IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 20, 2021, accepted July 13, 2021, date of publication July 26, 2021, date of current version August 5, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3100072*

# BFIM: Performance Measurement of a Blockchain Based Hierarchical Tree Layered Fog-IoT Microservice Architecture

**MD WHAIDUZZAMAN[1], (Senior Member, IEEE), MD. JULKAR NAYEEN MAHI[2], ALISTAIR BARROS[1], MD. IBRAHIM KHALIL[2], COLIN FIDGE[3], AND RAJKUMAR BUYYA[4], (Fellow, IEEE)**

[1]School of Information Systems, Queensland University of Technology, Brisbane, QLD 4000, Australia
[2]Department of Computer Science and Engineering, City University, Dhaka 1215, Bangladesh
[3]School of Computer Science, Queensland University of Technology, Brisbane, QLD 4000, Australia
[4]School of Computing and Information Systems, The University of Melbourne, Melbourne, VIC 3010, Australia

Corresponding author: Md Whaiduzzaman (wzaman@juniv.edu)

This work was supported in part by the Australian Research Council Discovery Project ''Re-Engineering Enterprise Systems for Microservices in the Cloud'' under Grant DP190100314.

**ABSTRACT** Fog computing complements cloud computing by removing several limitations, such as delays and network bandwidth. It emerged to support Internet of Things (IoT) applications wherein its computations and tasks are carried out at the network's edge. Heterogeneous IoT devices interact with different users throughout a network. However, data security is a crucial concern for IoT, fog and cloud network ecosystems. Since the number of anonymous users increases and new identity disclosures occur within the IoTs, it is becoming challenging to grow mesh networks to deliver end to end communications, as the extended IoT networks resemble a mesh architecture. To reinforce data security over IoTs, we deploy a microservice-based blockchain mechanism for fogs, which works as a decentralized client-server network medium (i.e., secured end device-based communication). We implement a blockchain equipped security scheme to be used with a fog-IoT hierarchical tree-based overlay mesh architecture to address and develop the network performance issues. In this study, we consider encryption and decryption delays from IoT and fog-integrated parts to monitor data records and compare them through the developed security scheme. The blocks of a blockchain offer the desired execution results mainly in terms of the algorithmic efficiency, which correlates with the existing algorithms, namely the Advanced Encryption Standard (AES), the Rivest Shamir Adleman (RSA), and the Data Encryption Standard (DES). Our 'BFIM' scheme has an enhanced task scheduling capacity and a more efficient throughput than the AES, DES, RSA resource deliverables (i.e., tasks). Our comprehensive performance evaluation implies that the Blockchain-based Fog IoT Microservice (i.e., BFIM) architecture provides a task delivery efficiency of 78.79% (i.e., task deliverable) and a service delivery efficiency of 83.24% (i.e., task scheduling). The 'BFIM' also has an overall process delivery efficiency of 75% (i.e., time delay, throughput) in the fog layer, rather than a central cloud layer running the AES, DES, and RSA algorithms.

**INDEX TERMS** Blockchain service, fog-IoT ecosystem, fog service delivery, hierarchical tree architecture, performance efficiency.

## I. INTRODUCTION

Cloud computing, a paradigm of on-demand delivery-based computing services, runs through the internet. However, it poses high latency and inefficient end-to-end data delivery in the cloud, concerns networks and allocated shared end devices. Fog computing is a decentralized computing infrastructure that shares resources or application processes within the cloud and harnesses potential user requests at the edge network. Fog computing reduces latency by allocating the workload to the right edge nodes as needed to distribute the network allocated data in a time based pairwise (e.g., timer scheduled packets or timed frames) manner. In this way, the edge network or end devices find their desired data

The associate editor coordinating the review of this manuscript and approving it for publication was Songwen Pei.

with a lower level of latency and share the in scheduled workloads within federated fog-cloud orchestrations [1], [2]. Due to the growing needs of mobile devices, IoT computing requires sharing of resources across embedded platforms. The increasing IoT network, distributing end-to-end delivery, finds mesh networks useful for dedicated short-range communication. However, the increasing number of users and anonymity are making it challenging to communicate with the right party across heterogeneous networks. Hence, the growing IoT interconnections employ an extra user load towards the gateway ends. When all processes are merged to be distributed via the IoT gateway, this procedure causes network jamming due to inconsistent data delivery at the network ends. Fogs can help these IoT gateways through a similar time-based priority mechanism in terms of data delivery [3], [4].

Blockchain is a distributed digital ledger technology that can be used for duplicated data transfer across a whole network (e.g., typically a mesh network). The operations or process interactions within a blockchain represent peer-to-peer based data sharing towards the block ends. A generic mesh network resembles a hierarchical tree overlay network, where the parent nodes act as primary responders for the child devices in a master–slave and slave–master pattern. However, the traditional mesh design employs different levels of cryptographic security to secure communication over the network. This procedure backlogs the system's back-end, as every front-end device needs to wait for the program to be executed and verified from the central device or primary master. In contrast, a blockchain has an irreversible logic pattern that can ensure better data security because there is no need to validate data records. However, the latency and exaggerated delay issue of blockchains around the increasing IoT network is a challenging task for end to end data delivery and secure communication transmission because of the frequent data offloading that occurs [5], [6]. Microservices can run several services in a single container and enrich distributed resource delivery for the network ends. This can help the blockchain activity to run through a single service scheduler to minimize the extensive delay issue and latency around the end-to-end hops. Deploying the blockchain activity over microservices can help to scale the overlay network through faster data delivery and minimized gathering of delay latencies around the network [7].

Traditional mesh networks typically share data within a peer-to-peer interconnection [8]. As a result, mesh networks often encounter contention during two-way verified communication and experience latency-based delays at hop-to-hop or gateway ends. The mesh network encounters data offloads and causes latency-based delays while verifying the end-to-end device interactions. The traditional and sophisticated (e.g., sensor networks) IoTs share and distribute data through mesh networks and cause delays (e.g., latencies towards the end to end communication) when there is an increased number of anonymous users. The generic fog allocations usually distribute data sequences in a schedule-based manner through dividing priority workloads. However, increasing user loads and numbers of end devices generate different priority workloads for end-to-end data deliveries, and the fog networks tend to have difficulty making decisions [9]. Hence, IoTs need to share the process schedulers (i.e., packet scheduling) of the fog end for data allocation procedures and increase the delay, producing data stress among the distributed fog data slots. The use of a valid developed scheme through proper scheduling or resource allocation may tune the fog services well, allowing integrated fog–IoT orchestrations [10].

To minimize the anonymity security issue over hop-to-hop mesh interconnections at the IoT end to end deliveries, we developed a blockchain-based security scheme for hierarchical-tree-based fog–IoT mesh architectures. A star-mesh network was designed to connect the primary controller fog with a cloud at the back end of the system delivery and IoT gateways at the front end. In this network, every fog node was designed according to its respective IoT gateway properties and data schedulers were configured according to IoT gateway end device interactions. The fogs can handle the extra data stress from the IoT device ends and quickly perform scheduled priorities according to the needed allocated task. We also configured preemptive schedulers and the fog for performing data distribution and verification towards the cloud backend. The increasing number of IoT devices introduces a question around the security issue of ensuring anonymity. To reduce this problem, we configured a blockchain-based microservice property to lessen the blockchain's delay issue and enhance its security along with that of the defined network. The microservice specifically works for attacked blocks in this developed procedure. Information is stored and distributed after being verified from the fog layer backend in IoT layer communication. The configured microservice property enhances the schedulability and scalability of fogs and IoT layer activities. The scheduled microservice minimizes the blockchain's extensive delay and offloads data during the attacking procedure and enhances information scheduling activity (e.g., less network jamming) during the fog–IoT orchestration.

If any intruder wishes to enter through the IoT cluster, the fog gathers the information from the IoT. The fog cluster sends the information to the cloud for verification and validation while stopping the blockchain schedulers. Our developed security scheme is three-layer tamper-proof resistant. The intruder is unaware of the IoT, fog, and cloud cluster identifies. If anyone tries to enter without an identity, they will be banned from the cloud, and the fog cluster will break the running block schedulers. Since our configured scheme breaks apart the blockchain's communication process, it is almost impossible to work through a man-in-the-middle attack within the network if any intruder counterfeiting occurs. Besides this, the PoW (proof of work) IP is tracked by the cloud verification process each time, causing the user to be terminated. An eavesdropping attack is not possible either because of the three-layer verification and validation process of this scheme over IoT, fog, and cloud clusters.

In our approach, the blockchain works only in the fog and IoT layer. The chain mainly works as a carrier to secure data processing over the fog and IoT network. The microservice on the blockchain's fog layer works only if an attack is encountered by an intruder. Our configured microservice-based blockchain security mechanism works with PoW (i.e., Proof of Work), IP targeting, and hashing power (i.e., mining capacities through graphics parser unit, GPU and Point over Ethernet, PoE) computation of the authorized users. For unauthorized access, the hashing power and IPs are tracked and calculated through the cloud reservoir. The connection could be terminated to prevent exaggerating delay issues of the whole network's scheduled processes for effective blockchain tuned services.

AES, DES, and RSA cryptosystems are established and widely used data security schemes. However, the main issue with these protocols is to do with the in hungry delay mechanism. The blockchain itself creates additional delays while circulating over the network nodes. Thus, the establishment of a minimized delay tolerance or low latency-based security scheme, defined in Fig.1 is still needed for overlay networks and the blockchain hierarchy. Fog redistributes the data or packet allocation to the IoT cluster and keeps the cloud free to track verified or unverified (e.g., intruder) interactions. A microservice-based containerized scheduler can enable intruder interpretation to ensure cloud verification and keep fog processes in the workflow through the configured scheduling process. The developed security approach ensures delay minimization within the fog–IoT orchestration and releases the cloud from verification-based delay processing loads.

In this paper, we use three popular cryptography algorithms—AES, DES, and RSA—to conduct a performance estimation analysis in the context of a blockchain-based security scheme. We compared the existing cryptographic algorithms with our blockchain-based fog IoT Microservice (BFIM) scheme to check its efficacy across the fog–IoT orchestration environment. We contrived iterative building blocks to act as a continuous information run-time for our blockchain carrier over the mesh network or IoT nodes. In this scheme, the building blocks are generated iteratively when needed to pass or process information but not overcome the One Time Session Password (OTSP) session (e.g., session-based delay for layer verification and validation) from fog–IoT and fog–cloud. The growing mesh network in the IoT layer represents a blockchain analogy, where the devices act as intermediate nodes, and their interactions may represent the interchange of blocks within a blockchain. In the fog environment, attacked block information is stored in a microservice, delivering the attacked block information with the processed blocks from the IoT layer towards the central cloud.

**Research contributions:** The main contributions of our research work include:

- We propose an efficient blockchain-based fog–IoT data security scheme to mitigate against existing challenges in a distributed data security framework.
- We implement the data security scheme for a hierarchical tree-based overlay mesh architecture in the cloud–fog and IoT ecosystem.
- We conduct a performance analysis of our scheme and architecture with widely used cryptographic schemes, namely AES, DES, and RSA. A block generation time delay analysis (e.g., system run-time tolerance or service load) is carried out to analyze the architecture's performance.

The remainder of this paper is structured as follows– **Section 2** represents a brief overview of the literature and related work to identify the current obstacles, solutions, and limitations. **Section 3** illustrates the methodology, including the use of a secure blockchain property, its block numbers, the component work method for fog and cloud distribution, and how these factors were used in the data security scheme. **Section 4** discusses our developed blockchain security scheme-based secure architecture. **Section 5** describes our developed algorithm and its working process in the fog cloud and central cloud. An in-depth discussion of our simulated results is described in **Section 6**. A rigorous discussion about our workflow process, its problems, and future developments is given in **Section 7**. Finally, **Section 8** concludes the paper by illustrating future outcomes and beneficial results. Essential abbreviations are shown in Table 1.

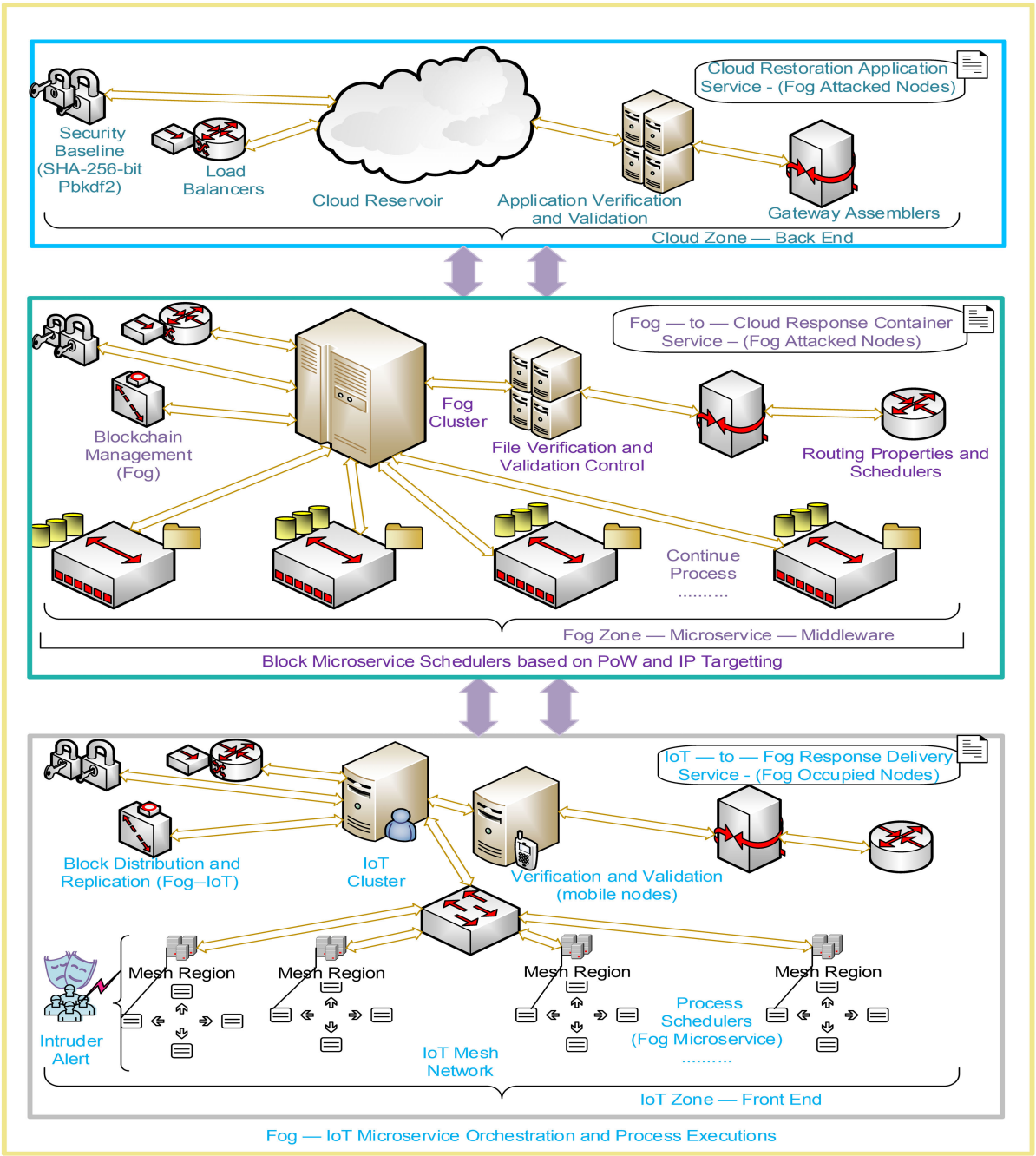
**TABLE 1. Essential abbreviations.**

| Short Form | Full Form or definition |
|---|---|
| *AES* | Advanced Encryption Standard |
| *DES* | Data Encryption Standard |
| *RSA* | Rivest Shamir Adleman Cryptosystem |
| *IoT* | Internet of Things |
| *OTSP* | One Time Session Password |
| *pbkdf2* | Password-Based Key Derivation Function 2 |
| *SHA-256* | Secure Hash Algorithm 256 bit |
| *CPU* | Central Processing Unit |
| *CP-ABE* | Cipher Text Policy Attribute Based Encryption |
| *BFIM* | Blockchain Fog-IoT Microservice |

## II. BACKGROUND AND MORPHOLOGICAL STUDY

Security is a vital concern for distributed heterogeneous networks. The blockchain in the fog–IoT orchestration may turn and tune the interaction in a secured motion to provide better service provisioning over fog–cloud data flow while ensuring security. The cloud performs on a better scale while interacting with a fog-integrated blockchain approach (Table 2.).

Vishwanath *et al.* [11] described a method that can be used to overcome the security issues related to the use of AES in fog-computing. In their process, the AES algorithm was shown to support better security privileges [12]. Their paper's primary outcome was to compare security issues while using data sets of different sizes [13], [14]. This was primarily done to verify the CPU time, encryption time, decryption time, and consumed memory sizes of algorithms. However, in our case, the data size was not fixed (e.g., random changes over the user's interaction) and differed from anonymous user




Legend:

| Layer | --- | Color (RGB) |
|---|---|---|
| Cloud | --- | Deep sky blue |
| Fog | --- | Light sea green |
| IoT | --- | Silver |
| BFIM | --- | Khaki |

**FIGURE 1.** The Fog–IoT hierarchical mesh architecture in the BFIM scheme.

exchange throughout the network. Zenil *et al.* [15] conducted a complete survey on the concepts and methods involved in classical and algorithmic information theory. They also described the limiting stages of working data parameters while using Shannon entropy and a lossless compression contrast algorithm [16] in their development. In contrast to their work, our created algorithm is not related to a specific parameter's threshold entropy.

**TABLE 2. A brief comparison of our research work with existing works.**

| Research Article | Network illustrations and services | Methodology | Research gap | Developed approach |
|---|---|---|---|---|
| Vishwanath et al. [11] | Increased fog security through AES. | AES modular security base within the fog-cloud. | Derived the efficiency through AES in fog only. | Verified CPU delay time and memory consumption through AES, DES and RSA. |
| Dehmer and Pickl [17] | Centralized structural network topology. | Discrimination aware based routing pattern. | Sharing response time is not defined over network. | Token wise packet delivery mechanism for response time validation. |
| Lin and Liao [22] | Blockchain apps for IoT's and financial markets. | Decentralized immutable architecture. | Blockchain architecture developed for overlay networks. | Cloud-watch facility for system run-time enhancement. |
| Khan et al. [30] | Fog-IoT architecture development review. | Pattern-wise activity to find fog-IoT security gaps. | In-depth survey. | Scheduled session basis block breaking mechanism. |
| Ronghua Xu et al. [51] | SOA microservice for IoT-CPS. | Fog-IoT microservices deployment. | Increased delay based fog-IoT microservices. | Delay minimized mechanism in Fog only microservice. |

Dehmer and Pickl [17] used a quantitative graph analysis and a sequence of data [18] indexes. They analyzed the graph through their developed measurements to view the structured pattern of networks [19]. To analyze information over the graph [20], they used discrimination aware based theoretic statistical methods [21]. In contrast, our algorithm depends on the token-based packet delay procedure (e.g., blockchain), adding to the run time complexity of the order of growth (e.g., algorithm) for the network architecture (e.g., partial mesh design). Lin and Liao [22] completed a top-down survey of blockchain-based applicable sectors as well as applications. They suggested that blockchains may be used in the financial market, IoT, Supply chain, Voting, Medical treatments, and Storage. They also triggered the blockchain's main key element, which is to follow a decentralized [23], transparent, open-source, autonomous, and immutable environment.

Yi *et al.* [24] conducted a complete review of past research work on security and privacy issues across conjoint networks in their paper. They considered the security issues to be data storage, computation security [25], and network security. Some concerning privacy issues [26]–[28] in their work were data privacy [29], usage privacy, and location privacy.

Khan *et al.* [30] derived different security scenarios or issues in fog computation. Fog security concerns primarily depend on the use of the IoT or other coupled devices within a layer-based network [31] to provide efficient computation. Their main role is to identify the security [32] gaps among past works on fog computing. Stanciu [33] discussed the effectiveness of a centralized blockchain computing model and infrastructure in the public cloud. They used a distributed control system for data processing [34] that considers low latency, location awareness, geographically distributed packets, mobility, the predominate role of wireless access, and heterogeneity [35] within the cloud interactions.

Mukherjee *et al.* [36] expanded the fundamental problems of cloud counterfeiting such as end to end delay, traffic congestion, data traffic, and communication [37] costs. They also described the challenges related to security and privacy countermeasures through reviewing research articles about the cloud, fog, [38] and edge computing paradigms [39]. Alrawais *et al.* [40] deployed an effective key exchange security protocol, termed the Cipher Text-Policy Attribute-Based Encryption (CP-ABE), in their work. However, in some cases, issues with fog computing arises [41] while checking the security [42] in terms of data alteration, unauthorized access, and the identification of attacks [43]. The developed protocol CP-ABE is better in terms of verifying the un-authorization of data [44] and eavesdropping attacks. In contrast, our constructed architecture successfully eliminates man in the middle and eavesdropping attacks within the fog–IoT [45]–[48] block-chained [49]–[50] orchestration hierarchy. Ronghua Xu *et al.* [51] employed an SOA (e.g., service oriented) blockchain-based decentralized microservice architecture for IoT cyber-physical systems. In their work, the permissible containerized microservices were deployed to entire fog and IoT clusters to ensure the utmost level of public safety and security with no cloud back end. Haipeng Yao *et al.* [52] represented a multi-agent reinforcement learning algorithm for near-optimal policy searching to find out the Nash equilibrium state. Their developed blockchain platform assisted in offloading of computational tasks across the industrial internet of things (i.e., IIoT) networks to allocate resource managements at the cloud back end. However, in our developed work, we considered the microservice only for the fog cluster [53], specifically in the blockchain attack scenario, to overcome the latency or scalability issues and release the extra load from the cloud's back end. A reservoir cloud is employed in our developed environment to keep a track record about intruder alerts and new blockchain circulation within the fog–IoT overlay network.

## III. SYSTEM MODEL AND METHODOLOGY

In this section, we describe our system model and methodology.

### A. BLOCKCHAIN DATA PROCESSING IN FOG, CLOUD AND IoT NETWORK LAYER

We developed a secure blockchain mechanism to increase IoT device security. The employed scheme enhances the security pattern in fog networks while performing distributed

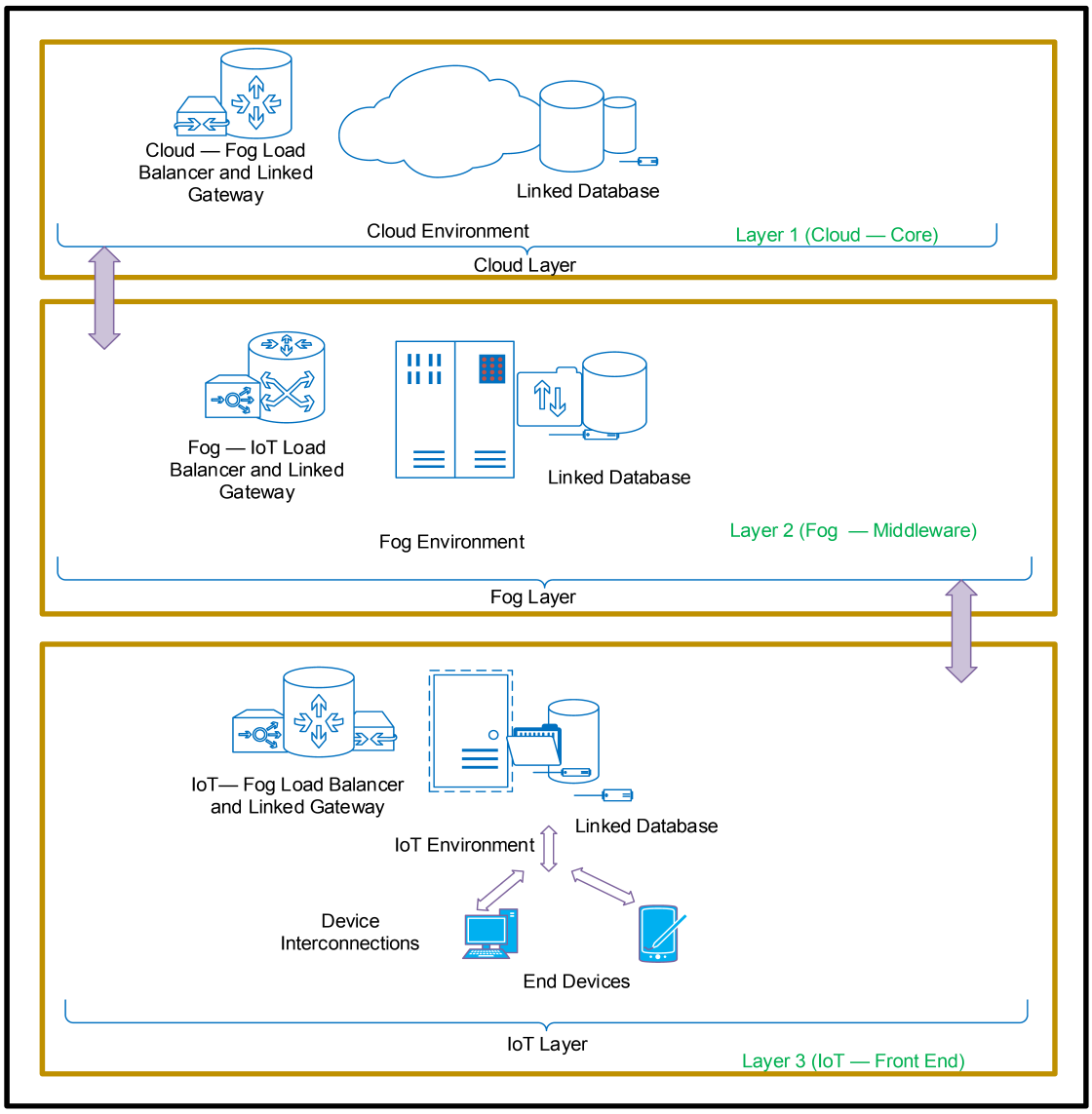



**FIGURE 2.** The generic architecture for Fog-IoT orchestration process.

data sharing activities. The layered architecture is categorized under three segments—cloud, fog, and IoT layer (e.g., end devices). The three-layer security architecture works in a bottom-up fashion, shown in Fig. 2. IoT network layers are the vulnerable or concerned sector that is usually aggrieved by various types of attack. We developed a secure fog layer within the fog–IoT orchestration to minimize these security thefts or attacks. Here, the fog layer acts as a primary helper for the IoT and a secondary helper towards the cloud. The main motive of this development is to overcome the possible security issues of mesh networks over the IoTs by providing the utmost system run-time enhancement or service provisioning. The fog has the carrier information (e.g., a container of individual attacked block information or microservice) for the vulnerable (e.g., commuted attacks) IoTs. If an attack is found in the IoT layer, the fog carrier cuts off the fog–IoT gateway connection and sends the information to the cloud. The fog layer has a secured (e.g., SHA-256 pbkdf2) timestamp and will shut off if an attack occurs behind its backbone. The fog generates block numbers one by one while carrying the blockchain's information after being overridden by the timestamp. Later on, the fog layer distributes the information towards the central cloud layer that contains the information of the pre-attack processes. Thus, the system's run-time improves with better provisioning.

### B. COMPONENTS OF THE HIERARCHICAL Fog–IoT MESH TOPOLOGY AND DERIVED ARCHITECTURE

Our system architecture is divided into three conventional layers: the back-end, middleware, and front-end services. The three-layer architecture may be categorized as 1) the cloud layer or core (back-end), 2) the fog layer (middleware), and 3) the IoT layer (front-end).

• ***The core or cloud Layer:*** The central cloud or service preserves the session's data that have been attacked before. All of the information before being attacked is delivered from the fog towards the cloud in a secure fashion through the OTSP (e.g., One-Time Session Password) SHA-Pbkdf2 scheme. The cloud sends a cryptographic authentication pattern (e.g., verified and validated OTSP) to the fog if an attack has been made (refer to Algorithm 1.). The current fog sessions are then turned off to give an initialization delay (e.g., session-based configured counter for patrolling). The fog sessions resume through re-scaling formatting of the previous information (e.g., before being attacked) and monitoring the current data with the previous data with a mixed synchronization pattern of blocks (e.g., new and attacked ones). A data center resides in the cloud that acts as a reservoir and contains specifically attacked fog node (e.g., a microservice container for affected fogs) information. Later, the reservoir sends a code of conduct to the cloud to cause a re-scheduling of synchronized blocks (e.g., those attacked from fog or the IoT, a newly created block from the fog or IoT, and the interacted block from the fog or IoT) through the verification of session-based timing. Each renewed block includes information on the attacked fog or IoT nodes and the new nodes of fog or IoT that are transferred through the cloud's fog layer. Thus, the developed secure model re-distributes the synchronized information with no loss of data but with a smaller time delay on average.

• ***The fog Layer or middle-ware:*** The fog layer is equipped with a blockchain pattern that acts as middleware for the cloud and IoT. The fog layer establishes a blockchain pattern to synchronize the information. The fog gathers direction from the cloud and verifies the re-scheduling of blocks (e.g., newly created ones in fog and validated ones from the IoT layer) to get the most possible information before carrying out the attack. The information-gathering procedure is done regularly after intruder attack (refer to Algorithm 2.). The fog layer is comprised of a timestamp, namely the OTSP (e.g., one-time session password), which is generated in each session of blocks after time. Furthermore, if an attack occurs, the fog regenerates the OTSP by sending the information back to the cloud. After that, a re-synchronization of the blocks is made through the re-initialization of delays. Thus, the re-scheduling of validated, attacked, and newly created blocks is synchronized, supporting the avoidance of information loss after being attacked. The fog is considered to be a standalone backbone for the developed system model, whereas the cloud is kept in hidden form. The fog cluster and the fog node (i.e., fog layer) configurations or system properties are same within their respected gateways and individual operators. The fog layer also has a similar security pattern to the cloud that uses an additional OTSP scheme to provide extra security for the blockchain.

• ***The IoT Layer or front-end service:*** The IoT layer acts as a front-end service for the underlying fog and cloud layer. The IoT layer has permission to connect with individual devices across the network that need verification from the fog–IoT blockchain service. In the IoT layer, only the cluster node properties are fixed and aligned to their respective gateway entities but the device configurations changes as per user experience or behavior. As a result, the IoT layer is vulnerable to attacks and attacks may be transferred from other devices

**Algorithm 1** Regular Blockchain Mining From Cloud to Fog and Fog Microservice-End to IoT (Both Attacked and Not attacked)

**Input**: SHA-256-bit ascending block generation from 1 to n

**Output**: Identifying not-attacked blocks and execute blockchain schedulers

```
/* isAttacked = A boolean value defines whether it is an Attack scenario or Not Attack scenario. */
/* blockList = SHA-256-bit ascending block list */
/* block = A particular block from the blockList */
foreach block in blockList do
    Step 1: Actor: Cloud
        attackInfo ← block.getAttackInfo()
        if block is NOT from valid network AND attackInfo verification is failed then
            isAttacked ← true
            n ← blockList.size()
            block.number ← n + 1
            Send the block To FogMicroservice
        else
            isAttacked ← false
            check previous attack info of the bock
            Assign Block Scheduling Process to FogMicroservice
    if isAttacked then
        Step 2: Actor: FogMicroservice
            Regenerate The Attacked Block
            n_c ← current executing block number
            traverseTo(n_c)
            block.number ← n_c+1
            Increment Other Block Numbers By One
    else
        Step 2: Actor: FogMicroservice
            Assign New Blocks To IoT Cluster
        Step 3: Actor: IoTClusterNode
            if Alpha-Neumeric Password is Verified then
                Continue The Cloud-Fog Verified Scheme
                Get Block Numbers From Scheduler
                Generate And Process Blocks
```

**Algorithm 2** Regular Blockchain Mining From Fog-Microservice to IoT-End and IoT-End to Fog-Microservice While Regenerating Blockchain Mining (Both Attacked and Not attacked)

**Input**: SHA-256-bit ascending block generation from 1 to n
**Output**: Identifying not-attacked blocks and execute blockchain schedulers

```
/* isAttacked = A boolean value defines whether it is an Attack scenario or Not Attack scenario. */
/* sysExec = The system is executing without any interruption. */
while sysExec do
    Step 1: Actor Cloud
        attackInfo ← block.getAttackInfo()
        if block is NOT from valid network AND attackInfo verification is failed then
            isAttacked ← true
            n ← blockList.size()
            block.number ← n + 1
            Send the block To FogMicroservice
        else
            isAttacked ← false
    if isAttacked then
        Step 2: Actor FogMicroservice
            Get Verified Block Data From Cloud
            Get Attacked Cluster Data From IoT
            Get Possible Attack Info if session is NOT expired then
                Create New Block Thread
                Send blocks To IoT
        Step 3: Actor Cloud
            Renew Session Data
    else
    Step 2: Actor FogMicroservice
        Assign New Blocks To IoT Cluster
    Step 3: Actor IoTClusterNode
        if Alpha-Neumeric Password is Verified then
            Continue The Cloud-Fog Verified Scheme
            Get Block Numbers From Scheduler
            Generate And Process Blocks
```

run by eavesdroppers (refer to Algorithm 3.). The expanded system design is followed by such an authentication pattern, so the fog–IoT layer may connect with different devices within the network; however, it proceeds with information only after initializing the session from the fog. On the other hand, for forgery, man-in-the-middle, or DoS attacks, the system pattern has an iterative OTSP (e.g., one-time session password) scheme. The OTSP scheme specifically works to secure information over the fog, saving the cloud from being traced or attacked.

**Algorithm 3** Fog-Microservice to IoT-End and IoT-End to Cloud-End via Fog-Microservice While Regenerating Blockchain Mining (Both Attacked and Not attacked)

**Input**: SHA-256-bit ascending block generation from 1 to n
**Output**: Identifying not-attacked blocks and execute blockchain schedulers

```
/* isAttacked = A boolean value defines whether it is an Attack scenario or Not Attack scenario. */
/* sysExec = The system is executing without any interruption. */
while sysExec do
    Step 1: Actor Cloud
        attackInfo ← block.getAttackInfo()
        if block is NOT from valid network AND attackInfo verification is failed then
            isAttacked ← true
            n ← blockList.size()
            block.number ← n + 1
            Send the block To FogMicroservice
        else
            isAttacked ← false
    if isAttacked then
        Step 2: Actor Fog Microservice
            if session is NOT expired then
                Create New Block Thread
                Assign New Blocks To IoT Cluster
            Get Data From Attacked Session
            Get Data From IoT Cluster
        Step 3: Actor IoT Cluster Node
            if Alpha-Neumeric Password is Verified then
                Continue The Cloud-Fog Verified Scheme
                Get Block Numbers From Cloud Scheduler
                Generate And Process Cloud Blocks
        Step 4: Actor Cloud
            if IoT-Cluster Node is Verified then
                Get Block Numbers From Scheduler
                Generate And Process Blocks
            Renew Session Data
    else
        Step 2: Actor FogMicroservice
            Create New Blocks
            Send New Blocks To Cloud
            Reallocate Fog IoT Block Scheduler
            Send New Scheduler To Cloud
```

### C. COMPLEXITIES OF OUR DERIVED ALGORITHMS

For every algorithm the complexities are mainly of two types; 1) Time Complexity and 2) Space complexity. However, the complexity details are depicted below for the **Algorithm 1.**, **Algorithm 2.** and **Algorithm 3.** respectively.

#### 1) COMPLEXITY OF ALGORITHM 1

For *isAttacked*, *blockList* and *block* independent variables, the *Cloud* executes *attackInfo* verification and validation,

*blockList*, and *block* number for a specific process or task allocation. However, each of these operation executes according to their respective scheduling properties of running services within *Cloud*, *Fog* and *IoT* orchestrations. The services contains task generation, task deliveries task scheduling, resource allocation and process delivery properties. Moreover, the algorithm is divided by several allocation of processes namely as Actors. The potential actors are defined as - *Cloud*, *FogMicroservice*, *IoTClusterNode*. We configured stack based process delivery option for our algorithm. Whereas, the process deliveries are generated by push and pop features of the allocated time slots. The priority listings are occupied according to preemption based ascending orders in priority en queue list. The overflow of processes delivered towards the *FogMicroservice* to execute the de queue option. The de queue feed-forwards the overflow processes to the microservice container occupied by empty queue list. The overall Blockchained secured property (i.e., BFIM) supports the respective en queue and de queue lists.

Considering the process operations, the space complexity of each operation relies over the order of $\mathcal{O}(n)$. On the other hand, the time complexity relies over the order of $\mathcal{O}(n \log n)$.

Therefore, The time complexity for (Algorithm 1.) derives over the order of

$$\begin{aligned} f(n) &\rightarrow 2n^3 + 4n^2 + 2n \\ \mathcal{O}(Algorithm\ 1.) &\rightarrow \mathcal{O}\ [2(n^3 + 2n^2 + n)] \\ &\rightarrow \mathcal{O}\ (n \log n) \end{aligned}$$

Therefore, the time complexity is

$$T(n) = \mathcal{O}\ (n \log n)$$

The space complexity considers seven instances. Whereas, each actors are independent for each actions and true for all properties and perspectives (see Algorithm 1.). $Cloud = 1$, $Fog = 1$, $IoT = 1$, $blockList = \text{n}$, $block = \text{n}$, $isAttacked = 1$, $notAttacked = 1$.

Therefore, the space complexity is

$$S(n) = \mathcal{O}\ (n)$$

### 2) COMPLEXITY OF ALGORITHM 2

For *isAttacked*, *blockList* and *block* independent variables, the *Cloud* executes *attackInfo* verification and validation, *blockList*, and *block* number for a specific process or task allocation. However, each of these operation executes according to their respective scheduling properties of running services within *Cloud*, *Fog* and *IoT* orchestrations. The services contains task generation, task deliveries task scheduling, resource allocation and process delivery properties. Moreover, the algorithm is divided by several allocation of processes namely as Actors. The potential actors are defined as - *Cloud*, *FogMicroservice*, *IoTClusterNode*. However, in the phase transition of (Algorithm 2.) *FogMicroservice*, *Cloud* has the property to execute double times for different instances within a same process scheduler. We configured stack based process delivery option for our algorithm. Whereas, the process deliveries are generated by push and pop features of the allocated time slots. The priority listings are occupied according to preemption based ascending orders in priority en queue list. The overflow of processes delivered towards the *FogMicroservice* to execute the de queue option and to the *Cloud* for executing the en queue option. The de queue feed-forwards the overflow processes to the microservice container occupied by empty queue list. The overall Blockchained secured property (i.e., BFIM) supports the respective en queue and de queue lists.

Considering the process operations, the space complexity of each operation relies over the order of $\mathcal{O}(n)$. On the other hand, the time complexity relies over the order of $\mathcal{O}(n \log n)$.

Therefore, The time complexity for (Algorithm 2.) derives over the order of

$$\begin{aligned} \mathcal{O}(Algorithm\ 2:) &\rightarrow f(n) \rightarrow 3n^4 + 11n^3 + 9n^2 + n \\ \mathcal{O}(Algorithm\ 2:) &\rightarrow \mathcal{O}\ [n(3n^3 + 11n^2 + 9n + 1)] \\ &\rightarrow \mathcal{O}\ (n \log n) \end{aligned}$$

Therefore, the time complexity is

$$T(n) = \mathcal{O}\ (n \log n)$$

The space complexity considers seven instances. Whereas, each actors are independent for each actions and true for all properties and perspectives (see Algorithm 2.). $Cloud = 1$, $Fog = 1$, $IoT = 1$, $blockList = \text{n}$, $block = \text{n}$, $isAttacked = 1$, $notAttacked = 1$.

Therefore, the space complexity is

$$S(n) = \mathcal{O}\ (n)$$

### 3) COMPLEXITY OF ALGORITHM 3

For *isAttacked*, *blockList* and *block* independent variables, the *Cloud* executes *attackInfo* verification and validation, *blockList*, and *block* number for a specific process or task allocation. However, each of these operation executes according to their respective scheduling properties of running services within *Cloud*, *Fog* and *IoT* orchestrations. The services contains task generation, task deliveries task scheduling, resource allocation and process delivery properties. Moreover, the algorithm is divided by several allocation of processes namely as Actors. The potential actors are defined as - *Cloud*, *FogMicroservice*, *IoTClusterNode*. However, in the phase transition of (Algorithm 3.) *Cloud*, *FogMicroservice* has the property to execute double times for different instances within a same process scheduler and respective order of process accumulation to be scheduled for the next run. We configured stack based process delivery option for our algorithm. Whereas, the process deliveries are generated by push and pop features of the allocated time slots. The priority listings are occupied according to preemption based ascending orders in priority en queue list. The overflow of processes delivered towards the *FogMicroservice* to execute the de queue option and to the *Cloud* for executing the en queue option. The de queue feed-forwards the overflow

processes to the microservice container occupied by empty queue list. The overall Blockchained secured property (i.e., BFIM) supports the respective en queue and de queue lists.

Considering the process operations, the space complexity of each operation relies over the order of $\mathcal{O}(n)$. On the other hand, the time complexity relies over the order of $\mathcal{O}(n \log n)$.

Therefore, The time complexity for (Algorithm 3.) derives over the order of

$$\begin{aligned}\mathcal{O}(Algorithm\ 3:) &\rightarrow f(n) \rightarrow n^8 + 3n^7 + 7n^6 + 8n^5 \\ &\quad + 4n^4 + 8n^3 + n^2 + n \\ \mathcal{O}(Algorithm\ 3:) &\rightarrow \mathcal{O}\,[n(n^7 + 3n^6 + 7n^5 + 8n^4 \\ &\quad + 4n^3 + 8n^2 + n + 1)] \\ &\rightarrow \mathcal{O}\,(n \log n)\end{aligned}$$

Therefore, the time complexity is

$$T(n) = \mathcal{O}\,(n \log n)$$

The space complexity considers seven instances. Whereas, each actors are independent for each actions and true for all properties and perspectives (see Algorithm 3.). $Cloud = 2$, $Fog = 2$, $IoT = 1$, $blockList =$ n, $block =$ n, $isAttacked = 1$, $notAttacked = 1$.

Therefore, the space complexity is

$$S(n) = \mathcal{O}\,(n)$$

### D. HIERARCHICAL-TREE-BASED BLOCKCHAIN MATHEMATICAL MODEL FOR IoT, FOG, AND CLOUD MESH INTERACTIONS

Plain text is an original readable format message that acts as an input for the encryption algorithm. Ciphertext is an unreadable message format output of the encryption algorithm. After generating the encryption algorithm, the ciphertext is converted to plain text through a decryption algorithm. The system model's established algorithms use plain text and session-wise secret keys as inputs and generate ciphertext as an executed output. The Decryption process at the cloud end converts ciphertext into plain text. The cloud stores the desired information through verification from the fog layer, and the mathematical equations are formulated by checking the verified parameters. The equations are described below as a part of the workflow for the whole procedure. The equations are generated considering the cloud, fog, and IoT interactions, primarily based on attacked and not attacked.

### E. CRYPTOGRAPHIC KEY EXCHANGE PATTERN AND MATHEMATICAL DERIVATION OF PROCESSES FOR IoT, FOG AND CLOUD HIERARCHY

#### 1) THE CRYPTOGRAPHIC KEY EXCHANGE PATTERN FOR THE DEVELOPED EDGE, FOG AND CLOUD HIERARCHY IS AS FOLLOWS-

- The Cloud private key encryption process is carried through getting the IP information from fog, run-time session id of cloud and SHA-256-bit Pbkdf2 key generation to secure the data packets of cloud.
- The Fog public key decryption process is carried through the previous Cloud run-time Encryption processes.
- The fog private key encryption process is carried through getting the IP information from the IoT parent device, the run-time session id of fog and SHA-256-bit Pbkdf2 key generation to secure the data packets of fog.
- The IoT public key decryption process is carried through the previous fog run-time encryption processes.
- IoT private key encryption process is carried through getting the IP information from the IoT child nodes, the run-time session id of IoT and SHA-256-bit Pbkdf2 key generation to secure the data packets of IoT.
- The IoT extended nodes public key decryption process is carried through the previous IoT child node run-time encryption processes.

### F. DERIVED MATHEMATICAL EQUATIONS FOR THE CONSTRUCTED PROCESSES

The following equations mainly illustrate the metrics of the process generation time, computation rate, packet interchange rate, packet delivery rate within the cloud, fog, and IoT orchestrations. These equations also lead to defines algorithms' performance differences within the results—specifically among the AES, DES, RSA, and the newly developed scheme BFIM.

#### 1) CLOUD IN ATTACKED SITUATION

$R_f$ = Cloud Current Computation Rate + Fog Packet; $R_d$ = Packet Interchange Rate (Current Delivery Delay) + Fog Packet (insertion)

$$\int \left(\int_0^{\infty} \sum_{i=1}^{n} R_c\right) dt = \int \left(\int_0^{\infty} \sum_{i=1}^{n} C_{cc} + \int_0^{\infty} \sum_{i=1}^{n} F_{pd}\right) dt \quad (1)$$

$$\int \left(\int_0^{\infty} \sum_{i=1}^{n} R_d\right) dt = \int \left(\int \sum_{i \rightarrow 1}^{i \rightarrow n} cd + \int \sum_{i \rightarrow 1}^{i \rightarrow n} P_i\right) dt \quad (2)$$

In Eqs. (1) and (2), $R_c$ describes the cloud session activity throughout the system's run-time or execution, $C_{cc}$ defines block renewal or block schedulers within the current process run-time over the cloud, $F_{pd}$ shows packet or data delivery from the fog cluster, and $R_d$ is a response delay within the carried processes. However, it may vary depending on the process execution; $cd$ stands for the current delay and contains the delay of every run-time process. $P_i$ concludes the packet or data interchange delay while in the configured schedule. The relationship of this equation relies on the phenomena of the attacked session and the cloud thus, a time derivation of integral values is essential for the results and experimental output. We verified all of the equation metrics with parametric simulation tools as shown below.

#### 2) CLOUD IN NOT ATTACKED SITUATION

$R_c$ = Cloud Current Computation Rate; $R_d$ = Packet Interchange Rate (Current Delivery Delay)

$$\int (\int_0^{\infty} \sum_{i=1}^{n} R_c)\, dt = \int (\int_0^{\infty} \sum_{i=1}^{n} C_{cc})\, dt \tag{3}$$

$$\int (\int_0^{\infty} \sum_{i=1}^{n} R_d)\, dt = \int (\int_0^{\infty} \sum_{i=1}^{n} D_d)\, dt \tag{4}$$

In Eqs. (3) and (4), $R_c$ describes the cloud session activity throughout the system's run-time or execution, $C_{cc}$ defines block renewal or block schedulers within the current process run-time over the cloud, and $R_d$ is a response delay within the carried processes. However, it may vary depending on the process execution. $D_d$ stands for the current packet delivery rate within every run-time of processes for an hourly limit. This equation's relationship relies on the phenomena of the not attacked session and cloud response delivery. This relationship is being carried with the developed algorithmic scheme, and the integral values depend on every time limit; thus, a time derivation of integral values is essential for the results and experimental output.

#### 3) FOG IN ATTACKED SITUATION

$R_f$ = Fog Current Computation Rate (Attacked Block) + IoT Packet Delivery; $R_d$ = Packet Interchange Rate (Current Delivery Delay) + IoT Packet (insertion)

$$\int (\int_0^{\infty} \sum_{i=1}^{n} R_f)\, dt = \int (\int_0^{\infty} \sum_{i=1}^{n} A_{bc} + \int_0^{\infty} \sum_{i=1}^{n} I_{pd})\, dt \tag{5}$$

$$\int (\int_0^{\infty} \sum_{i=1}^{n} R_d)\, dt = \int (\int_0^{\infty} \sum_{i=1}^{n} P_{dd} + \int_0^{\infty} \sum_{i\to 1}^{i\to n} D_d)\, dt \tag{6}$$

In Eqs. (5) and (6), $R_f$ describes the fog session activity throughout the system's run-time or execution, $A_{bc}$ defines the attacked block information within the current process run-time over the fog, $I_{pd}$ shows the packet or data delivery from the IoT cluster, and $R_d$ is a response delay within the carried processes. However, it may vary depending on process execution; $P_{dd}$ stands for the packet delivery delay and contains the delay of every run-time packet delivery, and $D_d$ determines the current packet delivery rate within every run-time of a process to give an hourly limit. The relationship of this equation relies on the phenomena of the attacked session and the fog response deliveries. This relationship is carried out with the developed algorithmic scheme and the integral values depend on every time limit; thus, a time derivation of integral values is essential for the resulting output.

#### 4) FOG IN NOT ATTACKED SITUATION

$R_f$ = Fog Current Computation Rate; $R_d$ = Packet Interchange Rate (Current Delivery Delay)

$$\int (\int_0^{\infty} \sum_{i=1}^{n} R_f)\, dt = \int (\int_0^{\infty} \sum_{i=1}^{n} F_{cc})\, dt \tag{7}$$

$$\int (\int_0^{\infty} \sum_{i=1}^{n} R_d)\, dt = \int (\int_0^{\infty} \sum_{i\to 1}^{i\to n} D_d)\, dt \tag{8}$$

In Eqs. (7) and (8); $R_f$ describes the fog session activity throughout the system's run-time or execution, $F_{cc}$ defines the block renewal or block schedulers within the current process's run-time over fog clusters, and $R_d$ is a response delay within the carried out processes. However, it may vary depending on process execution. $D_d$ determines the current packet delivery rate within every run-time of processes to give an hourly limit. This equation's relationship relies on the phenomena of the not attacked session and the fog response delivery. This relationship is carried out with the developed algorithmic scheme, and the integral values depend on every time limit; thus, a time derivation of integral values is essential for the resulting output.

#### 5) IoT IN ATTACKED SITUATION

#### 6) $R_i$ = 5 MINUTES + IoT CURRENT COMPUTATION RATE (ATTACKED BLOCK) + NO PACKET DELIVERY; $R_d$ = PACKET INTERCHANGE RATE (CURRENT DELIVERY DELAY) + NO PACKET (INSERTION)

$$\int (\int_0^{\infty} \sum_{i=1}^{n} R_i)\, dt = \int (\int_0^{5} t^5 + \int_0^{\infty} \sum_{i=1}^{n} A_{bc} + \int_0^{\infty} \sum_{i\to 0}^{i\to 0} I_{pd})\, dt \tag{9}$$

$$\int (\int_0^{\infty} \sum_{i=1}^{n} R_d)\, dt = \int (\int_0^{\infty} \sum_{i\to 1}^{i\to n} D_d + \int_0^{\infty} \sum_{i\to 0}^{i\to 0} P_i)\, dt \tag{10}$$

In Eqs. (9) and (10), $R_i$ describes the IoT session activity throughout the system's run-time or execution, $A_{bc}$ defines block renewal or block schedulers within the current process's run-time over IoT clusters, $I_{pd}$ shows packet or data delivery from the IoT cluster, and $R_d$ is a response delay within the carried processes. However, it may vary depending on process execution. $D_d$ concludes the current packet delivery rate within every run-time of processes to give an hourly limit. $P_i$ stands for the interchange delay of every run-time packet delivery. The relationship of this equation relies on the phenomena of the attacked session and the IoT response delivery. This relationship is carried out with the developed algorithmic scheme, and the integral values depend on every time limit; thus, a time derivation of integral values is essential for the resulting output.

#### 7) IoT IN NOT ATTACKED SITUATION

$R_i$ = IoT Current Computation Rate; $D_d$ = Packet Interchange Rate (Current Delivery Delay)

$$\int (\int_0^{\infty} \sum_{i=1}^{n} R_i)\, dt = \int (\int_0^{\infty} \sum_{i=1}^{n} I_{cc})\, dt \tag{11}$$

$$\int (\int_0^{\infty} \sum_{i=1}^{n} R_d)\, dt = \int (\int_0^{\infty} \sum_{i \to 1}^{i \to n} D_d)\, dt \tag{12}$$

In Eqs. (11) and (12), $R_i$ describes the IoT session activity throughout the system's run-time or execution, $I_{cc}$ defines block renewal or block schedulers within the current process run-time over IoT clusters, and $R_d$ is a response delay within the carried out processes. However, it may vary depending on the process execution. $D_d$ determines the current packet delivery rate within every run-time of processes to give an hourly limit. This equation's relationship relies on the phenomena of the not attacked session and the IoT response delivery. This relationship is carried out with the developed algorithmic scheme, and the integral values depend on every time limit; thus, a time derivation of integral values is essential for the resulting output.

## IV. OUR NEW ALGORITHM AND THE DERIVED DEVELOPMENT PROCESS

The established and configured algorithms work initially in the IoT network to secure the IoT nodes via fog commandments. Later on, the direction of algorithmic processes proceeds towards the fog to secure fog blocks. The central server cloud network separates from the blockchain procedure to reduce the data load and scale it efficiently. Initially, the blockchain process runs through the fog and IoT networks. When an attack on the IoT layer occurs, the IoT blocks are shut down before being entirely demolished to operate the network. Then, the carried information passes to the fog network from the IoT. Detail work pattern is employed in Fig. 3.

The fog network then creates a new blockchain process by maintaining a delay (e.g., configured OTSP for block generation) to rebuild the IoT network. The information from broken IoT blocks is transferred to the central server cloud via fog to save the data as a record log. The system's algorithmic process design ensures that there is a generous delay on both working sides (e.g., fog or IoT). The main focus is to confuse the intruder parties; depicted in Fig. 4. The fog network is also overcome through the same configured process if an attack occurs. Hence, we depicted the run-time conditions around the IoT, fog, and cloud networks as a conceptual scheme through the algorithmic processes.

## V. EXPERIMENTAL RESEARCH AND SIMULATION

We have compared the established scheme (e.g., BFIM) with the traditional AES, RSA, and DES algorithms. The algorithmic performance considerations were checked under three parametric service provisioning metrics across the cloud, fog, and IoT levels: the process execution delay (time), the block generation (time), the generation of the delay-time latency, and the process to process delay (time) (refer to Table 3.).

**TABLE 3. Notation used in mathematical equations.**

| Symbol | Description |
|---|---|
| $C$ | Cloud Process Interaction |
| $F$ | Fog Process Interaction |
| $R$ | Response |
| $cc$ | Current Computation |
| $pd$ | Packet Delivery |
| $cd$ | Current Delay |
| $R_c$ | Response from Cloud |
| $R_f$ | Response from fog |
| $R_i$ | Response from IoT |
| $R_d$ | Response delay |
| $A_{bc}$ | Attacked Block Computation |
| $I_{pd}$ | IoT Packet Delivery |
| $F_{cc}$ | Fog Current Computation |
| $I_{cc}$ | IoT Current Computation |
| $P_{dd}$ | Packet Delivery Delay |
| $D_d$ | Current Packet Delivery Rate |
| $\infty$ | System Runtime |
| $N$ | $N^{th}$ Packet |
| $i$ | Packet Interchange Delay |

### A. SCENARIO OVER THE AES, DES AND RSA CONSIDERING CLOUD LEVEL COMMUNICATION (CLOUD LAYER-IN THE CORE)

The generated graphs depict that during the attack, the AES performs faster than the DES and RSA. The DES has process efficiency values of about 96.8% and 99%, respectively, in the cloud layer, and its efficiency is about 75.53% in an average service provisioning to regenerate new blocks in comparison to the AES and RSA. In the no attack situation, the AES and DES remain the same, but the RSA has a service efficiency (average) of 89.6% compared to them. However, in the information passing scenario, the RSA has process efficiency values of 89.35% and 96.19%. However, during service provisioning, it has an average process delivery efficiency of about 76.95% compared to the AES and DES during process execution (Fig. 5(a).) (Fig. 5(b).) (Fig. 5(c).). On the other hand, the RSA seems to be quite inefficient in this cloud phase round when it is involved in higher (average) delay assessments.

### B. SCENARIO OVER THE AES, DES AND RSA CONSIDERING FOG LEVEL COMMUNICATION (FOG LAYER-IN THE MIDDLE-WARE)

The generated graphs show that during an attack, the DES performs faster than the AES and RSA. The RSA and AES respectively have process time efficiencies of 90.36% and 93.63% (e.g., task deliverable) compared with the DES in the fog layer, and the average service provisioning efficiency to regenerate new blocks is 73%. Moreover, in the no attack situation, the RSA and AES have processing efficiencies of 82.12% and 96.13% respectively, and an average service efficiency of 78% compared with the DES. Nonetheless,

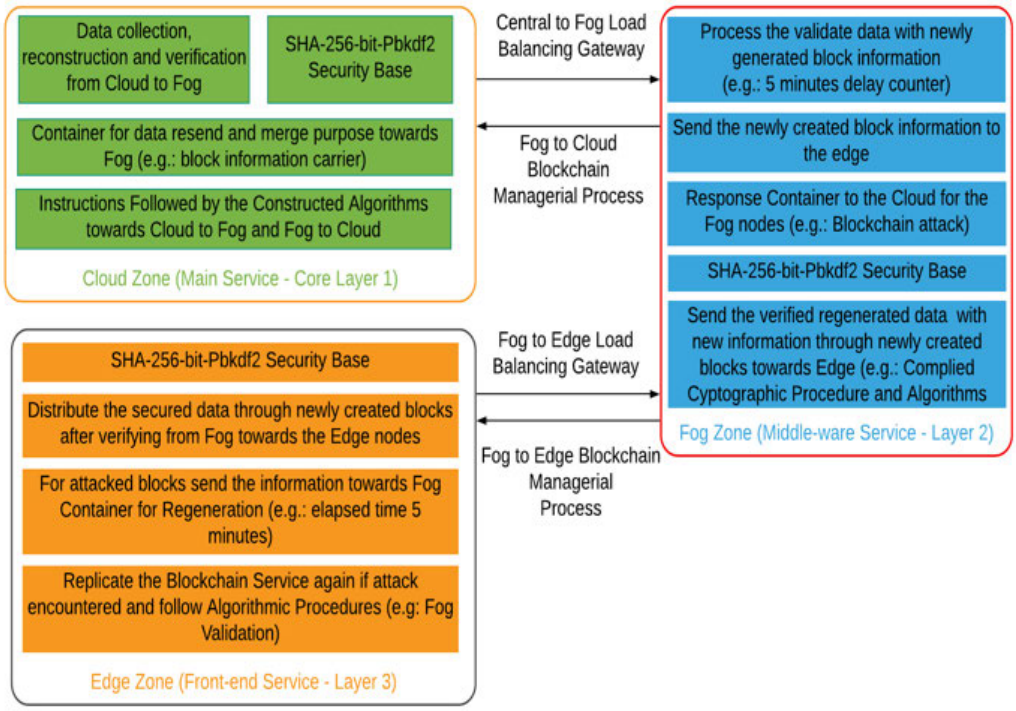



**FIGURE 3.** Blockchain-based secured Fog-IoT (e.g., edge or IoT) process allocation (Block Map).

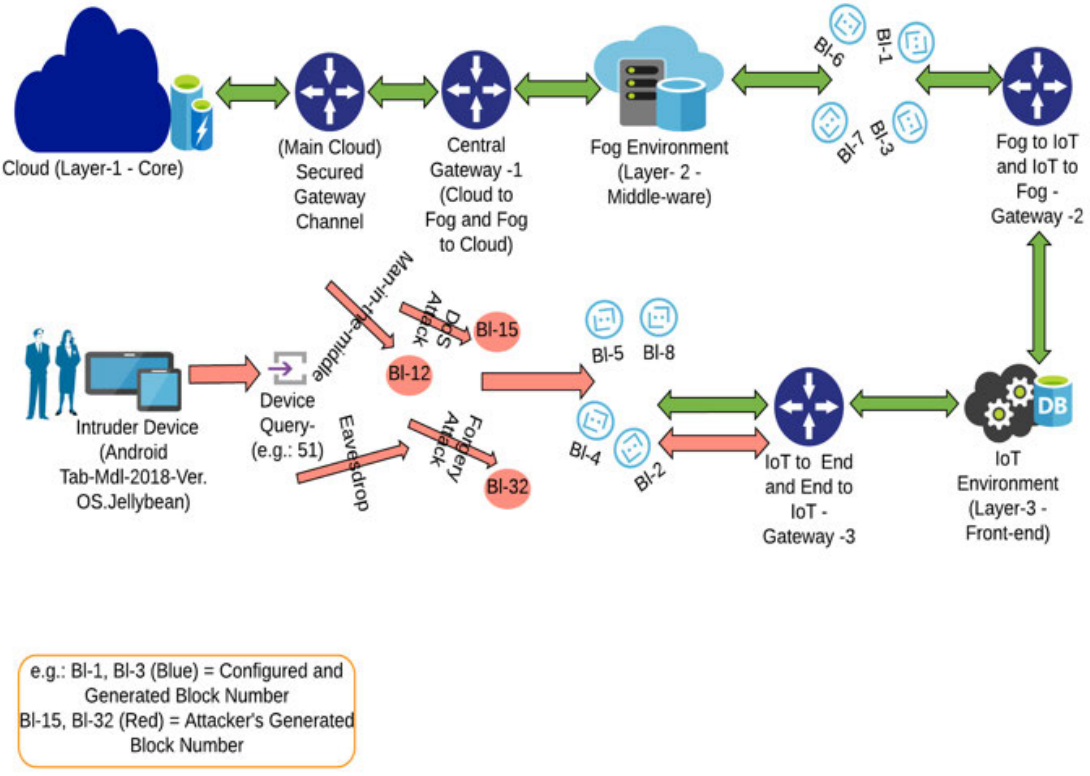



**FIGURE 4.** Blockchain-based secured Fog-IoT architecture and possible attack scenarios.

in the information passing scenario, the DES performs better within an average delay performance comparison, whereas the AES and RSA have task processing efficiencies of 95% and 91%, respectively, and an average service efficiency of 73.25% compared with the DES (Fig. 6(a).) (Fig. 6(b).) (Fig. 6(c).). However, the RSA seems to be quite inefficient in this middle-ware phase round when it is involved in higher process delay assessments.

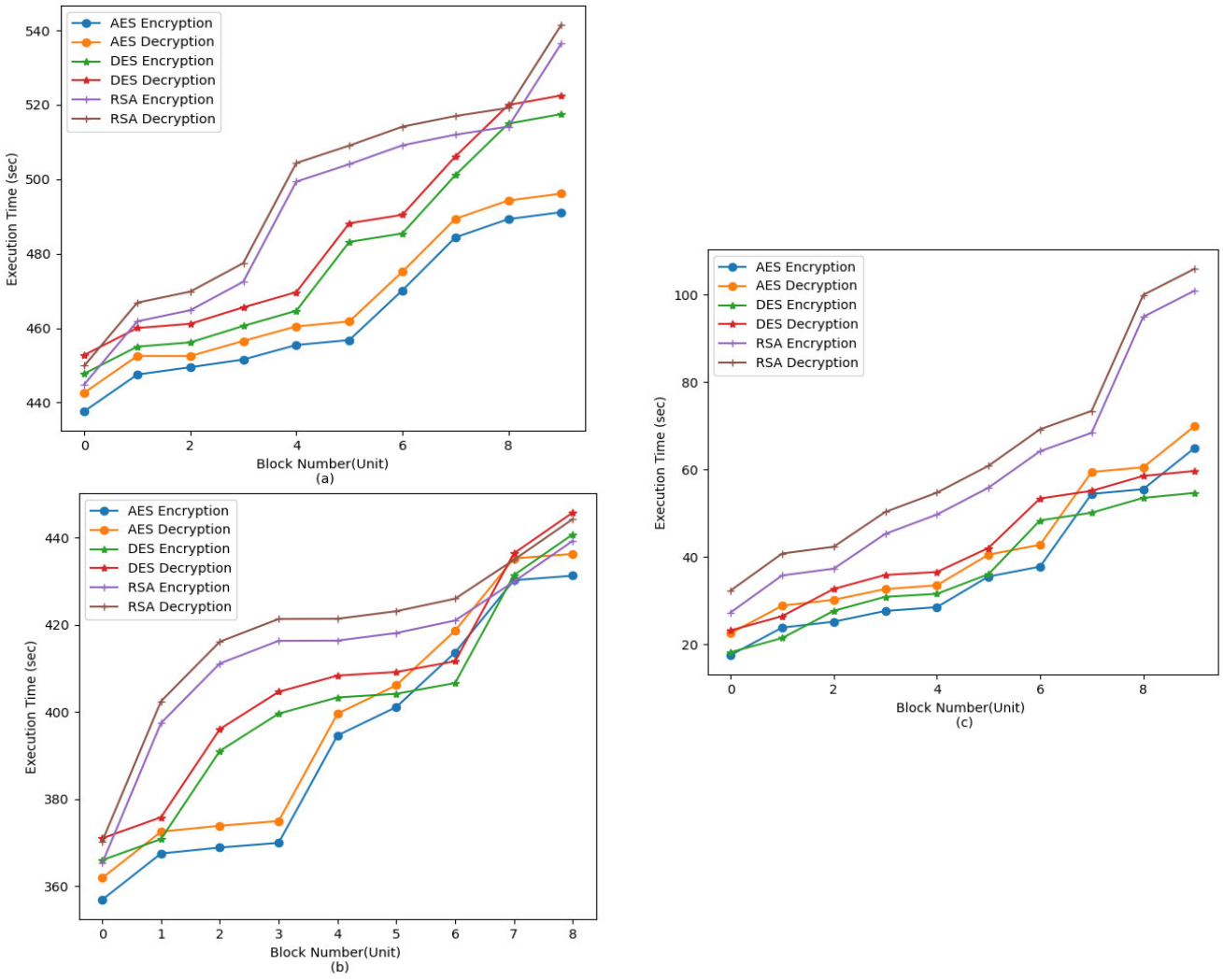



**FIGURE 5.** AES, DES, RSA Encryption and Decryption process scenario [e.g., information passing (a), Attacked (b) and not attacked (c)] within the cloud layer (core).

### C. SCENARIO OVER THE AES, DES AND RSA CONSIDERING IoT LEVEL COMMUNICATION (IoT LAYER-THE FRONT-END)

The generated graphs show that during the attack, the RSA performs faster than the AES and DES, providing a shorter time delay in the processes. AES and DES respectively encapsulate 76.87% and 93% higher process efficiencies (e.g., task deliverable) in comparison with the RSA in the IoT layer. The AES and DES have average service provisioning efficiencies of about 75.88% and 73.42% when regenerating new blocks. Moreover, in the no attack situation, the AES has an enhanced average service efficiency by 86.33% and an enhanced process efficiency (e.g., task scheduling) by about 61.62% and 76.15% compared with the DES and RSA, respectively. Nonetheless, in the information passing scenario, the AES has a better process delivery performance by about 79.53% and 66.33% respectively and an average service provisioning of about 85.9% in comparison with the DES and RSA (Fig. 7(a).) (Fig. 7(b).) (Fig. 7(c).). However, the AES seems to be quite inefficient in this IoT phase round when higher process delay assessments are involved.

### D. AES, RSA, DES AND PROPOSED ALGORITHM CONSIDERING IoT LEVEL COMMUNICATION (IoT LAYER-THE FRONT-END)

The generated graphs depict that during the attack, the RSA performs faster than our algorithm and the AES, with a shorter time delay (average) of in service provisioning. Our algorithm and the AES respectively have 80.54% and 85.51% greater process time efficiencies within the IoT layer in comparison with the RSA. Moreover, in this layer, our algorithm surpasses the DES and RSA with a process time efficiency that is about 85.94% and 78.52% greater, respectively. In the IoT layer, the average service provisioning comparison is about 83.24% greater through having a blockchain workflow pattern. The performance of our developed algorithm is superior to that of the DES and AES, and it overcomes a task scheduling possibility of 85.25% and provides better service

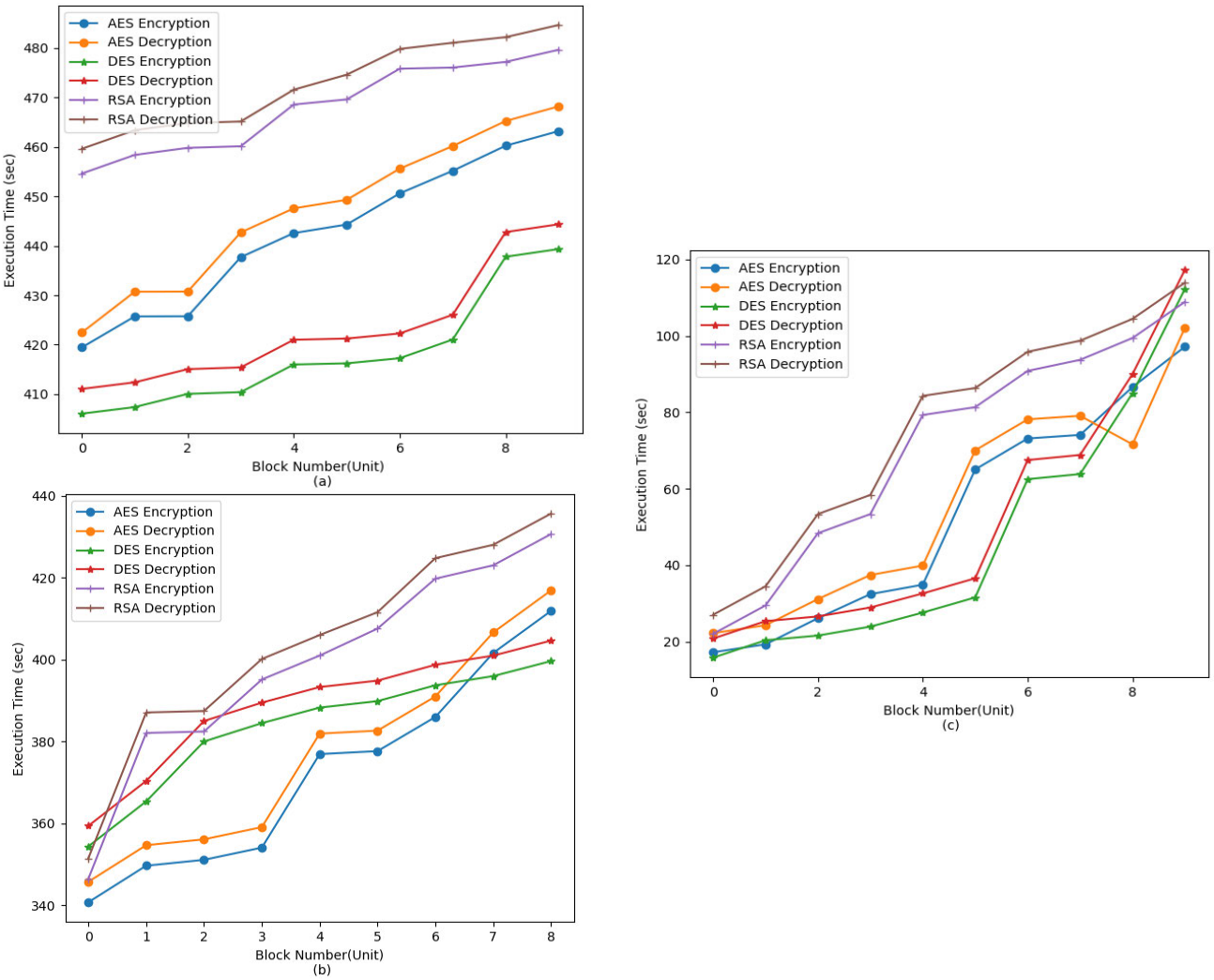



**FIGURE 6.** AES, DES, RSA Encryption and Decryption process scenario [e.g., Information passing (a), Attacked (b) and not attacked (c)] within the Fog layer (middle-ware).

delivery (on average) than the RSA (Fig. 8(a).-Attacked) (Fig. 8(b).-Attacked).

Meanwhile, in the no attack situation, our new algorithm performs faster by having a lower execution delay throughout service provisioning. In comparison to the Proposed Algorithm, the RSA has a higher process efficiency (e.g., task deliverable) by 76.38% across the edge layer. On the other hand, in this situation (e.g., no attack), the AES has greater process execution (e.g., task) by about 72.68%, whereas our algorithm has a process execution (e.g., task) of about 83.27% and an average service provisioning of about 83.24%. Our algorithm enhances the process delivery with an average task delivery of about 21.21% and a task scheduling process efficiency of about 76.38% in comparison with the DES and RSA, respectively.

Nonetheless, in this scenario, the AES has higher rates through having an average process delivery performance of about 83.15%. However, in this situation, the DES performs better while having a process delivery of about 96.25%. In the blockchain equipped 'BFIM' Algorithm the RSA, DES, and AES have average service provisioning values of about 83.24%, 84.35%, 96.25%, and 83.15% respectively (Fig. 8(a).-Non Attacked) (Fig. 8(b).-Non Attacked). However, the proposed algorithm has a better process delivery ratio (e.g., service provisioning) than the DES of about 85.48%. Yet, in comparison to other algorithmic schemes, the 'BFIM' scheme is good for service provisioning while having better administrative security privileges. Table 4. shows the system description for simulation environments.

### *E. AES, RSA, DES AND PROPOSED ALGORITHM PERFORMANCE ENHANCEMENT THROUGHOUT FOG LEVEL COMMUNICATION (FOG LAYER-THE MIDDLE-WARE)*

The generated graphs show that during information passing, the AES performs faster than the RSA and our algorithm, provide less time delay (average). Our algorithm has a greater process time delay within the fog layer by 8.18% and 6.04%

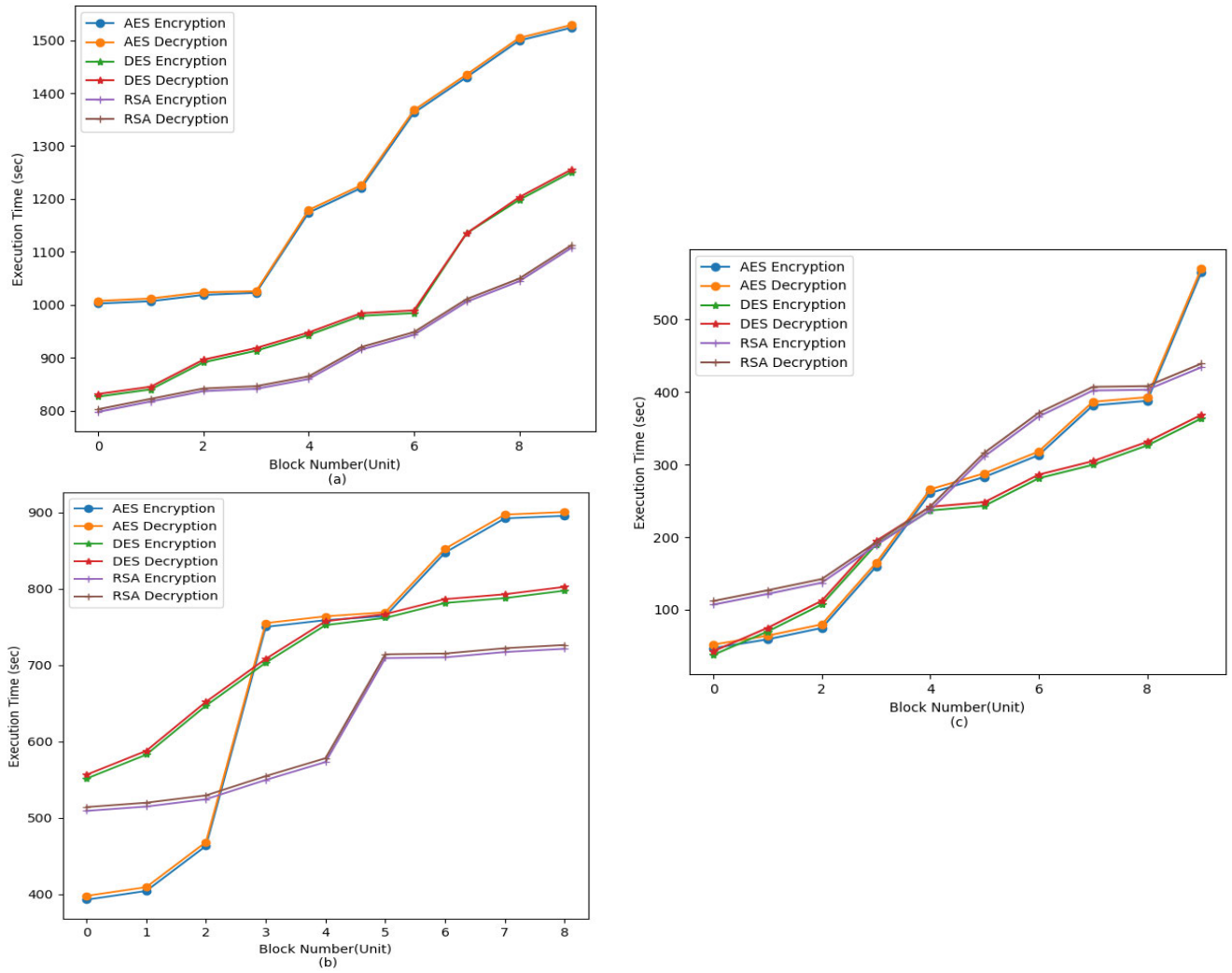



**FIGURE 7.** AES, DES, RSA Encryption and Decryption process scenario [e.g., Information passing (a), Attacked (b) and not attacked (c)] within the IoT layer (front-end).

**TABLE 4.** System descriptions for simulation environments.

| System properties | Specification |
|---|---|
| Operating System | Android Jellybean 4.2 ; Lin-Ubuntu 16.04 ; Win-10 |
| Processor | Qualcomm Snapdragon 636 ; Intel-i7-6700K ; Intel-Xeon-D-1653N |
| Clock Rate | 1.8 GHz ; 3.2 GHz ; 4.2 GHz |
| Device Type | Amazon Fire HD 7 ; HP Z4 G4 Workstation ; Dell Optiplex 3046 MT |
| Storage | 120 GB (HDD) ; 4 TB (SSD) ; 20 TB (SSD) |
| RAM | 4 GB ; 64 GB ; 256 GB |
| Number of Cores | 2 ; 8 ; 16 |
| System type | 64 bit OS ; x64 based processor |
| Simulation Framework | NSNam 3.3 ; Xgraph 3.3.0 ; R 4.0.3 |
| Programming Language | Java 9 ; C/C++ 17 ; Python 3.5.1 |
| IDE | NS3 3.33 ; OMNET++ 5.5.1 ; Jupyter Notebook 6.1.0 |
| Distribution | Exponential ; Geometric (Continuous Process) |

respectively in comparison to the RSA and DES, but it has efficient service delay provisioning (average) of about 12.71%. On the other hand, in this situation (e.g., information exchanging), the AES, DES, and RSA have average service delay provisioning values of 10.58%, 11.15%, and 12.01%, respectively. In this case, our algorithm performs almost the same as the AES in terms of the process time delay and overall service provisioning (see Fig. 9(a).).

Nonetheless, in the information passing scenario, our algorithm shows slightly higher rates with an average resource delivery of about 81.68%. The 'BFIM' scheme shows a remarkable change in average service provisioning of about 83.24% over the fog layer rather than the cloud layer, except in the AES, which shows a change of 89.42%. However, in terms of performance comparison with the DES, RSA, and AES, the blockchain equipped 'BFIM' scheme shows better results (see Fig. 9(b).). Moreover, running with less service provisioning delay and (see Fig. 9.) better security support, our algorithm 'BFIM' scheme is a good option (see Fig. 10.) for the middle-ware and vulnerable IoT layer service provisioning.

## VI. DISCUSSION

In this work, we provided a scheme to ensure data security within a fog–IoT network through a microservice-based

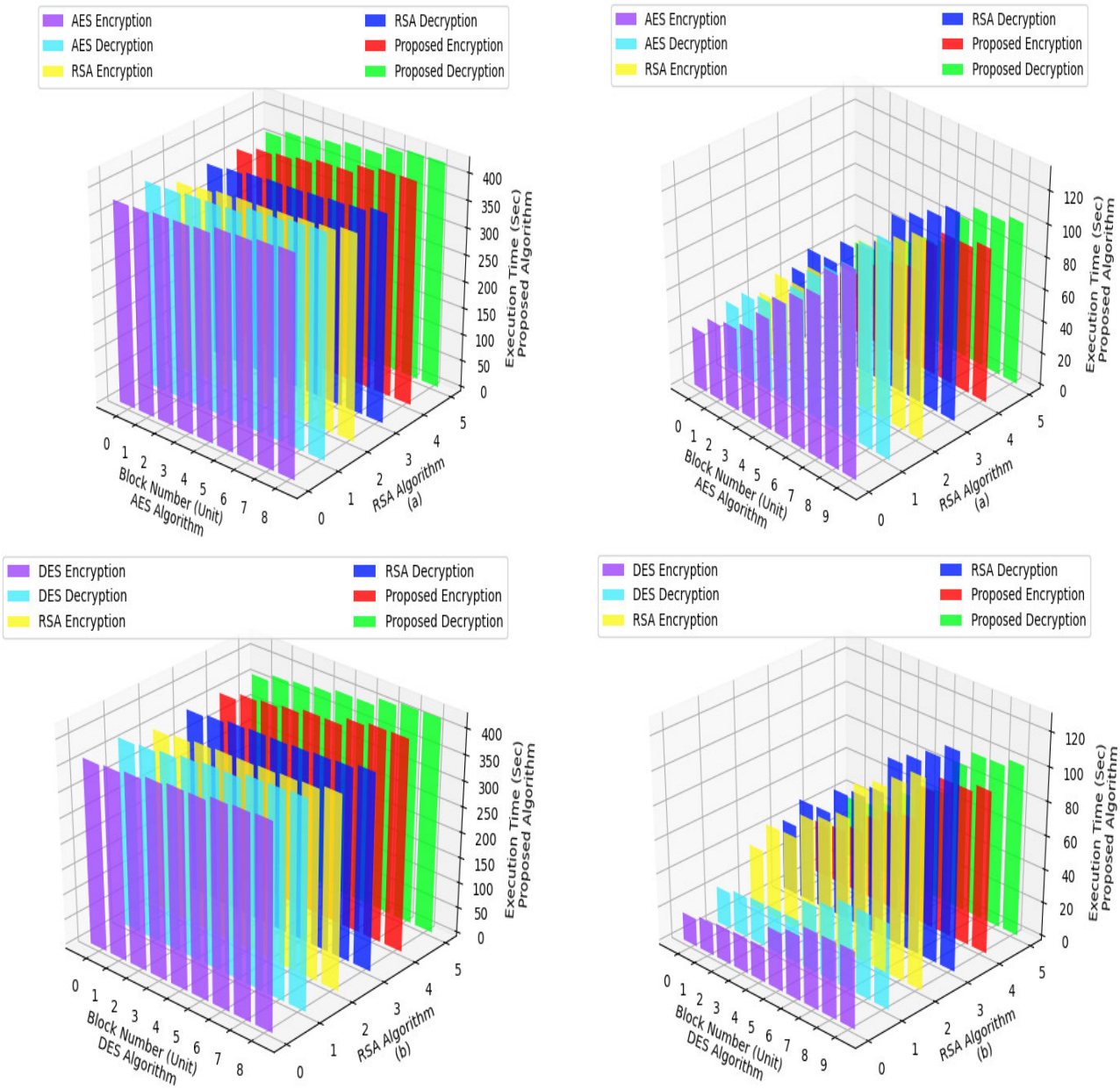



**FIGURE 8.** AES, DES, RSA Encryption and Decryption process scenario [e.g., Attacked AES, RSA and our algorithm (a), attacked DES, RSA and our algorithm (b); Not attacked AES, RSA and our algorithm (a), Not attacked DES, RSA and our algorithm (b)] within the IoT layer (front-end).

blockchain security algorithm. We employed a distributed hierarchical-tree-based mesh topology to display the data exchange within the heterogeneous fog–IoT orchestrations scenario through the developed blockchain mechanism. If an intruder attacks an IoT device within the IoT network layer, the valuable information around the network might be in danger of being stalked. Our scheme prevents this by breaking the network chain using an intelligent-timer-based algorithmic scheme (e.g., OTSP). Thus, the system is configured as a three-layer architecture where the central server cloud acts as a hidden backbone (e.g., reservoir). The main server (e.g., central cloud) processes the current operation by collecting the attacked block's information to give the command to regenerate the block queries towards the fog. The newly processed information is blocked from the fogs and then is delivered towards the IoT bases. Our decentralized blockchain process is compiled with an OTSP based Pbkdf2 security mechanism that works with an iterative password derivation process while having an elapsed session-based delay time for the individual devices. Thus, it provides identity authentication and verification of the devices within the data-link layer. An unauthenticated user will find a mismatch throughout the security mechanism. This part of our blockchain algorithmic scheme delivers a tamper-resistant strategy and is efficient across an authenticated user traceability pattern.

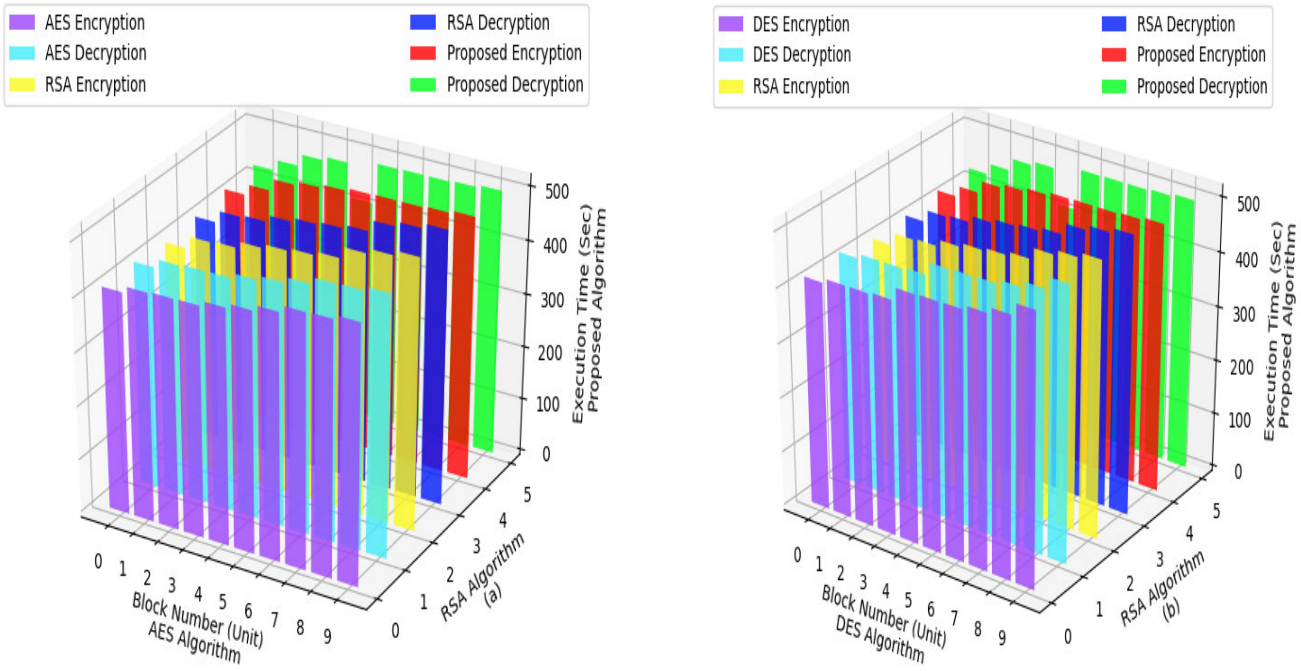



**FIGURE 9.** AES, RSA and our algorithm data passing scenario (a), DES, RSA and our algorithm data passing scenario (b) and our new algorithm Performance enhancement in Fog layer [e.g., Cloud-Fog comparison] within the Fog layer (or middle-ware).

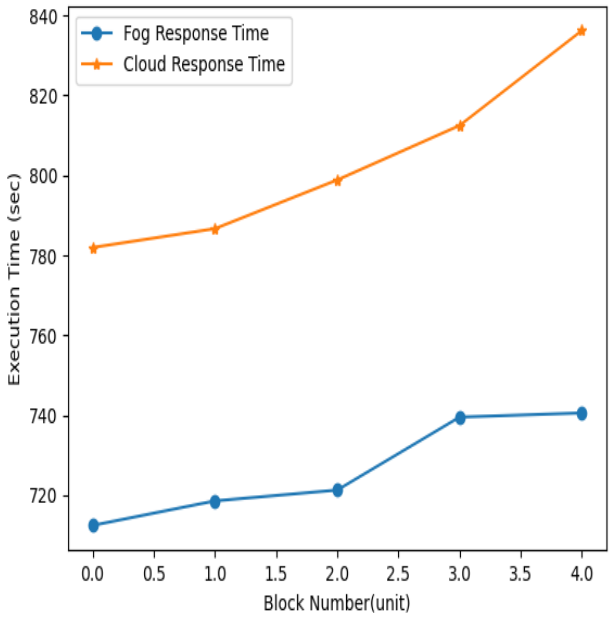



**FIGURE 10.** The 'BFIM' service response in Fog layer in comparison with the Cloud layer running with AES, DES and RSA [e.g., average response time].

In this new algorithmic scheme, the delay generated (i.e., regenerated) IoT network nodes have sessions with new IPs of information (e.g., OTSP session interchange format) from the fogs. The IP generation timestamp is carried upon an interchange if any suspicious activity of hashing power misagreements is found. In this architecture, the client sides (e.g., IoT and fog) rely on a hard fork blockchain pattern, whereas the server-side (e.g., cloud) acts on both soft and hard fork patterns. In the attacking situation, the disabled node sends information towards the adjacent node.

The intermediate node carries the disabled and its own information towards the cloud repository from the fogs (e.g., containers for attacked fog or IoT information). The established blockchain process will continue to regenerate the number of building blocks in an iterative fashion every time any third-party intruder is detected.

## VII. CONCLUSION

Maintaining data security within the heterogeneous cloud fog IoT network ecosystem is a challenging task. In this research, we developed a secure blockchain scheme for a fog–IoT hierarchical tree-based overlay mesh architecture. The performance of our algorithmic security scheme 'BFIM' was superior to that of the AES, DES, and RSA algorithms. The expanded blockchain-based algorithmic procedure suppresses the AES, DES, and RSA algorithms with an average efficiency in task delivery of 78.79%, especially in not attacked situations over a fog environment. The initialized 'BFIM' procedure outperforms others with an overall process delivery of 75% (e.g., time delay, throughput) and a service delivery efficiency of 83.24% (e.g., task scheduling) within the fog network. Individually, compared with RSA and DES, our algorithmic processes perform better while executing operation procedures across the fog. Finally, our implemented scheme has a greater efficiency in service delivery (e.g., task scheduling) of 83.24% across the fog layer compared with the central cloud layer network of the AES, DES, and RSA algorithms.

## REFERENCES


[1] H. Sun, H. Yu, and G. Fan, "Contract-based resource sharing for time effective task scheduling in fog-cloud environment," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 2, pp. 1040–1053, Jun. 2020.

[2] M. Whaiduzzaman, A. Gani, N. B. Anuar, M. Shiraz, M. N. Haque, and I. T. Haque, "Cloud service selection using multicriteria decision analysis," *Sci. World J.*, vol. 2014, Feb. 2014, Art. no. 459375.

[3] M. Whaiduzzaman, M. N. Haque, K. C. Rejaul, and A. Gani, "A study on strategic provisioning of cloud computing services," *Sci. World J.*, vol. 2014, Jun. 2014, Art. no. 894362.

[4] M. Ingham, J. Marchang, and D. Bhowmik, "IoT security vulnerabilities and predictive signal jamming attack analysis in LoRaWAN," *IET Inf. Secur.*, vol. 14, no. 4, pp. 368–379, 2020.

[5] T. Lin, X. Yang, T. Wang, T. Peng, F. Xu, S. Lao, S. Ma, H. Wang, and W. Hao, "Implementation of high-performance blockchain network based on cross-chain technology for IoT applications," *Sensors*, vol. 20, no. 11, p. 3268, Jun. 2020.

[6] S. Sahoo, A. M. Fajge, R. Halder, and A. Cortesi, "A hierarchical and abstraction-based blockchain model," *Appl. Sci.*, vol. 9, no. 11, p. 2343, Jun. 2019.

[7] J. Hu, M. Reed, N. Thomos, M. F. Ai-Naday, and K. Yang, "Securing SDN-controlled IoT networks through edge blockchain," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2102–2115, Feb. 2021.

[8] A. Javadpour, G. Wang, and S. Rezaei, "Resource management in a peer to peer cloud network for IoT," *Wireless Pers. Commun.*, vol. 115, pp. 2471–2488, Aug. 2020.

[9] M. Whaiduzzaman, A. Naveed, and A. Gani, "MobiCoRE: Mobile device based cloudlet resource enhancement for optimal task response," *IEEE Trans. Services Comput.*, vol. 11, no. 1, pp. 144–154, Jan. 2018.

[10] J. Moura and D. Hutchison, "Fog computing systems: State of the art, research issues and future trends, with a focus on resilience," *J. Netw. Comput. Appl.*, vol. 169, Nov. 2020, Art. no. 102784.

[11] A. Vishwanath, R. Peruri, and J. (Selena) He, *Security in Fog Computing Through Encryption*. Chesterfield, MO, USA: MECS Press, 2016.

[12] S. Singh and N. Singh, "Blockchain: Future of financial and cyber security," in *Proc. 2nd Int. Conf. Contemp. Comput. Informat. (IC3I)*, vol. 2, Dec. 2016, pp. 463–467.

[13] H. F. Atlam, A. Alenezi, M. O. Alassafi, and G. Wills, "Blockchain with Internet of Things: Benefits, challenges, and future directions," *Int. J. Intell. Syst. Appl.*, vol. 10, no. 6, pp. 40–48, 2018.

[14] V. Sharma, I. You, F. Palmieri, D. N. K. Jayakody, and J. Li, "Secure and energy-efficient handover in fog networks using blockchain-based DMM," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 22–31, May 2018.

[15] H. Zenil, N. Kiani, and J. Tegnér, "A review of graph and network complexity from an algorithmic information perspective," *Entropy*, vol. 20, no. 8, p. 551, Jul. 2018.

[16] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," *Concurrency Comput., Pract. Exper.*, vol. 28, no. 10, pp. 2991–3005, 2016.

[17] M. Dehmer and S. Pickl, "Network complexity measures. an information-theoretic approach," *J. Systemics, Cybern. Inform.*, vol. 13, no. 2, pp. 64–67, 2015.

[18] P. Mahajan and A. Sachdeva, "A study of encryption algorithms AES, DES and RSA for security," *Global J. Comput. Sci. Technol.*, vol. 13, no. 15, pp. 1–9, 2013.

[19] C.-C. Lu and S.-Y. Tseng, "Integrated design of AES (Advanced encryption Standard) encrypter and decrypter," in *Proc. IEEE Int. Conf. Appl.-Specific Syst., Archit., Processors*, Jul. 2002, pp. 277–285.

[20] T. Nie and T. Zhang, "A study of Des.and blowfish encryption algorithm," in *Proc. IEEE Region 10 Conf. (TENCON)*, Nov. 2009, pp. 1–4.

[21] X. Zhou and X. Tang, "Research and implementation of RSA algorithm for encryption and decryption," in *Proc. 6th Int. Forum Strategic Technol.*, Aug. 2011, vol. 6, no. 2, pp. 1118–1121.

[22] I. C. Lin and T. C. Liao, "A survey of blockchain security issues and challenges," *Int. J. Netw. Secur.*, vol. 19, no. 5, pp. 653–659, 2017.

[23] H. A. Al Hamid, S. M. M. Rahman, M. S. Hossain, A. Almogren, and A. Alamri, "A security model for preserving the privacy of medical big data in a healthcare cloud using a fog computing facility with pairing-based cryptography," *IEEE Access*, vol. 5, pp. 22313–22328, 2017.

[24] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *Proc. Int. Conf. Wireless Algorithms, Syst., Appl.*, vol. 1. Cham, Switzerland: Springer, 2015, pp. 685–695.

[25] J. Kishigami, S. Fujimura, H. Watanabe, A. Nakadaira, and A. Akutsu, "The blockchain-based digital content distribution system," in *Proc. IEEE 5th Int. Conf. Big Data Cloud Comput.*, Aug. 2015, vol. 5, no. 12, pp. 187–190.

[26] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[27] G. Gautam and B. Sen, "Design and simulation of wireless sensor network in NS2," *Int. J. Comput. Appl.*, vol. 113, no. 16, pp. 14–16, Mar. 2015.

[28] B. Girault, S. S. Narayanan, A. Ortega, P. Gonçalves, and E. Fleury, "Grasp: A MATLAB toolbox for graph signal processing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, vol. 1, Mar. 2017, pp. 6574–6575.

[29] A. R. Shovon, S. Roy, T. Sharma, and M. Whaiduzzaman, "A RESTful E-governance application framework for people identity verification in cloud," in *Proc. Int. Conf. Cloud Comput.*, vol. 1. Cham, Switzerland: Springer, 2018, pp. 281–294.

[30] S. Khan, S. Parkinson, and Y. Qin, "Fog computing security: A review of current applications and security solutions," *J. Cloud Comput., Adv., Syst. Appl.*, vol. 6, no. 1, p. 19, 2017.

[31] W.-M. Lee, "Implementing your own blockchain using Python," in *Beginning Ethereum Smart Contracts Programming*, vol. 1. Berkeley, CA, USA: Apress, 2019, pp. 25–59.

[32] M. A. K. Saluja and M. S. A. Darg, "A detailed analogy of network simulators NS1, NS2, NS3 and NS4," *Int. J. Future Revolution Comput. Sci. Commun. Eng.*, vol. 3, no. 12, pp. 291–295, 2017.

[33] A. Stanciu, "Blockchain based distributed control system for edge computing," in *Proc. 21st Int. Conf. Control Syst. Comput. Sci. (CSCS)*, vol. 21, May 2017, pp. 667–671.

[34] K. C. Okafor, I. E. Achumba, G. A. Chukwudebe, and G. C. Ononiwu, "Leveraging fog computing for scalable IoT datacenter using spine-leaf network topology," *J. Electr. Comput. Eng.*, vol. 2017, Apr. 2017, Art. no. 2363240.

[35] E. Gaetani, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "Blockchain-based database to ensure data integrity in cloud computing environments," in *Proc. 1st Italian Conf. Cybersecurity (ITASEC)*, Venice, Italy, vol. 1, 2017, pp. 146–155.

[36] M. Mukherjee, R. Matam, L. Shu, L. Maglaras, and M. A. Ferrag, "Security and privacy in fog computing: Challenges," *IEEE Access*, vol. 5, pp. 19293–19304, 2017.

[37] D. Shrier, W. Wu, and A. Pentland, "Blockchain & infrastructure (identity, data security)," Massachusetts Inst. Technol., Connection Sci., Cambridge, MA, USA, Tech. Rep. 3, vol. 51, no. 3, 2016, pp. 1–19. [Online]. Available: https://www.getsmarter.com/blog/wp-content/uploads/2017/07/mit_blockchain_and_infrastructure_report.pdf

[38] M. Whaiduzzaman, S. R. Tuly, N. Haque, M. R. Hossain, and A. Barros, "Credit based task scheduling process management in fog computing," in *Proc. Pacific Asia Conf. Inf. Syst.*, 2020, vol. 28, no. 1, p. 232.

[39] R. Yegireddi and R. K. Kumar, "A survey on conventional encryption algorithms of cryptography," in *Proc. Int. Conf. ICT Bus. Ind. Government (ICTBIG)*, vol. 1, 2016, pp. 1–4.

[40] A. Alrawais, A. Alhothaily, C. Hu, X. Xing, and X. Cheng, "An attribute-based encryption scheme to secure fog communications," *IEEE Access*, vol. 5, pp. 9131–9138, 2017.

[41] W. Stallings, *The Principles and Practice of Cryptography and Network Security*, vol. 20, no. 1. London, U.K.: Pearson, 2017, p. 7.

[42] M. T. Rahman and M. J. N. Mahi, "Proposal for SZRP protocol with the establishment of the salted SHA-256 bit HMAC PBKDF2 advance security system in a MANET," in *Proc. Int. Conf. Electr. Eng. Inf. Commun. Technol.*, vol. 1, Apr. 2014, pp. 1–5.

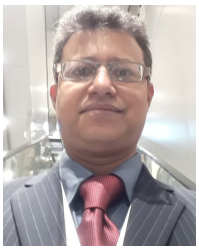
[43] D. P. Mozumder and J. N. Mahi, "Cloud computing security breaches and threats analysis," *Int. J. Sci. Eng. Res.*, vol. 8, no. 1, pp. 1287–1297, 2017.

[44] N. Farjana, S. Roy, J. N. Mahi, and M. Whaiduzzaman, "An identity-based encryption scheme for data security in fog computing," in *Proc. Int. Joint Conf. Comput. Intell.*, vol. 1, no. 1. Singapore: Springer, 2020, pp. 215–226.

[45] G. Ortiz, J. A. Caravaca, A. Garcia-de-Prado, F. O. de la Chavez, and J. Boubeta-Puig, "Real-time context-aware microservice architecture for predictive analytics and smart decision-making," *IEEE Access*, vol. 7, pp. 183177–183194, 2019.

[46] R. Pérez De Prado, S. García-Galán, J. E. Muñoz-Expósito, A. Marchewka, and N. Ruiz-Reyes, "Smart containers schedulers for microservices provision in cloud-fog-IoT networks. Challenges and opportunities," *Sensors*, vol. 20, no. 6, p. 1714, Mar. 2020.

[47] E. Ahmed, A. Akhunzada, M. Whaiduzzaman, A. Gani, S. H. Ab Hamid, and R. Buyya, "Network-centric performance analysis of runtime application migration in mobile cloud computing," *Simul. Model. Pract. Theory*, vol. 50, no. 1, pp. 42–56, Jan. 2015.
[48] M. Whaiduzzaman, K. Oliullah, M. J. N. Mahi, and A. Barros, "AUASF: An anonymous users authentication scheme for fog-IoT environment," in *Proc. 11th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, vol. 11, Jul. 2020, pp. 1–7.
[49] H. Liu, D. Han, and D. Li, "Fabric-IoT: A blockchain-based access control system in IoT," *IEEE Access*, vol. 8, pp. 18207–18218, 2020.
[50] G. Deep, R. Mohana, A. Nayyar, P. Sanjeevikumar, and E. Hossain, "Authentication protocol for cloud databases using blockchain mechanism," *Sensors*, vol. 19, no. 20, p. 4444, Oct. 2019.
[51] R. Xu, S. Y. Nikouei, Y. Chen, E. Blasch, and A. Aved, "BlendMAS: A blockchain-enabled decentralized microservices architecture for smart public safety," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, vol. 1, Jul. 2019, pp. 564–571.
[52] H. Yao, T. Mai, J. Wang, Z. Ji, C. Jiang, and Y. Qian, "Resource trading in blockchain-based industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3602–3609, Jun. 2019.
[53] R. Hossain, M. Whaiduzzaman, A. Barros, S. R. Tuly, and J. N. Mahi, "A scheduling-based dynamic fog computing framework for augmenting resource utilization," *Simul. Model. Pract. Theory*, vol. 111, Sep. 2021, Sep. 102336.



**MD WHAIDUZZAMAN** (Senior Member, IEEE) received the bachelor's degree in electronics and computer science and the M.Sc. degree in telecommunication and computer network engineering from London, U.K, and the Ph.D. degree from the University of Malaya, Malaysia. He currently works as an Associate Professor with the Institute of Information Technology (IIT), Jahangirnagar University. He also works with the Australian Research Council (ARC) funded projects at Queensland University of Technology, Australia. His research interests include mobile cloud computing, vehicular cloud computing, fog computing, microservices, and the IoT. He received the Elsevier *JNCA* Best Paper Award in Paris, France.

**MD. JULKAR NAYEEN MAHI** received the B.Sc. and M.Sc. degrees from IIT, Jahangirnagar University, Bangladesh. He is currently working as a Lecturer with City University, Bangladesh. His current research interests include distributed computer networks, embedded systems, the IoT, data mining, cloud computing, operating systems scheduling approach, network securities, and bio informatics.

**ALISTAIR BARROS** is currently a Professor of information systems and the Head of the Services Computing Program, Information Systems School, Queensland University of Technology. He has 32 years ICT experience across industry, industrial research and development and academic roles, including a Global Research Leader and the Chief Development Architect at SAP AG. His research interests include cloud, enterprise systems and microservices engineering, and evolution and provisioning using model-based techniques.

**MD. IBRAHIM KHALIL** received the B.Sc. degree in computer science and engineering from City University. He worked as a Graphic Designer and a Senior Software Developer (web) in different organizations around Bangladesh, Malaysia, and Singapore. He is currently a Certified and a Professional Wordpress Web Developer and Shopify Theme Expert. His current research interests include cloud native systems, e-commerce security, and distributed blockchain securities.

**COLIN FIDGE** is currently a Full Professor with Queensland University of Technology, Australia, where he teaches software development fundamentals. His research interests include high-integrity software engineering, and modeling and analysis of complex, computer-based systems.

**RAJKUMAR BUYYA** (Fellow, IEEE) is currently a Redmond Barry Distinguished Professor and the Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, The University of Melbourne, Australia. He is also working as the Founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in cloud computing. He serving/served as an Honorary/Visiting Professor for several elite universities, including Imperial College London, U.K., and University of Birmingham, U.K. He has authored over 850 publications and seven text books, including *Mastering Cloud Computing*. He also edited several books, including *Cloud Computing: Principles and Paradigms*. He has been recognized as a "Web of Science Highly Cited Researcher" for five consecutive years, since 2016, a Scopus Researcher of the Year 2017 with Excellence in Innovative Research Award by Elsevier, and the "Best of the World," in Computing Systems field, by The Australian 2019 Research Review. From 2012 to 2016, he worked as a Future Fellow of the Australian Research Council.