain platforms</td>
<td>Modular platform</td>
<td>Modular platform</td>
<td>Modular platform</td>
<td></td>
</tr>
<tr>
<td><strong>Security (Hashing Function)</strong></td>
<td>SHA 256 (SHA-2 based)</td>
<td>Keccak, SHA 256</td>
<td>Ethash, KDF, CKB, ECSHA</td>
<td>AE2596, Neo, Jilong</td>
<td>Chaincode, SHA256 and RIPemd160</td>
<td>BLAKE2s-256</td>
<td>SHA-256</td>
<td>Ed25519</td>
<td>Keccak-256 (Cirrus-256, BLAKE2s)</td>
<td>SHA-256</td>
<td>BLAKE2s, SHA-256, SHA-3</td>
<td></td>
</tr>
<tr>
<td><strong>Programming Language</strong></td>
<td>C++, Go, C, +++, Rust, Solidity, EVM, LLVM</td>
<td>Go, C++, Rust, Solidity, EVM, LLVM</td>
<td>Java, COF</td>
<td>C++</td>
<td>C++</td>
<td>WASM</td>
<td>Reach, PyTAL, TEAL</td>
<td>Solality, Rust</td>
<td>Rust, Java, JavaScript, C++, C#, Python, and Solidity</td>
<td>Python, Javascript, Python, Vue</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Smart Contract</strong></td>
<td>No (possible via sidechains)</td>
<td>Progress (Pure Networks)</td>
<td>Yes (Solidity, Vyper)</td>
<td>Yes (Chainsicle, Golang, NodeJS, Java)</td>
<td>Yes (Chainsicle, Go)</td>
<td>Yes (Plutus)</td>
<td>Yes (WebAssembly)</td>
<td>Yes (TEAL)</td>
<td>Yes (Solidity, Vyper)</td>
<td>Yes (Rugbychain, Iranian Steams)</td>
<td>Yes (Solidity++, C-Chains)</td>
<td></td>
</tr>
<tr>
<td><strong>Smart Contract Applications Programming Languages</strong></td>
<td>No</td>
<td>Partial</td>
<td>Partial</td>
<td>Golang, NodeJS, Java</td>
<td>Chainsicle, Go</td>
<td>Haskell (based on Plutus)</td>
<td>Rust, C++</td>
<td>TEAL, PyTal</td>
<td>Solidity, Rust</td>
<td>Java, Javascript, C++, C#, Python, and Solidity</td>
<td>Python, Javascript, Python, Vue</td>
<td></td>
</tr>
<tr>
<td><strong>Smart Contract Execution</strong></td>
<td>Native</td>
<td>Native</td>
<td>EVM</td>
<td>Docker</td>
<td>NOM</td>
<td>Truffle</td>
<td>JVM</td>
<td>AVM</td>
<td>EVM</td>
<td>EVM</td>
<td>EVM, JVM, WASM</td>
<td></td>
</tr>
<tr>
<td><strong>Cross-chain Interoperability</strong></td>
<td>No</td>
<td>No</td>
<td>Yes (using parity)</td>
<td>No</td>
<td>No</td>
<td>Yes (using parity)</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td><strong>Currency - Native Token</strong></td>
<td>Bitcoin</td>
<td>Litecoin</td>
<td>Ethereum (ETH), tokens via smart contracts</td>
<td>Nano, currency and tokens via blockchain</td>
<td>NEO</td>
<td>ADA</td>
<td>EOS, tokens via smart contracts</td>
<td>ALEGO</td>
<td>CFX</td>
<td>BNB</td>
<td>DSR</td>
<td></td>
</tr>
<tr>
<td><strong>Transaction Fee</strong></td>
<td>0.1% (50% fee)</td>
<td>$4 (USD)</td>
<td>$40 (GAS)</td>
<td>No</td>
<td>GAS</td>
<td>0.16</td>
<td>0 (need bandwidth to validate)</td>
<td>ALEGO</td>
<td>GAS</td>
<td>SH01</td>
<td>No</td>
<td>GAS</td>
</tr>
<tr>
<td><strong>Mining/Block Reward</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Blocks are not created - validating peers</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Hashrate</strong></td>
<td>384.331 TH/s</td>
<td>327.348 TH/s</td>
<td>613.999 TH/s</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>7.168 TH/s</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Throughput / Scalability</strong></td>
<td>50 TPS</td>
<td>54 TPS</td>
<td>56 TPS</td>
<td>5.8k - 11k TPS (2-4 Gs)</td>
<td>100k TPS</td>
<td>250 TPS</td>
<td>100k TPS</td>
<td>1000 TPS</td>
<td>1000 TPS</td>
<td>Up to 4000 TPS</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Block Time (interval)</strong></td>
<td>10 min (600 sec)</td>
<td>2.5 min (150 sec)</td>
<td>12-14 sec</td>
<td>0.5 - 2 sec</td>
<td>15 sec</td>
<td>20 sec</td>
<td>0.5 sec</td>
<td>3.7 sec</td>
<td>1 sec</td>
<td>3 sec</td>
<td>7 sec</td>
<td>2-3 sec</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>---------</td>
<td>-------------------------</td>
<td>---------</td>
<td>-------------</td>
<td>---------</td>
<td>-------------</td>
<td>-------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Median block prop-</td>
<td>8.7 sec</td>
<td>1.02 sec</td>
<td>0.5 - 0.75 sec</td>
<td>0.075 sec</td>
<td>Not Found</td>
<td>5 sec</td>
<td>Not Found</td>
<td>Not Found</td>
<td>Not Found</td>
<td>Not Found</td>
<td>Not Found</td>
<td>Not Found</td>
</tr>
<tr>
<td>agation time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average Block Size</td>
<td>2 MB</td>
<td>1 MB</td>
<td>20-30 KB</td>
<td>1 MB</td>
<td>1-2 MB</td>
<td>500 KB</td>
<td>1 MB</td>
<td>N/A</td>
<td>Up to 2 MB</td>
<td>1-2 MB</td>
<td>3-4 MB</td>
<td>2 GB</td>
</tr>
<tr>
<td>Average Transactions per Day</td>
<td>20 KB</td>
<td>30 K</td>
<td>1 M</td>
<td>5 K</td>
<td>4K</td>
<td>4K</td>
<td>3.5 M</td>
<td>5M</td>
<td>5K</td>
<td>5M</td>
<td>15K</td>
<td>20M</td>
</tr>
<tr>
<td>Internet of things</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Decentralized Finance (DeFi)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Energy Consumption</td>
<td>Very High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>


Moshe Zukerman received a B.Sc. degree in industrial engineering and management, an M.Sc. degree in operations research from the Technion—Israel Institute of Technology, Haifa, Israel, and a Ph.D. degree in engineering from the University of California, Los Angeles, in 1985. He was an independent consultant with the IRI Corporation and a Postdoctoral Fellow with the University of California, Los Angeles, during 1985–1986. During 1986–1997, he was with Telstra Research Laboratories (TRL), first as a Research Engineer and, during 1988—1997, as a Project Leader. He also taught and supervised graduate students at Monash University during 1990–2001. During 1997–2008, he was with The University of Melbourne, Victoria, Australia. In 2008 he joined the City University of Hong Kong as a Chair Professor of Information Engineering, and a team leader. From December 2020 to September 2022, he also serves as the Acting Chief Information Officer of CityU. He has over 300 publications in scientific journals and conference proceedings. He has served on various editorial boards such as Computer Networks, the IEEE Communications Magazine, the IEEE Journal of Selected Areas in Communications, the IEEE/ACM Transactions on Networking, and Computer Communications.