Revue d'Intelligence Artificielle
Vol. 37, No. 6, December, 2023, pp. 1607-1616
Journal homepage: http://iieta.org/journals/ria

Fuzzy Inference System for Byzantine Fault Tolerance in IoT Security

Sundareswaran Natarajan[1*], Sasirekha Selvakumar[2]

[1] Department of Computer Science and Engineering, Kalasalingam Academy of Research and Education, Krishnankoil, Srivilliputtur 626126, Tamilnadu, India
[2] Department of Information Technology, Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam, Chennai 603110 Tamilnadu, India

Corresponding Author Email: vethasundares@gmail.com

https://doi.org/10.18280/ria.370625

Received: 12 June 2023
Revised: 1 September 2023
Accepted: 9 October 2023
Available online: 27 December 2023

Keywords:
byzantine attacks, byzantine fault tolerance, behavior analysis, fuzzy system, Internet of Things, security

ABSTRACT

The Internet of Things (IoT) is progressing rapidly, transforming the way people interact with the world through improved connectivity. Nevertheless, as the number of devices surges, security has become a primary concern. Through a comprehensive review of literature, it has been identified that byzantine targets the physical layer of IoT systems. This attack is carried out when a compromised device spreads malicious information to other nodes in the network, leading to potentially compromising the entire system. To address this issue, this work introduces a Fuzzy-based Byzantine Fault Tolerance (F-BFT) mechanism derived from a Type-1 fuzzy system require less computational power and resources compared to complex fuzzy systems, this can be advantageous in IoT systems where processing capabilities are limited. Based on measurements of modeling error at roughly σ=0.05 and an accuracy rate of 99.5%, recall of 98.89%, and F-Score of 98.83%, it has been observed from the results that the type-1 fuzzy model is highly precise. When compared to the existing system, the average delay of the F-BFT detection algorithm was 4.7 % decreased, average throughput was 5.0 % (increased), and average communication complexity was reduced from $O(m^2)$ to $O(m)$ where m is the number of nodes in the network.

1. INTRODUCTION

As more devices become connected to the internet, the potential attack surface for hackers increases, making IoT security a critical concern for individuals and organization alike [1]. IoT security refers to the measures and practices put in place to protect IoT devices and networks from cyber threats. One of the main challenges in IoT security is that many IoT devices are designed to be low-cost and low-power, and may not have robust security features built in. This can make them vulnerable to attacks such as malware infections, data breaches, and Distributed Denial of Service (DoS) attacks [2]. Moreover, the diversity of devices and systems used in IoT requires a multi-layered approach to security, as different devices and applications require different types of protection. Hence, it is essential to take steps to ensure their security and protect against potential cyber threats. Some IoT devices, such as medical devices and industrial control systems, can directly impact physical safety. Implementing IoT device security ensured that these devices are secure and not vulnerable to cyber attacks that could cause physical harm. When the security of devices are compromised, also known as byzantine faults, they can be grouped together to form botnets, which can be used to launch DDoS (Distributed Denial of Service) attacks. These attacks can bring down entire IoT networks and causing financial loss. Detecting byzantine fault, also known as byzantine intrusion in an IoT network can be challenging, but there are some strategies that can be employed such as trust based systems, consensus algorithms, and intrusion detection systems in which trust based systems monitor the inconsistent behaviour of devices or nodes, it may be the sign of a byzantine intrusion [3]. Therefore, in this work, a behaviour-based analysis model to detect and eliminate byzantine intrusions is primarily discussed as a Byzantine Fault Tolerance (BFT) solution. The short introduction about the proposed work is discussed as follows.

The device security was thoroughly examined, focusing on the security measures for nodes or devices, and ensuring secure communication between them. The objective of this research was to examine the issue of the byzantine fault in the IoT system. To address this problem, a fault-tolerant mechanism was proposed by implementing F-BFT, which employed a strategy for byzantine detection and elimination. The study also involved analyzing the dynamic behavior of nodes, identifying byzantine nodes through a type-1 fuzzy system, and utilizing backup devices with automated failover mechanisms to maintain reliable operation despite the presence of disruptions or failures. Fuzzy logic systems are used in a wide range of applications, including control systems, decision-making, pattern recognition, and intrusion detection system. In many real-world situations, data is incomplete, uncertain, or imprecise. Fuzzy logic can handle this

uncertainty by allowing for partial truth values, which make it useful in situations where there is ambiguity or imprecision in the input data. It can be more flexible than traditional logic systems, which use binary true/false outputs. Fuzzy systems can adjust the degree of output based on the degree of input, which allows for a more precise response [4]. Hence, fuzzy logic can provide more robust, efficient, and accurate solutions in a wide range of applications. In this F-BFT, a type-1 fuzzy system-based algorithm is designed to detect compromised devices by analyzing device's behavior and communication pattern. The output of the fuzzy system is a binary value that indicates whether the device is compromised or not. These compromised devices can harm the network operations. Hence, this work applies automated failover mechanism for the fault tolerance. It is discussed as follows.

An automated failover mechanism is a system that automatically switches to a backup system or component in the event of a failure. (Kostrzewa & Ernst 2020) This mechanism is used in critical systems where downtime is not an option, such as in data centers, hospitals, and financial institutions. The need for an automated failover mechanism arises from the potential impact of system downtime. It can help minimize the impact of downtime by quickly switching to a backup system or component when a failure is detected. This mechanism can also reduce the need for manual intervention, which can save time and reduce the risk of human error. By having a backup system in place, organizations can ensure that their systems are always available and that they can quickly recover from any failure. In this F-BFT, backup devices are redundant components that can take over the function of disconnected devices in the event of faults (caused by an attacker) through automated failover mechanisms. Moreover, this work has obtained the following results from the simulations.

Based on the measurements of modeling error at approximately σ=0.05 and the accuracy rate of 99.5%, it can be inferred that the type-1 fuzzy model used to forecast byzantines in an IoT system is highly accurate. Moreover, this study illustrated the effectiveness of a fault-tolerant solution by comparing the average delay, average throughput, and communication complexity of Practical Byzantine Fault Tolerance (PBFT) and Scalable Tree based Byzantine Fault Tolerance (STBFT) systems. The simulations were implemented using OMNET ++ to demonstrate BFT by comparisons of various performance metrics with current works. In this work, the topology was designed by setting the participating nodes, routers, and links to connect as a decentralized graph between nodes and gateways, configured with a bandwidth, a delay, and a queue. The simulation results are described in the results and discussions section.

1.1 Motivation

1. The IoT network allows its components such as devices/sensors, data processing on the cloud, and user interface to communicate on an open public network. It allows a cyber-attacker to deploy malicious or byzantine nodes inside the network.
2. The scalability of IoT networks poses a challenge in distinguishing between legitimate and malicious nodes mostly in real-time networks, and is also an important task.
3. IoT devices are resource constrained in terms of storage and battery that can be physically compromised by an adversary to act as byzantine that has the right with false identity to access the existing resources of the IoT network.

Therefore, it is of utmost importance for an IoT network to eliminate byzantine intrusion to provide effective services to smart applications.

1.2 Contributions

The major research contributions of this work include detecting and eliminating byzantine nodes through a type-1 fuzzy system-based algorithm (a dynamic behavior-based analysis model).

Evaluating various experiment metrics, such as average delay, average throughput, and communication complexity are compared against existing solutions.

Discussing the security advantages of F-BFT through theoretical analysis.

The average delay and average throughput of this work (Average probability of faulty nodes and Average proportions of faulty nodes) were compared with existing works and proved that the proposed F-BFT obtained a smaller delay and higher throughput than the existing systems because of its type-1 fuzzy system based detection operation. This helps to make this system faster identification of byzantine nodes in the network than those of existing systems.

1.3 Paper organization

The remainder of the study is structured as follows: Section 2 details the related works and existing solutions, Section 3, discusses the proposed byzantine node detection and disconnecting operations, Section 4 details the experimental results of various scenarios, and proves by comparing the result metrics with existing systems, and finally, conclusions and future work are discussed in Section 5.

2. RELATED WORKS

This comprehensive review delves into the security of IoT devices and nodes. When it comes to the security of devices, byzantine faults can cause significant issues such as network outages and financial losses. This section explores a series of solutions for byzantine fault tolerance.

Driscoll et al. [5] suggested analyzing the byzantine fault that led to the simultaneous failures of a group of peers over duration of 0.5×10^{-12} and demonstrated that these failures were not random but were instead due to byzantine intrusion. They proposed that Time-Triggered Node Architecture (TTNA) could be utilized to filter out these byzantine failures in a distributed environment. However, the turn around time to identify the byzantine by TTNA was high. Rahli et al. [6] introduced Byzantine Fault Tolerance (BFT) state machine replication to achieve adequate fault tolerance in a distributed system. To detect the presence of byzantine nodes, they developed the "happened before" theoretical relation and utilized it to update the intrusion detection signature system for future use. They also designed a new Practical Byzantine Fault Tolerance (PBFT) protocol for this purpose, and analyzed its correctness mathematically using various lemmas related to message exchange authentication between peers. However, this method has taken only PBFT for comparisons to prove the efficiency. The byzantine detection and tolerance algorithm presented by Sousa et al. [7] is designed to run on a specific group of nodes, thereby minimizing the overhead

associated with executing Byzantine Fault Tolerance (BFT) on all nodes within the network. This approach effectively addresses the performance bottleneck in byzantine detection operations in the Blockchain (BC) network. However, it is important to note that this algorithm may not be well-suited for complex IoT networks.

In their study, Zhang et al. [8] presented a behavior-based analysis approach to detect byzantine nodes by monitoring message exchanges among peers. This research employed an event detector to monitor message transmissions within its distributed network system, with the objective of ensuring that each message was sent and received only once by a single peer in the network. However, it is important to note that analyzing all communication messages may impact the system's performance.

In their research, Thai et al. [9] introduced the Hierarchical Byzantine Fault Tolerance (HBFT) protocol for wide area networks with scalability in mind. In this work, an experiment was carried out on a network of 475 nodes, which were partitioned into 19 groups, with each group consisting of four byzantine nodes. The study measured and compared the message throughput performance of the PBFT protocol. However, the study did not discuss the byzantine node detection rate.

Bynun et al. [10] introduced a byzantine self-stabilization algorithm. This algorithm uses a technique that involves setting the maximum values of timestamps, message counter, and clock synchronization to recover from such faults. However, it should be noted that while the algorithm is capable of stabilizing the system, it is not effective against malicious intrusions and their related malfunctions

Vijaykumar et al. [11] presented a Deep Neural Network (DNN) approach to identify the most effective machine learning algorithm for detecting and controlling byzantine intrusion. The study found that the byzantine intrusion detection rate for the learning rate success ratio was 10%, and the accuracy rates were as high as 45% after 1000 epochs of learning units. However, the process of analyzing information to implement the DNN approach increased the execution time, which could potentially impact the network's overall performance.

Yu et al. [12] introduced the Byzantine Fault Tolerance based on dual administrator short Group signatures (GPBFT), a practical byzantine fault tolerant consensus algorithm that utilizes a dual administrator short group signatures method to detect malicious nodes in the network. The experimental results demonstrate that the GPBFT algorithm can effectively reduce the communication overhead, minimize consensus delay, and significantly enhance both efficiency and security when compared to the PBFT algorithm. However, the study did not address the complexity of the GPBFT implementation.

Almseidin and Alkasassbeh [13] presented a precise approach for intrusion detection in IoT networks, which employed a fuzzy-based interpolation reasoning method. The proposed method was specifically designed to detect IoT Botnet attacks using a dataset of such attacks. The results of the experiments showed that the proposed approach achieved a detection rate of 96.4%, which is highly accurate, while also effectively reducing the false positive rate. Palanikumar et al. [14] presented a PBFT-based Smart Contract (SC) algorithm aimed at securing real-time Internet of Medical Things (IoMT) data. The proposed system utilizes PBFT SC consensus protocols in combination with proof-of-work to enhance the privacy and availability of healthcare data, enable decentralized access, and increase the flexibility of data in the blockchain network.

Table 1. Literature survey analysis

Authors	Goals Achieved	Limitations
Driscoll et al. [5]	G2 and G3	TTNA has not addressed reliability (G1) and security (G4) goals due to the distributed network topology as any peer can act as a coordinator.
Rahli et al. [6]	G1, G2, and G3	State machine replication is prone to cyber-attack when the server is compromised. Hence, it has not been achieved the security (G4).
Sousa et al. [7]	G1 and G4	As this method executes BFT only on a certain group of nodes, the remaining nodes in the group may not be available (G3) and prone to byzantine attacks (G4).
Zhang et al. [8]	G1 and G2	The message communication between peers is prone to man-in-the-middle attacks (G4). As the message may not reach the destination, the system cannot withstand on faults. (G3)
Thai et al. [9]	G2 and G4	The system is not reliable (G1) when it scales high and the new joined node may not get the BFT in place. Hence, it cannot withstand (G3) on byzantine faults.
Bynum et al. [10]	G1 and G2	When the self-stabilization algorithm is compromised (G4) or redirected by an attacker, the system cannot withstand (G3) from the byzantine faults.
Vijaykumar et al. [11]	G1, G2, and G3	The DNN approach can identify the efficient byzantine algorithms but the security (G4) of the chosen byzantine algorithms was not proved in the application.
Yu et al. [12]	G2, G3, and G4	As GPBFT follows the group signature based approach, the system is not reliable (G1) due to the public keys involved in the group signature.
Almseiden and Alkasassbeh [13]	G1, G2, and G4	The system can accurately detect bots. However, it could not identify the malware intrusion to compromise nodes. Hence, the system cannot tolerate (G3) faults.
Palanikumar et al. [14]	G2, G3, and G4	As the system uses smart contract (SC) based PBFT model of the blockchain system. It may not be reliable (G1) due to decentralized nature of the blockchain as any node can control the operations.
Jiang et al. [15]	G1, G3, and G4	The STBFT falls short of fault tolerance when the number of participating node is high. Hence, the system may not be available to the legitimate users.

Jiang et al. [15] presented a solution to the issues faced by the PBFT algorithm in blockchain networks, such as high communication overhead and scalability bottlenecks. The proposed Scalable Tree-based Byzantine Fault Tolerance (STBFT) algorithm utilizes a tree topology network structure that organizes network nodes into groups and layers, resulting in a hierarchical multi-centralized consensus approach that reduces communication complexity and improves node scalability. The authors of this paper conducted simulation experiments using k=4, 10, and 16, where k represents the suspected byzantine node, to compare the communication complexity ratio of the STBFT algorithm to the PBFT

algorithm for different network sizes with fewer than 1000 nodes. The results of the experiments show that the STBFT algorithm has significantly lower communication complexity than the PBFT algorithm, regardless of the value of k. However, it is essential to note that the STBFT algorithm falls short of fault tolerance when the number of faulty nodes in the system is fixed. While the algorithm is highly scalable, many application scenarios require not only scalability but also high fault tolerance. Therefore, additional measures may be necessary to meet the high fault tolerance requirements of these scenarios.

2.1 Survey analysis

The goals of fault tolerance in IoT systems are to ensure the reliability, availability, resilience, and security of the interconnected devices and networks.

Reliability - Fault tolerance aims to improve the reliability of IoT systems by minimizing the probability of failures.

Availability - IoT systems often need to be available to provide continuous services and functionality.

Resilience - Resilience focuses on the ability of IoT systems to withstand and recover from faults or failures.

Security - Fault tolerance is closely linked to security in IoT systems. Ensuring fault tolerance helps mitigate potential security breaches or attacks that may exploit vulnerabilities in the system.

The survey analysis aids in comprehending the authors' objectives in their articles concerning the pursuit of goals (G) such as, G1– Reliability; G2- Availability; G3- Resilience; G4- Security, as shown in Table 1. Moreover, based on the literature survey mentioned below, a notable observation is that many researches have not satisfied all those important goals (G1 to G4) in their study. Hence, this research offered a distinct contribution to byzantine detection and deactivation by introducing novel fuzzy based fault detection method. This research uncovered new prospects for the cryptographically secured IoT networks. The findings of this study signifies a breakthrough in the understanding of securely managing byzantine behaviors in the network.

3. FUZZY BASED BYZANTINE FAULT TOLERANCE (F-BFT)

Yuan et al. [16] BFT refers to a system's ability to function correctly in the presence of faulty or malicious components. In the context of IoT, BFT is essential for ensuring the reliability and security of IoT systems. In IoT, BFT can be achieved by implementing redundancy or automated failover mechanism in the system. Moreover, BFT is critical in IoT because many IoT systems are deployed in mission-critical applications, such as healthcare, transportation, and industry automation. In these applications, any failure or security breach can have serious consequences, and BFT can help ensure that the system continues to function correctly even in the face of such events. Therefore, fault tolerance in F-BFT is proposed through a type-1 fuzzy system-based algorithm to detect compromised devices or byzantines.

Type-1 fuzzy system is used in this research for several reasons, despite the development of more advanced fuzzy logic techniques like type-2 fuzzy systems and other machine learning approaches. This fuzzy system is computationally less demanding compared to more complex techniques like type-2 fuzzy systems. This can be advantageous in a resource-constrained environments are a concern (IoT). It can handle uncertainty and imprecision in data or behavior without introducing excessive complexity [17]. In this research, the inherent uncertainty and vagueness present in the behavior analysis are effectively modeled using type-1 fuzzy systems. In this case, using a more complex approach might not necessarily lead to better results. While more advanced techniques (Machine learning and deep learning) might offer increased expressive power, the trade-off between complexity and performance needs to be carefully considered. Type-1 fuzzy systems might strike the right balance between them for many applications.

Hence, in this work, byzantine nodes are identified from the network by executing type-1 fuzzy system-based algorithm through monitoring the behavior of the devices or nodes in the network. The anomalous behaviours of devices are monitored by concerning its message communication factors, such as requesting, responding, and delaying factors. After detecting byzantines, they are eliminated or disconnected from the network. Finally, backup devices are used to take over the function of disconnected devices. This can be done through automated failover mechanisms. These operations of byzantine fault tolerance (Figure 1) is developed and detailed in the following section.

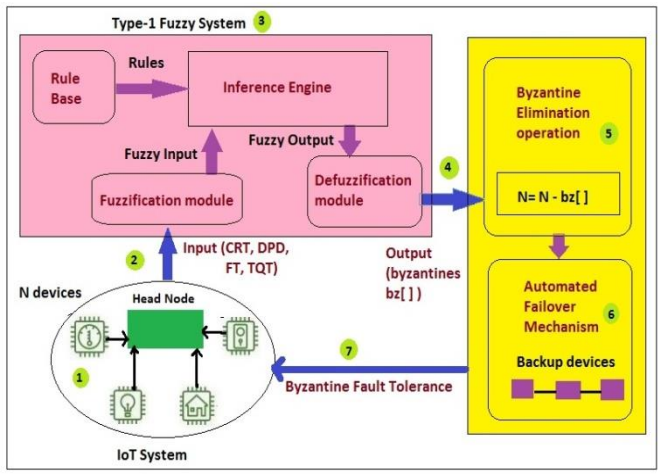

Figure 1. Byzantine fault tolerance system

3.1 Time-based anomalous behavior analysis

In IoT systems, devices or nodes communicate data and events at regular intervals, any deviation from the expected timing or frequency can indicate a potential attack or malicious activity [18]. In this F-BFT, time-based anomalous analysis involves monitoring the Communication Response Time (CRT), Data Processing Delay (DPD), Forwarding Time (FT) and computing Total Quality Time (TQT) of devices to identify any anomalies or deviations from the expected patterns. For example, in a smart home system, if a motion sensor triggers an alarm every 10 seconds, and suddenly the sensor starts triggering alarms every 2 seconds, it could indicate that the sensor has been hacked or compromised. This F-BFT can detect such anomalies of byzantine and alert the system to disconnect such devices or take other corrective actions to prevent any further damage. The individual behavior-based computation is calculated as follows.

The Communication Response Time (CRT) is reflected in the entire IoT network process. CRT is computed using the

data processing time, distance between nodes, and bandwidth (Eq. (1)). Hence, it directly corresponds to justifying the byzantine node. Data Processing Delay (DPD) helps identify byzantine nodes and is used to count the total network processing time. In the case of an IoT network, the delay of a particular node significantly deviates from that of other legitimate nodes. DPD is computed by the actual time taken to reach a node and process the data in this node (Eq. (2)). The time taken to forward data Forwarding Time (FT) from the source node to the destination node can be used to justify the integrity of the actual data. This helps identify the byzantine behavior of the node. FT is computed as the time from the source node to the destination node (Eq. (3)). The calculations of these time factors are listed in Eqs. (1), (2), and (3).

$$CRT=\frac{DPt}{BW}+\frac{Dbn}{S}+\frac{Dd}{T} \quad (1)$$

where, DPt – Data Processing time, Dbn – number of Data bits, Dd – distance of the Destination node, BW – Bandwidth, S – Speed, T-Time interval.

$$DPD=\frac{RS}{Tr}+\frac{PT}{Tp} \quad (2)$$

where, RS – Actual time taken to reach a node or device from a sensor, Tr – Normal time to reach a node, PT – Actual time taken to process data in a node, Tp – Normal time taken to process data in a node.

$$FT=\frac{(ST,size)}{(ET,size)} \quad (3)$$

where, ST – Starting time of source node, ET – Ending time of destination node, size – Amount of data, size determines any change in the data from the source to destination.

TQT can be calculated by using CRT, DPD, and FT as shown in Eq. (4).

$$TQT=k1*CRT+k2*DPD*k3*FT \quad (4)$$

where, k1, k2, and k3 are the variables for a certain time scenario. By default k1=k2=k3=1. TQT is particularly useful to confirm byzantines.

3.1.1 Type-1 fuzzy system

A type-1 fuzzy system-based algorithm is designed to detect compromised devices (byzantines) by analyzing device's behavior and communication pattern such as CRT, DPD, FT, and TQT. The following steps are followed to design this algorithm.

The input variables to the fuzzy system are different parameters such as CRT, DPD, FT, and TQT that are indicative of a compromised device.

The output variable of the fuzzy system is a binary value that indicates whether the device is compromised or not.

The membership functions are used to map the input variables to fuzzy values. These functions are designed in a way that they capture the vagueness of the input variables (large deviations of CRT, DPD, FT, and TQT). The design of member functions is defined as follows.

Membership function for a fuzzy set of anomalous behavior (large deviations CRT, DPD, and FT).

Let M be the universe of discourse for anomalous behaviour, and let m be an element of M. Then the membership function for a fuzzy set A that represents anomalous behaviour can be defined using the Eq. (5).

$$\mu\_A(m)= \min(\max((m-a)/(b-a),0),1) \quad (5)$$

where, a and b are the lower and upper bounds, respectively, of the range of anomalous behaviour that A represents. This function maps m to a value between 0 and 1, indicating the degree to which m belongs to the fuzzy set A.

Membership functions for a fuzzy set of anomalous TQT value.

Let N be the universe of discourse for anomalous TQT, and n be an element of N. Then the membership function for a fuzzy set B that represents anomalous TQT can be defined using the Eq. (6).

$$\mu\_B(n)=\min(\max((n-c)/(d-c),0),1) \quad (6)$$

where, c and d are the lower and upper bounds, respectively, of the range of anomalous behaviour that B represents. This function maps n to a value between 0 and 1, indicating the degree to which n belongs to the fuzzy set B.

Fuzzy rules are used to map the fuzzy input variables to the fuzzy output variable. The fuzzy rules are designed based on the expert knowledge of the characteristics of a compromised device.

The ANFIS (Adaptive Neuro-Fuzzy Inference System) classification model is used for the classification of byzantines in the system. It is a hybrid computational model that combines the principles of fuzzy logic and neural networks to create a powerful system for solving complex problems.

ANFIS aims to adaptively tune the parameters of both the fuzzy inference system and the neural network to provide accurate and flexible modeling of relationships between inputs and outputs.

The rules are listed below.

Rule 1: if x is A_1 and Y is B_1 then $f_1=a_1x+b_1y+r_1$
Rule 2: if x is A_2 and Y is B_2 then $f_2=a_2x+b_2y+r_2$

where, x and y inputs, the fuzzy sets are B_i and A_i, a_i, b_i, r_i indicates the parameter that is represented in training and f_i is the output defined by the fuzzy rule.

Each device or node i are assigned with following node function.

$$D_i=\mu_A(x)$$

where, x denotes the input to device i, μ_A denotes the membership function of A.

The membership function can be defined as follows.

$$\mu_A(x)=\exp(-(\frac{x-m}{a})^2)$$

where, a_i and m_i denote the principle parameter set and x indicates the input.

Similarly,

$$\mu_B(x)=\exp(-(\frac{y-n}{b})^2)$$

These rules can be normalized with the devices and marked as N.

$$N_i=\frac{\omega_i}{\omega_{1+\omega_2}}$$

where, ω_i denotes the output parameter. It can be defined as follows.

$$\text{Output}=\sum \omega_i.f=\frac{\sum \omega_i.f}{\sum \omega_i}$$

Defuzzification is used to convert the fuzzy output variable into a crisp binary value that indicates whether the device is compromised or not. Here the rule for combining fuzzy sets of A and B.

Let R be a fuzzy relation that combines A and B. Then R can be defined using the Eq. (7).

$$R(m,n)=\min(\mu\_A(m), \mu\_B(n)) \quad (7)$$

This function maps pair of elements from M and N to values between 0 and 1, indicating that degree to which they belong to the fuzzy relation R. It represents the degree of compatibility between A and B, and can be used to identify potential byzantines.

Therefore, in this F-BFT, a fuzzy system-based algorithm is used as an effective approach to detect compromised devices or byzantines.

3.1.2 Byzantine detection and elimination

In this F-BFT, byzantine node detection through type-1 fuzzy system based algorithm was performed as a behavior-based distributed computation algorithm (Algorithm1) without relying on a central server. However, there are head nodes (H) in this distributed environment (Network Topology) which are collaborate and work independently to complete this task.

Algorithm 1: Byzantine nodes detection using type-1fuzzy system.

Input: Dynamic responses of devices
Output: List of byzantine nodes
for each N (n_1,n_2.....n_m) at interval 't'
  H compute CRT, DPD, FT, TQT
 Call fuzzy( )
end for
function fuzzy ( )
start
  for (i=1; i<N; i++) do
  $\omega_i=\mu_A(x).\ \mu_B(y)$
    $\omega_i.f=\omega_i\ (m_ix+n_iy+r_i)$
     $\omega(x,y)=m(i)+n(i)$
done
$return\ \sum \omega_i.f=\frac{\sum \omega_i.f}{\sum \omega_i}$
end
    send→ msg ("byzant", bz[ ], H)

N, participating devices in the network; CRT, communication response time; DPD, data processing delay; FT, forwarding time; TQT, total quality time; H, Head node; mem, member function; R, relation; msg, message; bz[ ], list of byzantines.

After computing byzantines, H initiates operation to disconnect the byzantines from the network (Algorithm 2). In this, H executes Algorithm 2, receives the list of byzantines bz[ ] and disconnects those devices from the network.

Algorithm 2: Disconnecting byzantines from the network.

Input: Byzantines
Output: Byzantine-free network
for each time interval 't'
H call **disconnect** (bz[ ])
end for
function disconnect (bz[ ])
start
    call **Remove** (bz[1],bz[2]....bz[n_m])
    send→ msg ("disconnected", bz[ ], **H**)
end

msg, message; n, participating node in the network; bz[ ], list of byzantine nodes.

Disconnecting byzantine nodes from the network ensures the security and integrity of the network. For each time interval 't', the byzantine nodes are automatically disconnected from the network through the head node H. Once byzantine nodes are identified, isolate these nodes from the rest of the network can prevent further harm. This can prevent the malicious activity from spreading to other parts of the network. We implemented continous monitoring of byzantine nodes and disconnecting mechanism to quickly identify and respond to any future malicious activities. The logic associated with byzantine disconnecting mechanism is implemented as continuous monitoring (bz[ ] - arrays of byzantine nodes) to remove them from the rest of the network.

3.1.3 Automated failover mechanism

In general, fault tolerance in the context of the IoT refers to the ability of an IoT system to continue operating correctly and reliably even in the presence of failures or disruptions. In this work, BFT is designed as a self-healing mechanism to automatically detect and eliminate byzantines without human intervention. Hence, systems can continue operating even when some systems disconnected due to byzantines through the use of backup devices. Backup devices are redundant components that can take over the function of disconnected devices in the event of faults. This can be done through automated failover mechanisms, where the system automatically switches to the backup devices in the presence of byzantine faults.The most commonly used BFT method for distributed IoT system is PBFT (Practical Byzantine Fault Tolerance). To ensure fault tolerance in PBFT, it requires that at least two-thirds of the nodes in the system are not compromised and follow the protocol correctly. This ensures that any byzantine nodes cannot cause the system to fail or produce inconsistent results. In the results and discussion section, the proposed BFT has been compared with PBFT and similar current works to prove the efficiency of the proposed BFT. Furthermore, the performance of the fuzzy system was evaluated using a random sample inputs (CRT, DPD, FT, TQT) of 1000 nodes to assess the classification of fuzzy models (Discrete Time (DT), Continuous Time (CT) or Non-Continuous Time (NCT)), error rate and accuracy. These metrics provided a comprehensive view of the performance of the fuzzy system and used to fine tune the system for better performance.

4. RESULTS AND DISCUSSIONS

The simulations were implemented using OMNET++ to

design the Internet of Things (IoT) environments established with 200 to 1000 nodes as input. The communication setup, including bandwidth, packets, links to the network, baud rate, and other specifications was established. In this experiment, the topology was designed by setting the 1000 nodes, three routers, and nearly 2000 links to connect as a decentralized graph between nodes and gateways, configured with a 20-Mbps bandwidth, a 100 ms delay, and a 2GB capacity queue. The fuzzy logic system was implemented as an object (C++ version) in OMNET++ to process input, computation, and conclusion. The simulation results described in this section were performed in an AMD A10-PRO 7800B R7, 12 computer cores 4C+8G 3.50 GHz, 8-GB of random access memory (RAM) computer, and the operating system was Ubuntu 22.04.3 release. The experiments were evaluated by comparing the PBFT and STBFT methods for various parameters such as the average delay, average throughput, and communication complexity. Here, the time based type-1 fuzzy system for detecting and disconnecting byzantine nodes caused some delay in the process. Throughput is the rate at which transactions (detecting and disconnecting) are confirmed. Communication complexity is an algorithmic complexity used to evaluate the process of detecting and disconnecting operations. The mathematical model of type-1 fuzzy system is followed to represent membership functions, inference systems, input and output functions, accuracy, error rates, and conclusions in a more generic form.

4.1 Mathematical results of predicting byzantines using type-1 fuzzy system

It is possible to model the fuzzy system without datasets to confirm byzantines when actual results or datasets are not available in advance. The foresight model can be determined to find the accuracy for fuzzy logic system. Modeling of prediction process for dynamic anomalous behavior of certain devices is characterized by uncertainty. The accuracy of such a model will never be absolute, although the prediction is based on iterative computation of lower and upper bound to confirm byzantines. The common way of determining accuracy is to compare modeling results with actual data. In the case of dynamic behavior analysis, these data only be obtained after a fixed time period of upper bound computation for which prediction is carried out.

Mathematical model of Membership Function (MF) used in type-1 fuzzy system can be categorized as singleton, gaussion, and Z(S)-shaped forms as shown in Table 2.

Table 2. Mathematical model of membership function

Name	Mathematical Form	Remarks
Singleton	$\{1, x=a \mid 0, x=a\}$	a is the number that defines fuzzy set.
Gaussian	$\exp\left(\frac{(x-b)^2}{2c^2}\right)$	b is a coordinate of a function maximum, c is the concentration coefficient.
Z(S)-Shaped	$\mu=\frac{1}{1+e^{-a(x-b)}}$	For a>0, the function is S-shaped one, for a<0, the function is Z-shaped one, b is a coordinate for μ=0.5.

Since the inputs (CRT, DPD, FT, and TQT) for MF of the F-BFT are in discrete form, singleton function is suitable to generate output classification (byzantine (1) or not (0)). The other two functions (Gaussian and Z(S)-Shaped) are used for continuous input formats of applications.

This MF works on the basis of Mamdani fuzzy inference system (Max-Min inference method) to reach a particular conclusion (byzantine or not) based on some evidence (upper bound or threshold) associated with a logic.

In the case of Max-Min inference method, the following rules can be applied to confirm byzantines.

Rule 1: IF x_1 is A_1^1 and x_2 is A_2^1 THEN y^1 is B^1

In this rule, x denotes the CRT, DPD, FT inputs, A denotes the upper bound or threshold and y denotes the output 1 or 0.

Rule 2: IF x_1 is A_1^2 and x_2 is A_2^2 THEN y^2 is B^2

In this rule, x denotes the TQT input, A denotes the upper bound or threshold and y denotes the output 1 or 0.

The conclusion with the fuzzy model results can be classified, namely, disruptive (DT), critical (CT), and non-critical (NCT). This conclusion limits are determined by the method of equidistant points (Eq. (8)).

$$P=\sqrt{(x_1 - x_2)^2 + (y^1 - y^2)^2} \tag{8}$$

This model finds equidistant points to determine a threshold value. However, this approach will vary for multiple evaluations (iterative). Therefore, it is reasonable to use certain limits that do not change over multiple evaluations. In this case, the Harrington desirability function can be applied as shown in the following Eq. (9).

$$f(x)=\exp(-\exp(-x)) \tag{9}$$

The lower threshold is critically 0.37, and the upper threshold is 0.8. This model defines that when criticality level exceeds 0.8 as disruptive.

4.1.1 Determining the accuracy

The developed BFT model performs classification, i.e., an assessment of the input data to determine byzantine or not (1 or 0). The type-1 fuzzy model will estimate in the form of different classes such as disruptive, critical, and non-critical. Accuracy of this model is determined by the root-mean-square error of the type-1 fuzzy model to confirm byzantines. The acceptable error rate is only 5% for this modeling. Mean square error of model is determined using the Eq. (10).

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \Delta_i{}^2} \tag{10}$$

where, N is the total number of measurements and i is the current measurement.

In the testing phase, the testing inputs such as CRT, DPD, FT, and TQT are given to the fuzzy system, which classify the input as a byzantine or not. The obtained result is then used to compute accuracy of the fuzzy system. The accuracy of the system is computed based on the definitions, namely, precision, recall, and F-measure. They are defined using the Eq. (11).

$$\text{precision}=\frac{tp}{tp+fp}, \text{ recall}=\frac{tp}{tp+fn}, \text{ F-measure}=\frac{(\beta^2+1)(precision-recall)}{\beta^2.precision+recall} \text{ where } \beta=1, \quad \text{Accuracy}=\frac{tp+tn}{tp+tn+fn+fp} \text{ where, tp- true positive, tn-} \tag{11}$$

true negative, fn-false negative, fp-false positive

The following Table 3 (Input class) and Table 4 has listed out a random sample of 1000 nodes to assess the classification of fuzzy models (DT, NCT or CT), error rate and accuracy. For input variables: CRT, DPD, FT, and TQT.

Table 3. Class table

Categories	Class Lables	Class Names	No.of Instances
	C1	CRT	3297
Normal	C2	DPD	2564
Byzantine	C3	FT	9372
	C4	TQT	5060

Table 4. Experimental results of type-1 fuzzy system

Training (1000 Nodes)					Testing (1000 Nodes)	
CRT-ms	DPD-ms	FT-ms	TQT-ms	Category	Error	Accuracy
1.2	3.6	0.4	1.72	NCT	0.004	0.996
1.0	2.8	0.9	2.52	NCT	0.002	0.998
1.3	1.9	0.5	1.23	NCT	0.003	0.997
2.0	1.8	0.6	2.16	NCT	0.009	0.991
1.5	2.6	0.9	3.51	CT	0.006	0.994
1.6	2.0	0.7	2.24	NCT	0.005	0.995
2.2	3.4	0.5	3.74	CT	0.006	0.994
2.1	2.8	0.7	4.11	DT	0.004	0.996
2.0	1.4	0.5	1.4	NCT	0.001	0.999
1.6	1.8	0.9	2.59	NCT	0.006	0.994
2.5	2.0	0.8	4.0	DT	0.002	0.998
3.0	1.8	1.0	5.4	DT	0.005	0.995
2.8	1.8	0.4	2.02	NCT	0.003	0.997
1.6	2.0	0.7	2.24	NCT	0.004	0.996
2.0	1.7	0.9	3.06	CT	0.003	0.997

The following Figure 2 defines the confusion matrix for the total input instances of 20293.

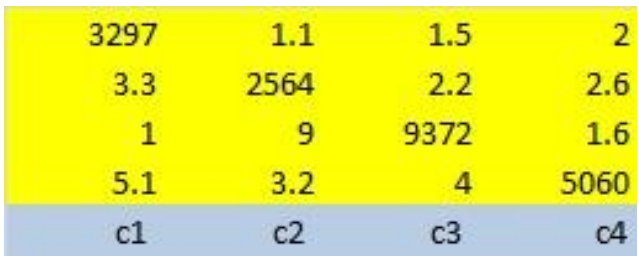

Figure 2. Confusion matrix

Table 4 presents a random set of 15 experimental cases fed to the input of the fuzzy model, as well as the computed criticality scores (DT, CT, NCT) with the corresponding errors and accuracies. From the results of measuring the modeling error approximately (σ=0.05) and the accuracy (0.995), it can be concluded that the type-1 fuzzy model to predict the byzantines in an IoT system foresight is accurate.

The Table 5 has listed out the actual outcome of accuracy, recall, and F-Score for the various classes.

Table 5. Outcome of type-1 fuzzy of various classes

Classes	Accu.	Reca.	F-Score
C1	99.79	98.98	98.17
C2	99.85	99.04	99.12
C3	99.81	98.75	98.97
C4	99.86	99.09	99.06
Average	99.50	98.89	98.83

4.2 Performance assessment of BFT

The average delay and average throughput were measured and compared with PBFT and Scalable Tree-based Byzantine Fault Tolerance (STBFT) for 1000 nodes, as shown in (Figure 3 and Figure 4 respectively).

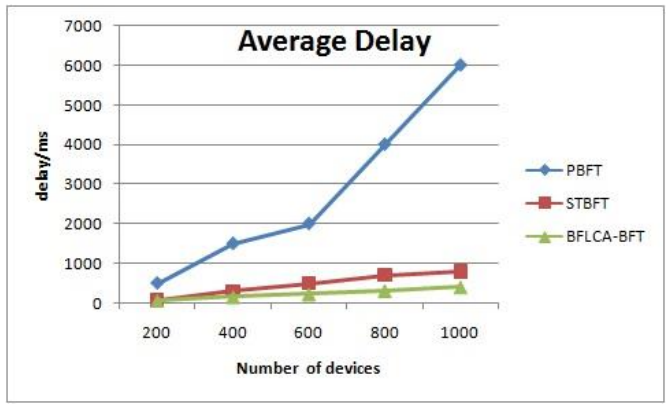

Figure 3. Analysis of average delay

The average delay and average throughput of STBFT (Average probability of faulty nodes and Average proportions of faulty nodes) were 8.48% (decreased) and 8.69% (increased), respectively, compared with those of PBFT. However, the proposed F-BFT obtained a smaller delay and higher throughput than the STBFT because of its type-1 fuzzy system based detection operation.

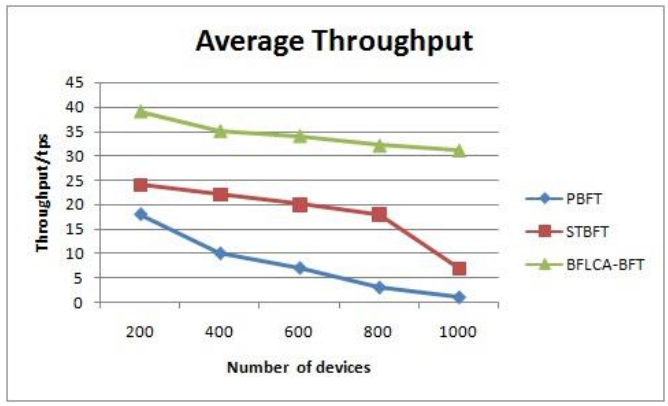

Figure 4. Analysis of average throughput

Here, the average delay of the F-BFT detection algorithm was 4.7% (decreased) and average throughput was 5.0% (increased) when compared to the STBFT. Thus, the proposed F-BFT algorithm was found to be more efficient than STBFT and PBFT.

The performance metrics such as delay and throughput of this work are listed in Table 6 and Table 7 respectively.

Table 6. Comparison of BFT for delay

Total Nodes	PBFT	STBFT	F-BFT
	Delay/ms	Delay/ms	Delay/ms
200	500	70	55
400	1500	300	150
600	2000	500	220
800	4000	700	310
1000	6000	800	400

Table 7. Comparison of BFT for throughput

Total Nodes	PBFT Throughput/tps	STBFT Throughput/tps	F-BFT Throughput/tps
200	18	24	39
400	10	22	35
600	7	20	34
800	3	18	32
1000	1	17	31

An increasing number of devices or nodes in the network increases the delay and decreases throughput. Throughput is the rate at which the network sends or receives data between devices and gateway. When the number of devices that this network reserves for network performance increases, the network throughput increases.

The results were also compared with those of the existing system, which proved that the F-BFT has a smaller delay and higher throughput than the existing system.

The reliability of participating devices in an IoT environment can be extensively increased by establishing a strong security layer (F-BFT) based communication system. Fairness in device selection (H) for computing anomalous behaviors was executed dynamically for the detection process of byzantines. In this F-BFT, the communication complexity was evaluated based on the number of participating devices (m) in the network by which network operations and byzantine node detection were executed. These participating devices (m) were subsets of the network (n). When compared with various BFT methods, the complexity of the F-BFT process was reduced from multiple evaluations to a single evaluation using type-1 fuzzy system, thereby reducing the complexity from $O(m^2)$ to O(m). The detailed comparisons are presented in Table 8.

Table 8. Analysis of reliability, fairness, and communication complexity

BFT Methods	Node Reliability	Node Fairness	Communication Complexity
PBFT	×	×	$O(n^2)$
STBFT	×	✓	$O(m^2)$ where m<n
F-BFT	✓	✓	O(m) where m<n

5. CONCLUSIONS

This work presented a detailed discussion on the security of devices, and investigated the byzantine fault problem in an IoT system, for which the proposed F-BFT implemented a fault tolerance mechanism through a byzantine detection and elimination strategy. This work included a dynamic behavior analysis of nodes by CRT, DPD, FT, and TQT, computing confirmed byzantine nodes through type-1 fuzzy system, and applying automated failover mechanisms. From the results of measuring the modeling error approximately ($\sigma = 0.05$) and the accuracy (99.5%), Recall (98.89%), and F-Score (98.83%).

The average delay and average throughput were measured and compared with PBFT and Scalable Tree-based Byzantine Fault Tolerance (STBFT) for 1000 nodes. The average delay and average throughput of STBFT (Average probability of faulty nodes and Average proportions of faulty nodes) were 8.48% (decreased) and 8.69% (increased), respectively, compared with those of PBFT. However, the proposed F-BFT obtained a smaller delay and higher throughput than the STBFT. Here, the average delay of the F-BFT detection algorithm was 4.7% (decreased) and average throughput was 5.0% (increased) when compared to the STBFT. An increasing number of devices or nodes in the network increases the delay and decreases throughput. In this F-BFT, the communication complexity was evaluated based on the number of participating devices (m) in the network When compared with various BFT methods, the complexity of the F-BFT process was reduced from multiple evaluations to a single evaluation using type-1 fuzzy system, thereby reducing the complexity from $O(m^2)$ to O(m).

While our work focuses on the impact of BFT solution through detection operations, by using a simulation test bed and mathematical design, a significant set of challenges is derived from the deployment of proposed BFT. Indeed, the computational requirements of the BFT approaches might not be satisfied by constrained IoT devices in terms of computing power, energy consumption, and memory. To address such limitations, the use of edge computing by establishing intermediate nodes shall resolve the issues of low resource constraints of IoT devices.

In the future, this work can be further extended by incorporating artificial intelligence techniques to reduce the complexity and dynamic decisions associated with byzantine behaviors of participating nodes in the network.

ACKNOWLEDGMENT

The authors are grateful to all who supported us in producing this article and to those who contributed to this study.

REFERENCES

[1] Abiodun, E.O., Jantan, A., Abiodun, O.I., Arshad, H. (2020). Reinforcing the security of instant messaging systems using an enhanced honey encryption scheme: The case of WhatsApp. Wireless Personal Communications, 112: 2533-2556. https://doi.org/10.1007/s11277-020-07163-y

[2] Williams, P., Dutta, I.K., Daoud, H., Bayoumi, M. (2022). A survey on security in internet of things with a focus on the impact of emerging technologies. Internet of Things, 19: 100564. https://doi.org/10.1016/j.iot.2022.100564

[3] Zawaideh, F., Salamah, M., Al-Bahadili, H. (2017). A fair trust-based malicious node detection and isolation scheme for WSNs. In 2017 2nd International Conference on the Applications of Information Technology in Developing Renewable Energy Processes & Systems (IT-DREPS), Amman, Jordan, pp. 1-6. https://doi.org/10.1109/IT-DREPS.2017.8277813

[4] Ram Prabha, V., Latha, P. (2017). Fuzzy trust protocol for malicious node detection in wireless sensor networks. Wireless Personal Communications, 94(4): 2549-2559. https://doi.org/10.1007/s11277-016-3666-1

[5] Driscoll, K., Hall, B., Sivencrona, H., Zumsteg, P. (2003). Byzantine fault tolerance, from theory to reality. In International Conference on Computer Safety, Reliability, and Security, Edinburgh, UK, pp. 235-248. https://doi.org/10.1007/978-3-540-39878-3_19

[6] Rahli, V., Vukotic, I., Völp, M., Esteves-Verissimo, P. (2018). Velisarios: Byzantine fault-tolerant protocols

powered by Coq. In Programming Languages and Systems: 27th European Symposium on Programming, ESOP 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, pp. 619-650. https://doi.org/10.1007/978-3-319-89884-1_22

[7] Sousa, J., Bessani, A., Vukolic, M. (2018). A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Luxembourg, Luxembourg, pp. 51-58. https://doi.org/10.1109/DSN.2018.00018

[8] Zhang, T., Wang, C., Chandrasena, M.I.U. (2023). Blockchain-assisted data sharing supports deduplication for cloud storage. Connection Science, 35(1): 2174081. https://doi.org/10.1080/09540091.2023.2174081

[9] Thai, Q.T., Yim, J.C., Yoo, T.W., Yoo, H.K., Kwak, J.Y., Kim, S.M. (2019). Hierarchical Byzantine fault-tolerance protocol for permissioned blockchain systems. The Journal of Supercomputing, 75(11): 7337-7365. https://doi.org/10.1007/s11227-019-02939-x

[10] Binun, A., Dolev, S., Hadad, T. (2019). Self-stabilizing byzantine consensus for blockchain: (Brief announcement). In Cyber Security Cryptography and Machine Learning: Third International Symposium, CSCML 2019, Beer-Sheva, Israel, pp. 106-110. https://doi.org/10.1007/978-3-030-20951-3_10

[11] Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A., Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. IEEE Access, 7: 41525-41550. https://doi.org/10.1109/ACCESS.2019.2895334

[12] Yu, X., Qin, J., Chen, P. (2022). GPBFT: A practical byzantine fault-tolerant consensus algorithm based on dual administrator short group signatures. Security and Communication Networks, 2022: 83-94. https://doi.org/10.1155/2022/8311821

[13] Almseidin, M., Alkasassbeh, M. (2022). An accurate detection approach for IoT botnet attacks using interpolation reasoning method. Information, 13(6): 300. https://doi.org/10.3390/info13060300

[14] Palanikkumar, D., Alrasheedi, A.F., Parthasarathi, P., Askar, S.S., Abouhawwash, M. (2023). Hybrid smart contracts for securing IoMT Data. Computer Systems Science and Engineering, 44(1): 457-469. https://doi.org/10.32604/csse.2023.024884

[15] Jiang, W., Wu, X., Song, M., Qin, J., Jia, Z. (2023). A scalable byzantine fault tolerance algorithm based on a tree topology network. IEEE Access, 11: 33509-33519. https://doi.org/10.1109/ACCESS.2023.3264011

[16] Yuan, X., Luo, F., Zeeshan, M., Haider, Chen, Z., et al. (2021). Efficient byzantine consensus mechanism based on reputation in IoT blockchain, Wireless Communication and Mobile Computing, 9(5): 218-232.

[17] Lachouri, C.E., Mansouri, K., Lafifi, M.M. (2022). Greenhouse climate modeling using fuzzy neural network machine learning technique. Revue d'Intelligence Artificielle, 36(6): 925-930. https://doi.org/10.18280/ria.360614

[18] Tyagi, H., Kumar, R. (2021). Attack and anomaly detection in IoT networks using supervised machine learning approaches. Revue d'Intelligence Artificielle, 35(1): 11-21. https://doi.org/10.18280/ria.350102