JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 13(2), JUNE, 2025
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.com.ng

Byzantine Fault Tolerance Consensus Protocol for IoT Devices

[1]S. K. Alaramma, [2]Adamu Adamu Habu, [3]Maishanu, M., [4]S. W. Mustapha, [5]M. K. Dauda
[1,3&4]Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria,
[2&5]University of St Andrews, Jack Cole Building, North Haugh, St Adrews, KY16 9SX, UK

ABSTRACT
Blockchain relies on a consensus method for agreeing on any new data. Most of the consensus methods which are currently used for the blockchain require high computational power and thus are not apt for resource-constrained systems. Byzantine Fault Tolerant (BFT) based consensus ensures that the blockchain network will continue to operate even in the presence of malicious or failure nodes, thus providing reliability. A typical Internet of Things (IoT) network consists of several devices which have limited computational and communications capabilities. Most often, these devices cannot perform intensive computations and are starved for bandwidth. Therefore, an improved BFT consensus algorithm is proposed in this paper to make it applicable for resource constrained IoT devices and networks. Firstly, by analyzing the architecture and core processes of BFT implementation, the HotStuff, factors affecting its adoptability in blockchain based IoT networks were identified. Secondly, an optimized tree-based HotStuff with sharding and latency compensation techniques is proposed. Finally, the improved consensus scheme is tested and analyzed. Our results show that the scheme significantly improves the performance of HotStuff and makes it suitable for utilization in blockchain IoT networks.

ARTICLE INFO
Article History
Received: January, 2025
Received in revised form: February, 2025
Accepted: April, 2025
Published online: June, 2025

KEYWORDS
Blockchain, BFT, Consensus Mechanism, IoT

INTRODUCTION

Internet of Thing (IoT) applications is gradually becoming common in our houses, offices, neighbourhoods, and cities. Adoption of IoT based technologies are poised to make a big impact on various sectors of our daily lives, including energy, manufacturing, intelligent transportation and smart cities. With more and more autonomous deployments of potentially large scale IoT systems, ensuring security, availability and confidentiality of the data, the devices and the networks become utmost critical (Bhattacharjee, et al., 2017).

Blockchain is decentralized, confidential and traceable, which meets the needs of distributed, interactive and data security of IoT. The consensus algorithm of a blockchain guarantees data security and authenticity across nodes in a blockchain system without central nodes. To realize an entirely autonomous IoT network, different sensors and devices in an IoT network need to communicate with each other in a distributed fashion. This requires a mechanism by which different nodes in an IoT network can agree upon the validity of any communicated data. One of the best ways to achieve this is to use consensus method provided by blockchain. There exist different consensus methods by which different nodes can reach a common decision without the participation of a central controller (Debus, 2017).

In Byzantine Fault Tolerance (BFT) consensus mechanism, all the nodes should participate in the voting process to add the next block and the consensus is reached when more than two-thirds of all nodes agree upon that block. Practical Byzantine Fault Tolerance (PBFT) consensus mechanism can tolerate malicious behaviour from up to one-third of all nodes to perform normally.

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 13(2), JUNE, 2025
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.com.ng

ResilientDB is a recent approach which combines hierarchical algorithms within a blockchain (Rahnama et al., 2020). *ResilientDB* leverages sharding to utilize the left-over resources to boost the system throughput significantly. Nonetheless, it comes with restrictive failure assumptions, as most honest nodes are required within each sub-group. However, the approach results in a significant latency overhead, which affects the potential maximum throughput.

HotStuff is a more recent algorithm that uses a communication pattern based on a star network (Abraham et al., 2018). *HotStuff* consists of three rounds where all communication runs through the leader that collects and distributes aggregated signatures. Being careful in rotating the leader, can help the protocol but does not combat the inherent scalability issues. The protocol has a linear communication cost but takes eight communication steps to terminate as shown in Figure 1. The protocol is very similar to PBFT. Detail explanation of this protocol can be found in (Kokoris-Kogias et al., 2018).

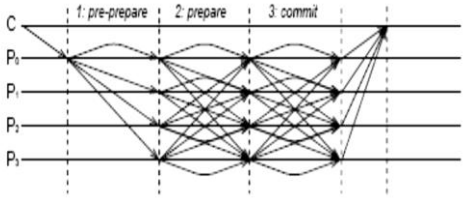

Figure 1: Three Phase Protocol – PBFT (Kokoris-Kogias et al., 2016)

This paper proposes a modified tree-based *HotStuff* consensus mechanism that has low latency, high throughput and scalability which can be well suited for blockchain-based IoT networks.

Analysis of the Performance Constraints of *HotStuff* Consensus Mechanism

The *HotStuff* algorithm is proposed to improve PBFT, making the BFT algorithm and blockchain achieve a better combination. *HotStuff* adopts a partially synchronous network model, and the adversary model is u = 3f +1. *HotStuff* mainly has three adaptions. First, it uses the core technology to process proposals, that is, the message in a round contains a quorum certificate of the previous round and a new proposal. Second, it adopts the BLS threshold signature to aggregate 2f + 1 vote messages into a single signature, cutting down the communication complexity. Third, in each round, the leader who is responsible for collecting votes and sending the proposal will be changed. The essence is to integrate the view-change process into the normal operations by adding a voting round (Liu et al., 2019).

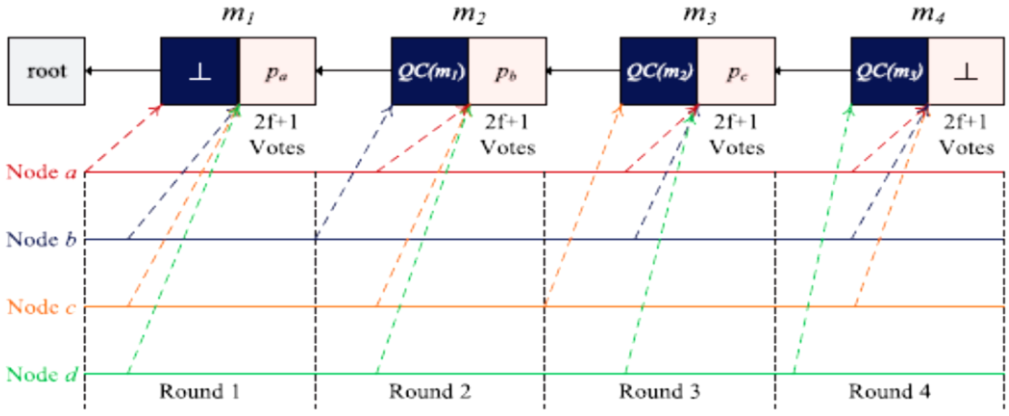

Figure 2: Process of *HotStuff* algorithm. (Abraham et al., 2018).

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 13(2), JUNE, 2025
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.com.ng

As shown in Figure 2, P_a refers to the proposal of node *a* in the first round. The message of the first round is denoted as m_1. So, m_1 is broadcast by node *a*. Then the other nodes verify m_1 and vote for it. Node *b* acts as a leader and collects valid votes. When the number of valid votes reaches 2f + 1, node *b* reconstructs a BLS threshold signature using the *2f + 1* valid votes and forms a quorum certificate QC(m_1). Then node *b* constructs a message m_2 by combining QC(m_1) and a new proposal P_b, and broadcasts m_2 to other nodes. Currently, node *a*, *c* and *d* act as replicas. After receiving m_2, they first verify the validity of QC(m_1) and P_b, and then vote to m_2 if the verification is passed. The subsequent operations are similar. The linear view-change in *HotStuff* refers to the message sent by the leader since only a single threshold signature is needed during view-change instead of 2*f*+ 1 vote in PBFT Yin et al., (2019). Despite its simplicity, with this arrangement, *HotStuff* suffers additional latency and low throughput in addition to scalability issues which might discourage IoT blockchain developers from building applications on top of HotStuff.

Modified *HotStuff* Consensus algorithm

High latency, scalability and low throughput are the major problems that hinder the suitability of *HotStuff* as a consensus mechanism in IoT networks. Sharding and latency compensation techniques have been proposed to be introduced into *HotStuff* to address these issues. Figure 3 shown below depicted the process flow.

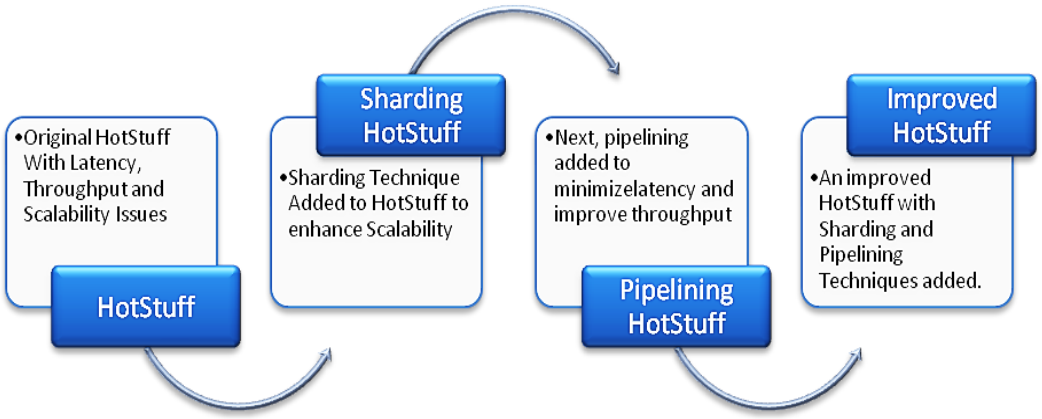

Figure 3: Process workflow of Modified *HotStuff*

Addressing the Scalability Issue

To make *HotStuff* suitable for IoT networks, we introduce sharding which was initially used in database technologies to improve scalability (Luu et al., 2016). As shown in Figure 4, by dividing all participating nodes in the network into multiple shards, each shard is only responsible for maintaining its own corresponding data. In this way, the scalability of network processing capabilities could be achieved. As the number of nodes in the network increases, the enhancement of processing capabilities is realized by adding more shards.

The proposed model would have sharding split the nodes in consensus into groups and utilize parallel processing methods. To provide smooth implementation of sharding into our proposed model, we include node selection, epoch randomness, node assignment, intra-shard consensus, cross-shard transaction processing, and shard reconfiguration modules. Each module would be implemented by different methods, and then by composing all the modules, a complete sharding based consensus mechanism is obtained.

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 13(2), JUNE, 2025
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.com.ng

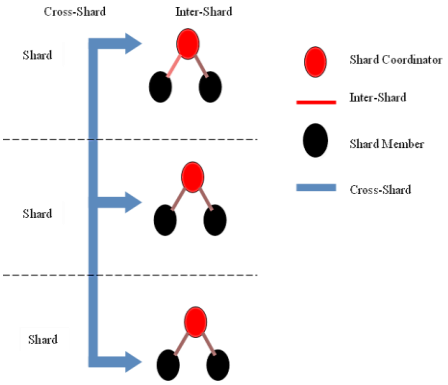

Figure 4: Proposed Sharding in *HotStuff*

Addressing the Throughput and Latency Issues

In *HotStuff*, the leader must disseminate *N* Messages and receive and process *N* Votes in each given round (see Figure 2). Thus, the leader is busy for a long time, which is further aggravated at larger system sizes. Meanwhile, in a tree, each process must disseminate at most *m* messages and process at most *m* votes. Thus, not only will each process be significantly busy for a shorter period, but due to the additional communication steps, between a given "busy period", there are significantly longer idle times. See Figure 5 below.

Figure 5: Idle Time in *HotStuff*

Given there is an ample enough idle time, the leader can use the knowledge of the previous block b_i they proposed, and, tentatively, propose block $b_i + 1, b_i + 2, \ldots, b_i + n$before receiving the consensus result for block b_i. This process is depicted in Figure 6, where each distinct arrow color represents a different block proposal. As we leverage the same parallelism as HotStuff, considering a pipelining stretch of *n* and a given tentative block b_i, the quorum signature of b_i is propagated with the next tentative block in round $b_i + n + 1$. As a result, in the presence of failures, similarly to HotStuff, all tentative blocks that have not received three consecutive quorums have to be aborted, and the pipeline must be rebuilt from scratch. The latency compensation technique in the proposed model and can easily be seen that subsequent blocks are proposed before receiving the signatures of the previous block thereby decreasing the latency incurred.

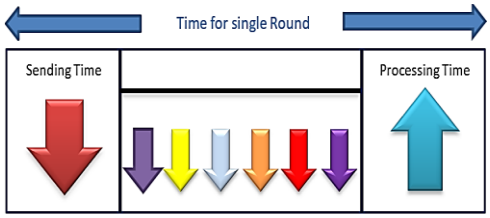

Figure 6: Latency compensation in the proposed model

EXPERIMENTAL RESULTS AND ANALYSIS

Experimental setup

We implemented the prototype of our modified (tree-based) *HotStuff* and *HotStuff* using the Go programming language. For cryptographic operations, we used dedis advanced crypto library in Go. We used SHA256 hashing in the Go crypto library. We tested the models performance on Amazon cloud (AWS) with different network sizes 20, 40, 60, 80, 100, 120, 140, 160 and different block sizes (1MB and 2MB). We used t2:micro replicas in AWS, where each replica has a single vCPU comparable to a single core with 1GB memory. The bandwidth was set to 500 Mb/sec (62.5 MB/sec) and the latency between two endpoints was set to 50 msec. We also performed forking attack on each network of different sizes, k = 1000 times, and took the mean of the average throughput of the network and the latency incurred by blocks affected by this attack.

Improved *HotStuff* experiments Results

We compared our proposed model's performance (throughput and latency) against *HotStuff* in two scenarios: 1) when the previous primary fails and the next primary in the proposed model has to include the aggregated Quorum

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 13(2), JUNE, 2025
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.com.ng

Certificate in the block and 2) when *HotStuff* and our proposed model come under forking attack. Results achieved from each case are discussed below:

Latency and Throughput

To evaluate latency, we measure the time it takes for a single quorum to be obtained for a given set of transactions (block). To evaluate how this compares in practice, we executed a scenario considering a fixed RTT of 100*ms* and varied the bandwidth from 25*Mbs* to 1000*Mbs*. The results of this experiment are shown in Figure 7 for a system size of $N = 100$. First, we note that when sufficient bandwidth is available *HotStuff* will have approximately half the latency. However, with decreasing available bandwidth, proposed model already shows an advantage at 100*Mbs* bandwidth and less than half the latency at 25*Mbs* due to the significantly reduced sending time.

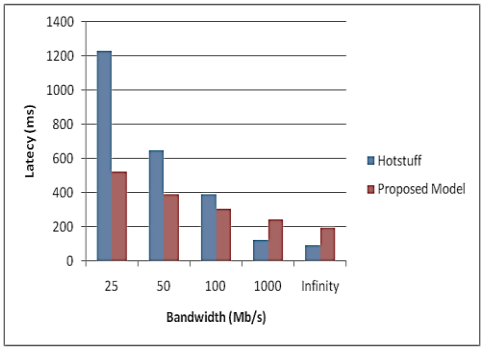

Figure 7: Impact of bandwidth on latency (N=100, RTT=200*ms*).

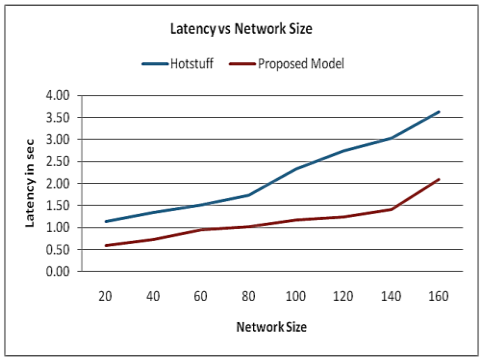

Figure 8(a) Latency for 1 MB Block Size

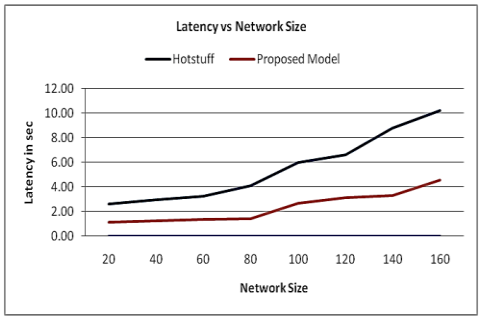

Figure 8(b) Latency under the Forking Attack for 1MB Block Size.

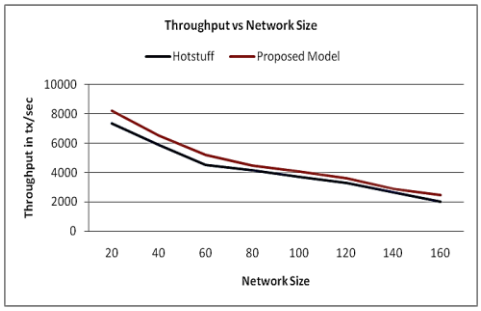

Figure 9(a) Throughput for 1MB Block Size

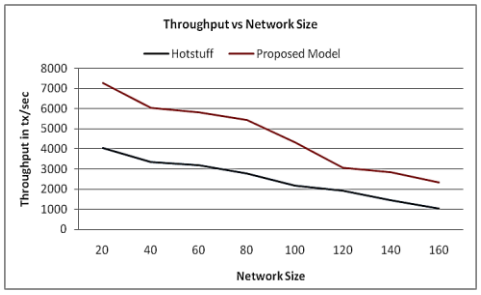

Figure 9(b) Throughput under the Forking Attack for 1MB Block Size.

Just after GST (after primary failure), when an honest (correct) primary is selected, *HotStuff* does not incur additional overhead. However, *HotStuff* needs an additional round of consensus during failure cases in comparison to proposed model. As the results shown in Figure 8(a), when failure occurs, our proposed model outperforms *HotStuff* in terms of latency. HotStuff's throughput slightly decreases against proposed model's throughput due to $O(n^2)$ time

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 13(2), JUNE, 2025
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.com.ng

complexity for interpolation calculation at the primary when network size increases (Figure 9(a)).

Forking attack effects in our proposed model and *HotStuff* can be observed in Figure 8(b) and 9(b). Since forking is not allowed in our proposed model, the proposal will be rejected and not chosen again. Hence, forking attack has no significant effect on the performance of the proposed model.

Scalability

Here we implemented sharding consisting of seven shards in *HotStuff* and vary the system from 100 to 800 nodes with (200*ms* RTT and 25*Mb/s* Bandwidth). The results of this experiment are depicted in Figures 10. First, we note that our proposed model outperforms *HotStuff* by a large margin independent of the scenario and number of processes. This is especially apparent in scenarios with limited bandwidth and at larger system sizes (see Figure 10(a)). *HotStuff* uses signatures without any aggregation mechanism. As such, larger sets of signatures have to be disseminated alongside the block at larger numbers of participants, which further aggravates the problem. With sharding introduced into HotStuff, the throughput has been better amidst increasing processes since processes are grouped into shards and managed their communication and computations in the shard.

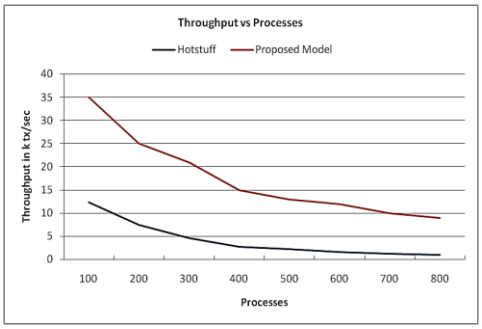

Figure 10(a). Throughput with increasing number of processes

In this experiment, we assumed fixed depth h =2 for our proposed model and adjusted the fanout accordingly. Thus, as the system always fully operated at its limit, we fully expected the system to maintain a stable throughput even with an increasing number of processes.

Finally, we show how the system can maintain that high throughput independent of the number of processes by maintaining a stable fanout m and adjusting the tree sizes accordingly. The results of these experiments are shown in Figure 10(b). In this context, there is a tree of depth two in the case of 100 processes and a tree of depth three, for the executions from 200 to 800 processes (considering the fanout m = 10, a tree of depth two might have atmost 111 processes and a tree of depth three up to 1110 processes).

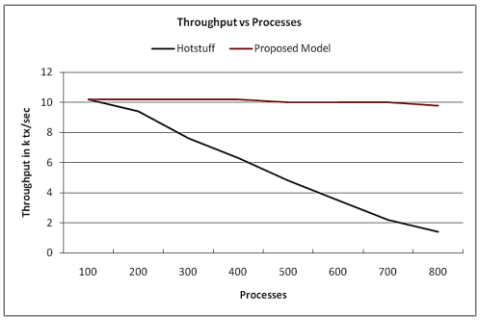

Figure 10(b): System Throughput with stable fanout

CONCLUSION

In this paper, we argue that for a consensus model to be suitable for IoT, it needs to have low latency, high throughput and enhanced scalability which necessitated the development of better consensus model, an improved *HotStuff* which is a two-chain consensus protocol with efficient and simplified view change. It achieves consensus in two rounds of communication. Furthermore, the model has low latency, enhanced scalability and is robust against forking attack better than HotStuff. Our experimental results show that the proposed model has better scalability and outperforms *HotStuff* in terms of latency. The proposed model also outperforms *HotStuff* in terms of latency and throughput under forking attacks. This shows that the model is

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria

suitable for IoT networks and can provide the networks with immunity to plausible attacks.

REFERENCES

Bhattacharjee, S., Mehrdad S., Mainak C., Kevin K., and Charles K." Preserving Data Integrity in IoT Networks Under Opportunistic Data Manipulation." In Dependable, Autonomic and Secure Computing (DASC), 2017 IEEE 15th Intl, P. 446-453.

Debus, J. (2017). Consensus methods in blockchain systems." Frankfurt School of Finance & Management, Blockchain Center, Tech. Rep

Rahnama, S., Gupta, S., Qadah, T. M., Hellings, J., and Sadoghi, M. 2020. "Scalable, Resilient, and

Configurable Permissioned Blockchain Fabric". In: *Proc. VLDB Endow.* 13.12 (Aug. 2020), P. 2893–2896.

Abraham, I., Gueta, G., and Malkhi, D. 2018. "Hot-Stuff the Linear, Optimal-Resilience, One- Message BFT Devil". In: *CoRR* abs/1803.05069. arXiv: 1803.05069. url: http://arxiv.org/abs/1803.05069.

Liu, Y., Liu, J., Zhang, Z., Xu, T., and Yu, H. (2019). Overview on consensus mechanism of blockchain technology, J. Cryptol. Res. 6 (4) (2019) P. 395–432.

Yin, M., Malkhi, D., Reiter, M. K., Golan-Gueta,G., and Abraham, I. 2019. HotStuff: BFT consensus with linearity and responsiveness, in: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, on, Canada, July 29 - August 2, 2019, 2019, P. 347–356.

Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P. A secure sharding protocol for open blockchains, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016, 2016, P. 17–30.

Corresponding author: S. K. Alaramma
skalarammane@gmail.com
Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria