Journal of Innovations in Computer Science and Engineering
Shahid Beheshti Unversity

January 2026, Volume 3, Issue 2

GossipTrust: A Decentralized Trust Management Model for SIoT via Gossip Learning and MAPE-K Control Loop

Moeinaddini, Elham , **CODE ORCID**:
Department of Computer Engineering, University of Jiroft, Jiroft, Iran, el_moin@ujiroft.ac.ir
Nazemi, Eslam✉, **CODE ORCID**: 0000-0001-9084-3789
Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran, nazemi@sbu.ac.ir

ABSTRACT

The Social Internet of Things (SIoT) enhances device interoperability through social relationships but introduces significant trust management challenges in dynamic, resource-constrained environments. Existing machine learning-based trust models often rely on static training data or centralized architectures, limiting their adaptability and scalability in real-world SIoT deployments. This paper proposes GossipTrust, a novel decentralized trust model that integrates fog computing with adaptive control mechanisms for robust trust management. Our approach deploys machine learning models on fog nodes to enable distributed trust computation while minimizing resource constraints on individual devices. The core innovation combines a MAPE-K (Monitor, Analyze, Plan, Execute, and Knowledge) control loop for real-time attack detection with a gossip learning protocol that allows fog nodes to collaboratively refine trust models through peer-to-peer updates without central coordination. Experimental results demonstrate that GossipTrust significantly outperforms baseline models, achieving 13% higher accuracy, 14% higher success rate, 10% higher F-measure, and 12% lower loss across various trust attack scenarios. The proposed solution effectively addresses key SIoT challenges, including scalability, adaptability, and resource efficiency.

Keywords: Trustworthiness, Self-adaptation, Social Internet of Things, scalability, resource efficiency, Machine Learning.

1. INTRODUCTION

The Internet of Things (IoT) has transformed modern systems by enabling smart devices to autonomously collect, exchange, and process data [1]. Extending this paradigm, the Social Internet of Things (SIoT) incorporates social networking principles, allowing devices to establish human-like relationships and collaborative structures [2]. While this social framework enhances interoperability and service discovery, it introduces critical trust management challenges. In SIoT environments, devices must reliably evaluate peer credibility while mitigating risks from malicious actors and unreliable data, making robust trust models essential for secure and autonomous decision-making [3].

Trust management in SIoT faces three fundamental challenges: (1) the dynamic nature of device behaviors and relationships requiring adaptive evaluation mechanisms; (2) scalability limitations of traditional approaches in handling exponential device growth; and (3) resource constraints of IoT devices necessitating lightweight computational solutions [4]. While Machine Learning (ML) techniques have emerged as promising tools for analyzing complex device interactions [5], current ML-based trust models suffer from a critical limitation: their reliance on static training datasets fails to accommodate the evolving nature of SIoT environments [6], [7].

Existing architectural approaches present significant trade-offs. Centralized ML models, while computationally efficient, create single points of failure and performance bottlenecks [6], [7], [8]. Distributed alternatives eliminate central dependency but often exceed the resource constraints of typical IoT devices [9], [10]. Decentralized architectures using fog or edge nodes offer a middle ground [11], [12], but remain vulnerable to data poisoning attacks where malicious devices submit falsified information. The Monitor, Analyze, Plan, Execute, and Knowledge (MAPE-K) control loop provides a structured framework for developing self-adaptive systems capable of dynamic runtime adjustment [13]. Recent work by Moeinaddini et al. [14] applied MAPE-K for malicious device identification in SIoT, but their approach retains vulnerabilities to erroneous transaction data and relies on federated learning with centralized cloud coordination.

To address these limitations, this paper proposes a novel trust evaluation model that integrates gossip learning with the MAPE-K framework. Gossip learning enables decentralized model refinement through peer-to-peer parameter exchange [15], offering privacy preservation and scalability advantages particularly suited to SIoT environments [16]. To the best of our knowledge, our

Submit Date: 2025-10-13
Revise Date: 2026-06-15
Accept Date: 2026-06-27
✉ Corresponding author

approach represents the first application of gossip learning to SIoT trust management. The key contributions of our approach are:

- The proposed model incorporates a MAPE-K control loop that actively monitors transaction data streams in real-time, analyzes behavioral patterns to detect falsified data, and dynamically isolates malicious devices attempting trust attacks.
- A gossip learning mechanism enabling fog nodes to collaboratively refine trust models through periodic parameter exchange, eliminating central coordination while improving accuracy.
- To address the dynamic nature of SIoT environments, our solution employs incremental retraining of local ML models through gossip learning, which utilizes newly generated transaction data. This approach ensures that trust evaluations remain accurate as network conditions and device behaviors evolve.

The paper proceeds as follows: Section 2 explains SIoT fundamentals, gossip learning, and trust attacks. Section 3 reviews related work. Section 4 defines notations and presents our model. Section 5 discusses experiments, and Section 6 concludes.

LIST OF ACRONYMS

ANN	Artificial Neural Network
BMA	Bad Mouthing Attack
BSA	Ballot Stuffing Attack
CEC	Computational and Energy Capability
CLOR	Co-location Object Relationship
CWOR	Co-work Object Relationship
DA	Discriminatory Attack
DBN	Deep Belief Network
DNN	Deep Neural Networks
DT	Distrust
HT	High Trust
IoT	Internet of Things
KNN	K-Nearest Neighbors
LR	Logistic Regression
MAPE-K	Monitor, Analyze, Plan, Execute, and Knowledge
ME	Malicious for everyone
ML	Machine Learning
MLP	Multi-Layer Perceptron
NB	Naive Bayes
OOA	On-Off Attack
OOR	Ownership Object Relationship
OSA	Opportunistic Service Attack
POR	Parental Object Relationship
RBF	Radial Basis Function
RT	Random Tree
SIoT	Social Internet of Things
SA	Sybil Attack
SOR	Social Object Relationship
SPA	Self-Promoting Attack
SVM	Support Vector Machine
T	Trust
UT	Untrust
WA	Whitewash Attack

2. BASIC CONCEPTS

This section establishes the conceptual foundation for our proposed model by defining four core components: (1) relationships in SIoT, (2) prevalent threat models targeting trust-based systems, (3) the autonomic computing paradigm, and (4) the gossip learning mechanism. These interconnected concepts collectively enable robust trust management in dynamic SIoT environments.

2.1. Relationships in Social IoT

The SIoT represents an evolutionary integration of IoT technologies with social networking principles. In this framework, smart devices acquire social capabilities beyond traditional data functions, enabling autonomous relationship formation akin to human social structures. These device-to-device interactions enhance trust-based information sharing while optimizing network efficiency [2]. SIoT networks exhibit several characteristic interaction patterns:

- Ownership Object Relationship (OOR): Connections formed between devices belonging to a single user, creating a private device ecosystem [17].
- Parental Object Relationship (POR): Relationships among devices manufactured simultaneously by the same vendor, leveraging shared hardware/software heritage [18].
- Co-location Object Relationship (CLOR): This relationship is established when devices operate in physical proximity, enabling location-aware cooperation [18].
- Co-work Object Relationship (CWOR): Dynamic groupings where devices collaborate temporarily to achieve specific operational objectives [19].
- Social Object Relationship (SOR): Indirect relationships mediated through owner interactions, extending human social graphs to devices [1].

2.2. Threats to Trust-Based Systems

Trust evaluation mechanisms in IoT ecosystems are vulnerable to sophisticated adversarial strategies that compromise system integrity. Here, prevalent attack methodologies targeting these systems are defined.

- Malicious for Everyone (ME): A globally hostile device consistently delivers fraudulent services indiscriminately, disregarding requester identity or context. This represents the most fundamental form of trust exploitation, where malicious intent is applied uniformly across all interactions [20]
- Discriminatory Attack (DA): Adaptive malicious nodes dynamically alter their behavior based on target profiles, preferentially victimizing devices with weaker social connections or non-affiliated status. This selective targeting complicates detection through conventional trust metrics [21].
- On-Off Attack (OOA): Attackers intermittently deliver substandard services rather than maintaining consistent quality. This randomized poor performance strategy evades detection by occasionally meeting baseline expectations [22].

- Opportunistic Service Attack (OSA): This attack occurs when nodes deliberately improve service quality only when trust metrics approach critical thresholds. This calculated behavior masks malicious intent during routine monitoring periods [22].
- Whitewash Attack (WA): Ephemeral node attacks exploit network dynamics by periodically disconnecting and rejoining with cleared trust histories, effectively erasing evidence of prior malicious activity [21].
- Sybil Attack (SA): A SA occurs when a single malicious entity forges multiple pseudonymous identities (Sybil nodes) to gain disproportionate influence in a decentralized network [23].
- Bad Mouthing Attack (BMA): Through systematic reputation sabotage, adversarial nodes falsify evaluations to disadvantage legitimate service providers while promoting compromised alternatives [22].
- Ballot Stuffing Attack (BSA): This collusive strategy involves coordinated deception where a malicious actor deliberately issues inflated positive evaluations to another adversarial node [21].
- Self-Promoting Attack (SPA): This attack occurs when a malicious node systematically fabricates self-promoting trust evaluations to artificially elevate its reputation score [22].

2.3 Autonomic Computing

Self-adaptation refers to a system's capability to dynamically modify its behavior in response to changing environmental conditions or internal states without external intervention. This evolutionary characteristic enables intelligent systems to maintain optimal performance through continuous monitoring, decision-making, and execution of adaptation strategies [24].

This paradigm is operationalized through IBM's autonomic computing framework, which implements self-management via the MAPE-K control loop [25]. The architecture decomposes autonomous functionality into four interconnected phases: (1) Monitoring to collect system/environment data, (2) Analysis to detect deviations from desired states, (3) Planning to determine corrective actions, and (4) Execution to implement changes—all supported by a shared Knowledge base that preserves historical and contextual information [13], [25]. While the MAPE-K framework was initially conceived for enterprise computing environments, its autonomic control paradigm demonstrates significant applicability in IoT ecosystems and trust management architectures [26], [27], [28], [29]. The model's inherent capabilities for real-time system monitoring, adaptive decision-making, and knowledge-driven execution align precisely with the dynamic requirements of distributed IoT networks and evolving trust relationships.

2.4. Gossip Learning

Gossip learning is a decentralized ML paradigm inspired by epidemic protocols in distributed systems, where nodes iteratively exchange and merge model parameters with direct neighbors rather than relying on centralized coordination. This approach enables collaborative model training while preserving data locality, as nodes only share learned parameters (not raw data), making it particularly suitable for privacy-sensitive applications and resource-constrained environments like IoT [15]. By mimicking human rumor spreading, gossip learning achieves global knowledge dissemination through local interactions, offering inherent scalability and fault tolerance absent in traditional federated or centralized learning architectures [30].

The technique's robustness stems from its asynchronous, peer-to-peer communication model, which eliminates single points of failure and adapts dynamically to network changes. Each node maintains its local model, periodically synchronizing with randomly selected peers to incrementally refine global knowledge. This makes gossip learning ideal for dynamic IoT ecosystems where device availability fluctuates, while its lightweight nature accommodates heterogeneous hardware capabilities [16], [31].

Federated Learning and Gossip Learning are both distributed paradigms designed to train machine learning models without sharing raw data, yet they diverge fundamentally in their coordination mechanisms and network topologies. Federated Learning typically employs a centralized star architecture where a parameter server orchestrates model aggregation from local client updates, offering deterministic convergence but presenting a single point of failure and potential communication bottlenecks at the server [14]. In contrast, Gossip Learning adopts a fully decentralized peer-to-peer approach, where nodes iteratively exchange and merge model parameters with their immediate neighbors. While this decentralized nature enhances system robustness against node churn and eliminates the need for a central authority, it often results in slower, stochastic convergence compared to the coordinated execution of Federated Learning. Consequently, Federated Learning is best suited for managed infrastructures with a reliable central entity, whereas Gossip Learning provides a resilient solution for ad-hoc or serverless environments such as IoT and other scenarios requiring distributed intelligence without centralized infrastructure [16]. Table 1 shows the key difference between Gossip Learning and Federated Learning.

Table 1: Fundamental Differences between Gossip Learning and Federated Learning

Attribute	Federated Learning	Gossip Learning
Network Architecture	Centralized (Star Topology)	Decentralized (P2P Mesh)
Aggregation Mechanism	Server-side	Local Peer-to-Peer Merging
Reliability	Susceptible to server failure	Highly robust to node churn
Communication Bottleneck	Central Server Bandwidth	Per-node Network Traffic

3. RELATED WORKS

This section reviews existing trust management models in the SIoT, with a focus on approaches leveraging ML techniques. These models are categorized based on their architectural use of ML.

Some trust models employ a centralized ML model (typically hosted on a server or cloud) to predict trust across the entire system. In this approach, all trust evaluations rely on a single ML model, thereby simplifying the decision-making process. This architecture is particularly advantageous in resource-constrained IoT environments, as it reduces computational load on individual nodes [32]. While some studies have adopted centralized ML-based trust models [6], [33], this approach introduces critical limitations, most notably a single point of failure and performance bottlenecks. Since all trust computations depend on the central model, any disruption (e.g., cyberattacks or system overload) can compromise the entire network. Moreover, as the

network scales, the central server may struggle with increasing request volumes, leading to significant delays in transaction processing [32].

In contrast, distributed trust models deploy separate ML models on each IoT device, eliminating reliance on a central authority. This enhances privacy, security, and scalability, as the network can expand without performance bottlenecks. Many researchers have explored distributed ML-based trust frameworks [8], [9], [10], [20], [34], [35], [36], [37]. Despite these benefits, distributed architectures face challenges, including lack of global coordination among nodes, making it difficult to establish a unified trust perspective, and resource constraints, as many IoT devices lack the computational power to efficiently run independent ML models [32], [38].

To address these limitations, some trust models adopt a decentralized approach, deploying ML models on trusted local nodes (e.g., fog or edge devices). Here, decision-making is distributed across multiple trusted entities rather than being centralized or fully dispersed. This hybrid design balances security (by eliminating single points of failure) and efficiency (by fairly distributing computational loads), making it ideal for heterogeneous IoT environments [32], [38]. Relevant studies include [11], [12], [14]. However, decentralized models require robust consensus mechanisms to ensure coordination among independent nodes and a mechanism to detect incorrect data, as malicious devices may submit falsified inputs that mislead ML models. Thus, ensuring data integrity through feedback validation is critical to maintaining model accuracy.

Another problem with most ML-based trust models is that they train on a fixed dataset and do not incorporate updates, which limits their effectiveness in dynamic SIoT environments. The studies by Marche et al. [20] and Moeinaddini et al. [14] are notable exceptions, as they explore incremental learning. Marche et al. in [20] proposed a trust evaluation framework employing sequentially updated Support Vector Machine (SVM). While innovative, this methodology implements the classification model uniformly across all network nodes, disregarding the significant heterogeneity in processing power, storage capacity, and energy resources among IoT devices. In contrast, the model in [14] employs Multi-Layer Perceptron (MLP) models within fog nodes, which are periodically updated based on transaction results sent by devices. A challenge in this model arises when malicious devices submit incorrect feedback data, leading to misleading updates in the ML models.

Additionally, while federated learning has been explored for trust assessment in IoT and SIoT [14], [39], [40], [41], [42], [43], the application of gossip learning (a decentralized alternative) for trust computation in SIoT environments remains unexplored. This study introduces an innovative trust management model that combines two key components: a polling mechanism embedded in a MAPE-K feedback loop to detect malicious behaviors, and a gossip learning approach enabling decentralized collaboration among local ML models. The proposed approach effectively detects trust attacks while adapting to the dynamic and evolving nature of SIoT environments. Table 2 presents a comparative analysis of existing trust models and our proposed approach, evaluating five key aspects: (1) ML techniques employed, (2) ML architecture design, (3) trust attack detection capability, (4) adaptation mechanism implementation, and (5) erroneous data detection functionality.

Table 2: Related works comparison

Study	ML technique	ML Architecture	Attack Detection	Adaptation	Erroneous Data Detection
[33]	(Deep Neural Networks) DNN	Centralized	✓	✗	✗
[6]	K-Means	Centralized	✓	✗	✗
[9]	Artificial Neural Network (ANN)	Distributed	✗	✗	✗
[10]	(NaiveBayes) NB, (Random Tree) RT, MLP	Distributed	✓	✗	✗
[34]	MLP	Distributed	✓	✗	✗
[35]	K-Means	Distributed	✓	✗	✗
[36]	Logistic Regression (LR), ANN, (Deep Belief Network) DBN	Distributed	✓	✗	✗
[37]	LR, ANN, DBN	Distributed	✗	✗	✗
[20]	Incremental SVM	Distributed	✓	✗	✗
[12]	(K-Nearest Neighbors) KNN, Weighted Sum	Decentralized	✓	✗	✗
[11]	NB, (Radial Basis Function) RBF, MLP, RT	Decentralized	✓	✗	✗
[14]	MLP, Federated Learning	Decentralized	✓	✓	✗
Proposed Model	MLP, Gossip Learning	Decentralized	✓	✓	✓

4. PROPOSED TRUST MODEL

In the proposed trust model, it is assumed that there are N IoT devices in the system, denoted $\{D_1, D_2, \dots, D_N\}$. Devices collaborate to exchange data and offer various services. Every IoT device maintains a direct association with its nearest fog node. The system comprises M distinct fog nodes distributed across the network.

Fog nodes have a local training dataset to store trust-related information, a local ML model to predict trust levels, and a MAPE-K loop to detect malicious behaviors. Using fog nodes, the local network can reduce computational demands and minimize energy consumption [44]. The system model of the proposed trust model is depicted in Figure 1.

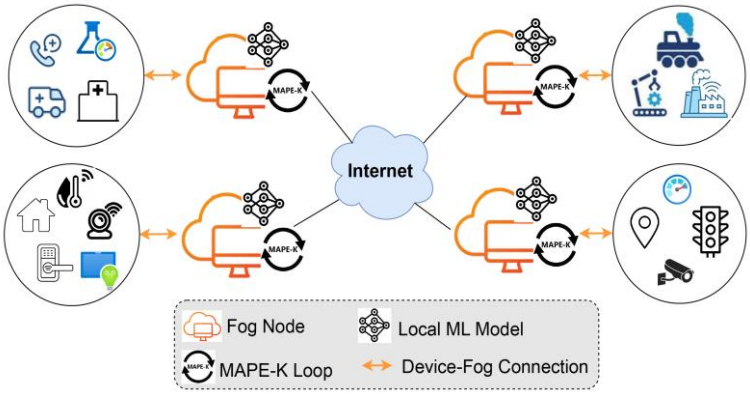

Figure 1. System model of the proposed trust model

Each device in the network can provide a variety of services. Let S_{D_i} represent the set of services offered by the device D_i. In the context of social IoT, it is assumed that devices can establish social relationships, forming a reliable friendship network [27]. Device D_i can establish social connections with other nodes, collectively designated as its friends and represented by $\mathcal{F}_{D_i}$. An overview of the proposed method is shown in Figure 2.

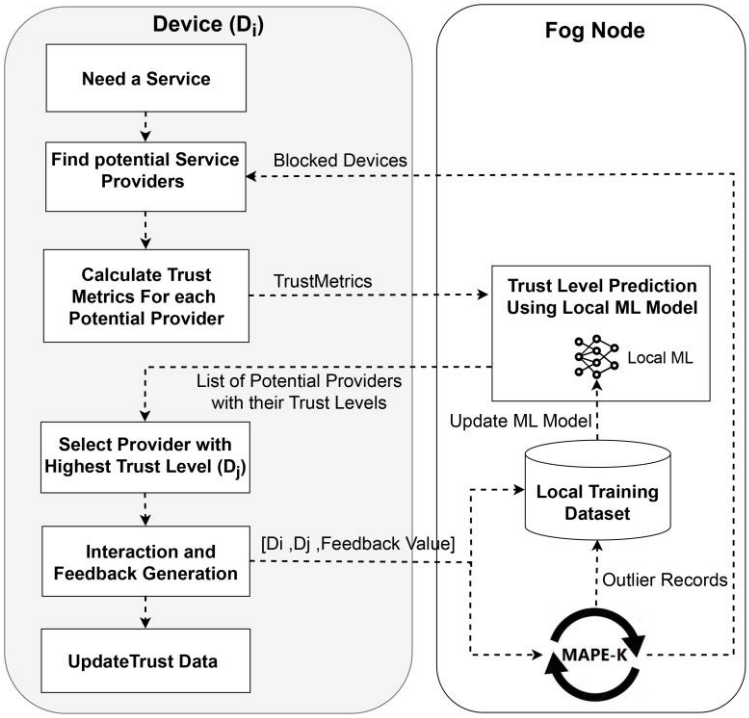

Figure 2. An overview of the proposed trust model

Upon requesting service s_h, device D_i initiates a discovery process to identify all capable service providers located within its operational proximity. This compilation of available nodes represents potential candidates for fulfilling the service request. For each potential provider D_k, the trustor device D_i calculates eight trust metric values (which are explained in Section 4.1). D_i then sends computed trust metrics to its corresponding fog node, which uses its ML model for trust level predictions. Trust metrics serve as feature vectors for the local ML model, enabling predictions of trustworthiness for each service provider. The ML model outputs a trust assessment $\mathcal{TL}_{D_i,D_k}^{s_h}$, representing the predicted trust of device D_k from the perspective of D_i for service s_h. These predictions are discretized into four classification tiers: High Trust (HT), Trust (T), Untrust (UT), or Distrust (DT). The trust level is predicted as follows:

$$\mathcal{TL}_{D_i,D_k}^{s_h} = \mathcal{M}(\text{TrustMetrics}_{D_i,D_k}^{s_h}) \quad (1)$$

Where $\text{TrustMetrics}_{D_i,D_k}^{s_h}$ is a set of eight trust metrics computed for device D_k from the perspective of device D_i for service s_h. These trust metrics are used as input features by the local ML model, denoted as $\mathcal{M}$.

The fog node subsequently communicates the computed trust assessments for all candidate providers back to the requesting device D_i. Device D_i then identifies the optimal service provider D_j by selecting the node with the maximum trust level, subject to the following constraints:

i. If there is insufficient historical transaction data between D_i and D_j for service s_h (specifically, fewer than γ), D_i will choose the provider with the highest trust level.

ii. Conversely, if there is adequate historical transaction data (more than or equal to γ), D_i will select D_j with the highest trust level only if its direct trust value exceeds 0.5. The direct trust between D_i and D_j for service s_h is denoted as $DT_{D_i,D_j}^{s_h}$ and computed using Equation (2).

$$DT_{D_i,D_j}^{s_h} = \frac{\sum_{k=1}^{\mathcal{K}_{D_i,D_j}} w^k . fb(k)}{\sum_{k=1}^{\mathcal{K}_{D_i,D_j}} w^k} \tag{2}$$

where w_k is the round-weighted value calculated using the following formula:

$$w_k = \begin{cases} 1/(\text{current\_round} - r_k + \epsilon) & \text{if current\_round} - r_k \leq R_{max} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

In this context, fb(k) represents the k-th feedback value, and r_k denotes the round number when the feedback fb(k) was added. The variable current_round refers to the current round during the calculation, while the term R_{max} represents the maximum number of rounds for which feedback is considered valid, effectively disregarding older feedback. Additionally, ϵ is a small constant (e.g., 1) used to avoid division by zero. Finally, $\mathcal{K}_{D_i,D_j}$ signifies the total number of feedback entries generated by D_i for D_j.

When multiple candidate providers achieve identical maximum trust levels, device D_i employs a randomized selection algorithm to choose D_j. If no available provider meets the predefined trust threshold (T), D_i aborts the transaction, resulting in a service request failure. Upon successful selection, D_i formally initiates the service request to the designated provider D_j. Following service delivery, D_i generates a feedback assessment for evaluating the received instance of s_h. For each service provider D_j offering service s_h, device D_i maintains a feedback set of the 20 most recent quality evaluations, structured as formalized in Equation (4):

$$FB_{D_i,D_j}^{s_h} = \{fb(1), fb(2), \dots, fb(\mathcal{K}_{D_i,D_j})\} \tag{4}$$

In this equation, $\mathcal{K}_{D_i,D_j}$ denotes the total count of transactions between devices D_i and D_j. The feedback value is derived from a multi-objective evaluation of Quality of Service metrics, including availability, response time, and success rate. These metrics are aggregated and discretized into one of four distinct satisfaction levels: 1 (High Satisfaction), 0.75 (Satisfaction), 0.25 (Low Satisfaction), and 0 (Dis-Satisfaction). After each interaction, the feedback list is updated to facilitate future trust computations. The generated feedback value for the service provider is then transmitted to the fog node for storage in its training dataset and for monitoring by the MAPE-K loop.

Feedback values are critical to updating the ML model at the fog node, but are vulnerable to manipulation by malicious devices. Malicious devices may send inaccurate feedback values to their edge server to mislead the ML model. To address this issue, the proposed trust model employs the MAPE-K control loop to detect devices that send incorrect feedback values, subsequently removing their data from the training dataset. This process is discussed in Section 4.3.2.

In dynamic SIoT environments, device behavior patterns evolve, potentially diminishing the accuracy of established trust models. To maintain relevance, the ML model undergoes continuous incremental updates, allowing the trust framework to adapt to these behavioral shifts and preserve its predictive validity amidst network dynamics. Consequently, local ML models are aggregated through gossip learning using newly collected records from the local training sets, facilitating collaborative learning and ongoing updates, as detailed in Section 4.2. The MAPE-K loop also plays a crucial role in continuously monitoring and analyzing behavior to detect malicious devices and erroneous feedback, as explained in Section 4.3.

The inherent heterogeneity of SIoT networks makes acquiring pre-labeled training data impractical in real-world deployments. To overcome this limitation, our model implements an automated labeling system through transactional interactions. During system initialization, trustors engage in randomized provider selection, deliberately disregarding existing trust assessments. This unbiased sampling strategy ensures comprehensive behavioral representation across all provider types. Each service interaction produces trust metrics and feedback labels. These elements form structured training tuples stored in localized datasets. Local ML models are then trained using these datasets. Once ML models are trained, trust levels for each service interaction are predicted by these models. After selecting a trusted service provider and completing the transaction, the requester generates feedback on the received service. This feedback is then mapped into discrete labels for the transaction: High Satisfaction, Satisfaction, Low Satisfaction, and Dis-Satisfaction. Since these transaction results feed into the incremental learning process, malicious requesters who intentionally generate incorrect feedback could distort this process. To prevent this, the proposed trust model analyzes feedback values to detect and filter erroneous entries, as detailed in Section 4.3.

4.1 Trust Metrics

Trust metrics are quantitative measures that assess the reliability of devices based on various factors. They aid in decision-making by helping select trusted service providers and identify potentially malicious entities through a numerical representation of device behavior. Here are eight trust metrics utilized in the proposed trust model:

Social Relationship: The Social Relationship metric ($\mathcal{SR}_{D_i,D_j}$) evaluates the social proximity between devices D_i and D_j by analyzing their established social ties. Following the methodology in [3], the input data for this metric consists of a set of active social links identified between the two nodes. The classification logic involves mapping each identified link to a predefined numerical value based on its relationship category, where $S_{\text{OOR}} = 1$, $S_{\text{CLOR}} = 0.8$, $S_{\text{SOR}} = 0.6$, and $S_{\text{CWOR}} = 0.5$. In cases where multiple concurrent links exist, the computational logic selects the highest score among all active relationships to represent the metric value, and zero otherwise. Regarding efficiency, the sub-algorithm maintains a computational time complexity of $O(K)$, where K is the number of relationship types (at most 5, effectively $O(1)$ in practice) and a space complexity of $O(1)$ to store the maximum value during comparison [3].

Centrality: The centrality measure quantifies a service provider's structural significance within the network topology from the requesting device's perspective [45]. This metric is computed as:

$$\mathcal{C}_{D_i,D_j} = \frac{|\mathcal{F}_{D_i} \cap \mathcal{F}_{D_j}|}{|\mathcal{F}_{D_i}|} \quad (5)$$

Where $|\mathcal{F}_{D_i} \cap \mathcal{F}_{D_j}|$ denotes the cardinality of mutual friends between devices D_i and D_j, while $|\mathcal{F}_{D_i}|$ represents the total number of D_i's friends.

By employing a Jaccard-like similarity logic, this metric quantifies the structural integration of the service provider within the requester's immediate social circle. Computationally, finding the intersection of these sets yields a time complexity of $O(|\mathcal{F}_{D_i}| + |\mathcal{F}_{D_j}|)$. Given that the number of friends per node is limited to a constant, this simplifies to $O(1)$. Along with a constant space complexity of $O(1)$ beyond initial input storage, this ensures high efficiency for real-time network topology analysis.

Recommendation: While devices may incorporate peer recommendations when evaluating service providers, such mechanisms remain vulnerable to adversarial attacks, including BSA and BMA. To enhance robustness, a post-transaction validation protocol is used where requesters assess recommender credibility through direct experience. Following [20]'s verification method, when a recommender's positive assessment aligns with satisfactory service, the recommender earns full credibility (score=1), but if a negative recommendation contradicts actual satisfactory service, the recommender receives a score=0. The recommendation feedback set for each $D_x \in \mathcal{F}_{D_i}$ is defined as:

$$FBR_{D_i,D_x} = \{fbR(1), fbR(2), \dots, fbR(\mathcal{L}_{D_i,D_x})\} \quad (6)$$

where $\mathcal{L}_{D_i,D_x}$ represents the total number of feedbacks given by D_i to D_x. The recommendation metric is expressed as:

$$\mathcal{R}^{s_h}_{D_i,D_j} = \frac{1}{|\mathcal{F}_{D_i} \cap \mathcal{F}_{D_j}|} \sum_{D_x \in \mathcal{F}_{D_i} \cap \mathcal{F}_{D_j}} Weight_{D_i,D_x}.\mathcal{DT}^{s_h}_{D_x,D_j} \quad (7)$$

where

$$Weight_{D_i,D_x} = \frac{1}{\mathcal{L}_{D_i,D_x}} \sum_{k=1}^{\mathcal{L}_{D_i,D_x}} fbR_{D_i,D_x}(k) \quad (8)$$

$Weight_{D_i,D_x}$ represents the average feedback on the recommendation given by the device D_i to D_x and is used to weight opinions. Meanwhile, $\mathcal{DT}^{s_h}_{D_x,D_j}$ denotes the direct trust value of the device D_x towards the device D_j.

The computational overhead of the Recommendation metric is characterized by a time complexity of $O(M.\bar{\mathcal{L}})$, where M is the number of mutual friends ($M = |\mathcal{F}_{D_i} \cap \mathcal{F}_{D_j}|$) and $\bar{\mathcal{L}}$ denotes the average volume of historical feedback associated with each recommender. Since the number of friends per node is limited to a constant, M is also bounded by a constant, effectively reducing the time complexity to $O(1)$. This reduction is further supported by the fact that each device maintains a maximum of 10 feedback entries per friend, making the average historical feedback volume, $\bar{\mathcal{L}}$, a constant factor that does not scale with network size. Additionally, the algorithm maintains a space complexity of O(M) to store the calculated credibility weights for the identified recommenders, which similarly becomes $O(1)$ under the bounded friend constraint.

Computational and Energy Capability: Device reliability in service provisioning correlates strongly with available computational resources, storage capacity, and energy efficiency. The Computational and Energy Capability (CEC) metric categorizes IoT devices into three distinct classes based on these operational constraints, as standardized in [46]:

- CEC-0 (Highly Constrained): Devices with severe limitations in processing, storage, and power (CEC score = 0).
- CEC-1 (Moderately Constrained): Devices balancing moderate resource availability with operational demands (CEC score = 0.5).
- CEC-2 (Unconstrained): Resource-rich devices capable of sustained high-performance operation (CEC score = 1).

Four additional trust metrics evaluate trustee reliability through empirical analysis of historical interactions. If the number of transactions between the trustor and trustee on a specific service is equal to γ or more, the metrics are computed accordingly; otherwise, they are set to 0.5. This approach ensures that the metrics reflect the trustee's historical behavior, requiring sufficient transactions for accuracy.

The CEC metric is formalized as a deterministic lookup function. From a computational perspective, this involves a constant-time attribute retrieval, resulting in a time complexity of $O(1)$. Similarly, since no additional memory is required for processing beyond reading the class attribute, the space complexity is also $O(1)$, ensuring negligible overhead for even the most resource-constrained IoT nodes.

Goodness: This metric quantifies service reliability as the mean value of all historical feedback scores for a given service interaction, following the methodology established in [20]. This is formally expressed by Equation (9).

$$\mathcal{G}^{s_h}_{D_i,D_j} = \frac{1}{\mathcal{K}_{D_i,D_j}} \sum_{k=1}^{\mathcal{K}_{D_i,D_j}} fb^{s_h}_{D_i,D_j}(k) \quad (9)$$

Here, $\mathcal{K}_{D_i,D_j}$ represents the total transaction count between D_i and D_j for service s_h.

Usefulness: The usefulness metric evaluates recent behavior [20] and is calculated using the following formulas:

$$\mathcal{U}^{s_h}_{D_i,D_j} = \xi \left(\sum_{k=0}^{k_s-1} w^{(k+1)} fb^{s_h}(\mathcal{K}_{D_i,D_j} - k) \right) \quad (10)$$

$$w^k = \rho(1-\rho)^{(k-1)} \quad (11)$$

where the normalization variable ξ ensures all usefulness values remain bounded within the unit interval [0,1] and the weight w^k prioritizes recent interactions. In our model, we set the window size k_s= 10 and the prioritization weight ρ= 0.4 [20]. This parameterization establishes a balanced evaluation framework that weights recent interactions while maintaining mathematical consistency.

Perseverance: This measure evaluates a service provider's ability to maintain reliable and superior service quality over time [20]. The computation is based on the Equation below:

$$\mathcal{P}^{s_h}_{D_i,D_j}(\mathcal{K}_{D_i,D_j}) = \begin{cases} 0.5 & \text{if } \mathcal{K}_{D_i,D_j} \leq 1 \\ \mathcal{P}^{s_h}_{D_i,D_j}(\mathcal{K}_{D_i,D_j} - 1) + \mathcal{V}_{D_i,D_j} & \text{if } \mathcal{K}_{D_i,D_j} > 1 \end{cases} \quad (12)$$

The value $\mathcal{V}_{D_i,D_j}$ represents incentives (α = 0.1) for positive feedback or sanctions (β = 0.2) for negative feedback, scaled by the consistency streak θ_{D_i,D_j} (Equation (13)). Any value exceeding the [0, 1] range is truncated to the nearest valid limit.

$$\mathcal{V}_{D_i,D_j} = \begin{cases} \theta_{D_i,D_j}\alpha & \text{if } fb(k) = 1 \\ -\theta_{D_i,D_j}\beta & \text{if } fb(k) = 0 \end{cases} \quad (13)$$

The variable θ_{D_i,D_j} captures the streak of consistent favorable or unfavorable ratings in $FB^{s_h}_{D_i,D_j}$. The index k spans from the initial interaction ($k = 1$) to the latest ($k = \mathcal{K}_{D_i,D_j}$), where $\mathcal{K}_{D_i,D_j}$ denotes the cumulative exchanges between the paired devices for service s_h. Should the computed consistency metric fall outside the normalized range [0,1], it is truncated to the closest valid limit.

Fluctuation: This measure captures the tendency of a provider to oscillate between satisfactory and unsatisfactory performance levels [11]. The calculation involves tracking shifts between distinct evaluation states and normalizing by the overall quantity of assessments, as expressed in Equation (14).

$$\mathrm{F}^{s_h}_{D_i,D_j} = 1 - \frac{1}{\mathcal{K}_{D_i,D_j}} \sum_{k=1}^{\mathcal{K}_{D_i,D_j}-1} |fb(k) \neq fb(k+1)| \quad (14)$$

Here $\mathcal{K}_{D_i,D_j}$ denotes the number of transactions for service s_h between devices D_i and D_j.

For the empirical metrics derived from historical interaction data (specifically Goodness, Usefulness, Perseverance, and Fluctuation), the computation involves a systematic evaluation of past feedback records. Since each device maintains the latest service feedback with a maximum length of 20 for each service provided by any service provider, the total number of historical transactions T is bounded by a constant factor per service provider. Consequently, this process yields a time complexity of $O(1)$, reflecting that the linear scan of interaction logs operates within a fixed upper bound. Despite the temporal analysis, these metrics maintain a constant space complexity of $O(1)$, as the calculation requires only the iterative accumulation of scores or the updating of statistical variables.

The set of trust metrics for device D_j from the perspective of device D_i for service s_h is defined as:

$$\text{TrustMetrics}^{s_h}_{D_i,D_j} = [\mathcal{SR}_{D_i,D_j}, \mathcal{C}_{D_i,D_j}, \mathcal{R}^{s_h}_{D_i,D_j}, \mathcal{CEC}_{D_j}, \mathcal{G}^{s_h}_{D_i,D_j}, \mathcal{U}^{s_h}_{D_i,D_j}, \mathcal{P}^{s_h}_{D_i,D_j}, \mathrm{F}^{s_h}_{D_i,D_j}] \quad (15)$$

Collectively, the eight trust metrics are designed with computational efficiency as a foundational principle, ensuring suitability for deployment on resource-constrained IoT devices. Each metric leverages bounded input parameters—such as a constant maximum number of relationship types (at most five), a fixed upper bound on the number of friends per device, a limited feedback window of ten entries per friend for recommendation credibility, and a maximum of twenty historical service feedback entries per service provider for empirical metrics. As a result, all metrics achieve constant time complexity, $O(1)$, and constant space complexity, $O(1)$, regardless of network size or transaction history length. This efficiency guarantees that trust evaluation can be performed in real time without introducing significant computational overhead or memory footprint, enabling scalable and responsive trust management even in highly dynamic IoT environments.

4.2 Collaborative Learning through Gossip Learning

Gossip learning enables decentralized collaborative training of local ML models among fog nodes, where nodes iteratively exchange and refine model parameters (e.g., weights, gradients) without centralized coordination. This approach preserves data privacy by retaining raw data locally and only sharing learned model insights [15], [16]. Gossip learning is triggered every 500 rounds of local trust prediction, allowing sufficient time for fog nodes to accumulate meaningful transaction data for model updates. The process consists of the following phases:

i. **Model Initialization:** At the beginning of each 500 rounds, each fog node begins with a locally trained ML model based on its accumulated transaction data and trust assessments. This initial model serves as the starting point for the collaborative refinement process.
ii. **Peer Selection:** At each gossip interval, every fog node randomly selects k fog nodes to initiate model parameter exchange, ensuring diverse knowledge dissemination across the network.
iii. **Model Exchange:** Selected fog nodes exchange their current model parameters (e.g., weights, gradients) without sharing raw training data, maintaining data privacy while enabling knowledge transfer.
iv. **Model Aggregation:** Upon receiving parameters from peers, each fog node aggregates the external models with its local model using a weighted averaging function (Federated Averaging), incorporating insights from multiple nodes while preserving its unique local knowledge.
v. **Local Refinement:** Each fog node fine-tunes the aggregated model using its local dataset, allowing personalized adaptation while benefiting from collective intelligence, thus enhancing the model's robustness and generalization capability.

This cyclic process continues every 500 rounds, enabling fog nodes to collaboratively improve their trust prediction accuracy while maintaining decentralization and data privacy. The procedural steps of the gossip learning mechanism implemented at the fog node level are detailed in Algorithm 1. By periodically aggregating peer insights through lightweight parameter exchanges, nodes achieve global consensus on trust patterns without centralized coordination. The approach not only mitigates single points of failure but also adapts dynamically to emerging threats, as local detections of malicious behavior propagate organically through the network.

Algorithm 1 Gossip-Based Collaborative Learning

Input: Gossip interval $G = 500$, peer count k, Local dataset D_i
Output: Updated local trust model M_i
for round $= 1$ to G **do**
  Train local ML model M_i on D_i using transaction data {Local Training Phase}
  Store model updates as (weights, gradients)
  Select k random peers $P = \{P_j | j = 1, ..., k\}$ {Gossip Learning Phase}
  for each peer $p \in P$ **do**
    Send current M_i to p
    Receive M_p from p
  end for
  $M_i^{\text{new}} = l_r \cdot M_i + (1 - l_r) \cdot \left(\frac{1}{k} \sum_{p \in P} M_p \right)$ {where $l_r \in [0, 1]$ is the learning rate}
  Fine-tune M_i^{new} using D_i
  $M_i = M_i^{\text{new}}$
end for
return M_i

In the current version of our model, the aggregation process employs a uniform weighting scheme for all peer models to maintain computational efficiency and establish a baseline for gossip-based trust propagation. However, we acknowledge that weighting peer contributions based on node reputation or local sample counts could further enhance the model's robustness. This enhancement is planned as a primary direction for our future research.

4.3 MAPE-K Loop in the Trust Model

The MAPE-K loop plays a crucial role in ensuring the integrity and performance of the proposed trust model. Each fog node uses the MAPE-K control loop to adapt to changes in device behavior, facilitating the identification of malicious devices. The following sections outline the steps involved in the MAPE-K loop, with a focus on the detection of outliers and malicious devices.

4.3.1 Monitor

The *Monitor* component continuously collects feedback data regarding service providers. The collected feedback values are stored in the format $[D_i, D_j, FV]$, where D_i represents the trustor device, D_j denotes the trustee device, and FV indicates the feedback value generated by D_i for D_j. The *Monitor* component continuously collects feedback data for each trustee device, updating the $feedback\_aggregation$ list stored within the *Knowledge* component. The aggregated results are subsequently transmitted to the *Analyze* component.

4.3.2 Analyze

Leveraging both the policies stored in the *Knowledge* component and the aggregated feedback from the *Monitor* component, the *Analyze* component performs a comprehensive analysis to identify malicious activities within the system. As outlined in Algorithm 2, the *Analyze* component first identifies devices generating outlier feedback values and adds the trustor device that sends the outlier feedback value, along with the corresponding trustee and feedback value, to $OTList$. Outlier feedback values are identified using the Interquartile Range (IQR) method [47], ensuring that the feedback accurately reflects genuine experiences.

Algorithm 2 Outlier Feedback and Malicious Device Identification

```
Input: feedback_aggregation (aggregated feedback dictionary)
Output: OTList (Outlier list), SuspectedList (Malicious devices list)
OTList ← []                                          ▷ Initialize empty outlier list
Suspectedlist ← []                                   ▷ Initialize empty suspected devices list
for each (trustee, trustor, feedbacks) IN feedback_aggregation do
    if LENGTH(feedbacks) > γ then                    ▷ Sufficient feedback threshold
        Q1, Q3 ← QUARTILES(feedbacks, [25, 75])      ▷ Calculate quartiles
        IQR ← Q3 - Q1                                ▷ Interquartile range
        lower_bound ← Q1 - 1.5 × IQR
        upper_bound ← Q3 + 1.5 × IQR
        for each feedback IN feedbacks do
            if feedback < lower_bound OR feedback > upper_bound then
                OTList.APPEND((trustee, trustor, feedback))   ▷ Add outlier feedback
            end if
        end for
        n ← LENGTH(feedbacks)                        ▷ Total feedback count
        count_0s ← COUNT(feedbacks WHERE feedback = 0)   ▷ Count distrust
    feedbacks (0s)
        if count_0s / n > 0.5 then                   ▷ Majority distrust condition
            SuspectedList.APPEND(trustee)            ▷ Flag trustee as suspected malicious
        end if
    end if
end for
    return OTList, SuspectedList
```

Additionally, the *Analyze* component determines potentially malicious devices by calculating the proportion of dissatisfied trustors (those providing feedback value equal to 0) relative to the total number of trustors that sent feedback for each trustee. If the majority of trustors are dissatisfied with a particular trustee, that trustee will be added to the $SuspectedList$. The polling technique is used to examine the members of $OTList$ and $SuspectedList$ to confirm their misbehavior.

Algorithm 3 shows the polling process for $OTList$. This polling algorithm evaluates the validity of outlier feedback by collecting additional opinions from a random selection of devices. It determines whether to retain or remove the outlier feedback based on majority consensus, ensuring that only reliable feedback contributes to the training dataset. This algorithm randomly selects devices to gather feedback about a trustee and determines whether to remove the outlier feedback based on the majority opinion. The algorithm iterates through each entry of outlier feedback in $OTList$ and randomly selects 10% of the total devices, excluding the trustor itself, as polling devices. For each selected polling device, the algorithm collects feedback regarding the trustee. After gathering feedback from all polling devices, the algorithm counts how many times the original outlier feedback FV appears in the collected feedback for that trustee. If the proportion of the outlier feedback FV is less than 50% of the total feedback collected, the trustor device, along with the corresponding trustee and feedback value, is added to the $RemoveList$.

Algorithm 3 Polling for OTList Validation

Input: OTList (Outlier trustor tuples), N (Total network devices)
Output: RemoveList (Invalid records to remove from dataset)
polling_aggregation ← {} ▷ Initialize empty dictionary
RemoveList ← [] ▷ Initialize empty removal list
num_polling_devices ← $\lceil 0.10 \times N \rceil$ ▷ 10% sample of total devices
available_devices ← `GET_ALL_DEVICES`()
for each (trustee, trustor, FV) IN OTList **do**
    excluded ← `FILTER`(available_devices, trustee, trustor) ▷ Exclude involved devices
    polling_devices ← `RANDOM_SAMPLE`(excluded, num_polling_devices) ▷ Select polling sample
    for each device IN polling_devices **do**
        feedback ← `COLLECT_FEEDBACK`(device, trustee) ▷ Poll device about trustee
        if trustee NOT IN polling_aggregation **then**
            polling_aggregation[trustee] ← [] ▷ Initialize trustee entry
        end if
        polling_aggregation[trustee].APPEND(feedback) ▷ Aggregate poll responses
    end for
    total_polls ← LENGTH(polling_aggregation[trustee])
    count_matches ← COUNT(polling_aggregation[trustee], FV) ▷ Count FV occurrences
    if count_matches / total_polls < 0.5 **then** ▷ Consensus threshold (50%)
        RemoveList.APPEND((trustee, trustor, FV)) ▷ Flag as invalid outlier
    end if
end for
return RemoveList

Similarly, Algorithm 4 implements the polling technique for the $SuspectedList$. For each trustee in the $SuspectedList$, the algorithm randomly selects polling devices and collects their feedback values. If the majority of the feedback indicates dissatisfaction with the trustee (i.e., feedback value equal to 0), that trustee is added to the $MaliciousList$. Finally, the *Analyze* component sends both the $RemoveList$ and the $MaliciousList$ to the Plan component for further processing.

Algorithm 4 Polling for SuspectedList Validation

```
Input: SuspectedList (Potential malicious devices), N (Total network devices)
Output: MaliciousList (Confirmed malicious devices)
MaliciousList ← []                                   ▷ Initialize empty malicious list
polling_aggregation ← {}                             ▷ Initialize polling results dictionary
num_polling_devices ← ⌈0.10 × N⌉                     ▷ 10% device sample size
available_devices ← GET_ALL_DEVICES()
for each trustee IN SuspectedList do
    excluded ← FILTER(available_devices, trustee, trustor)   ▷ Exclude involved devices
    polling_devices ← RANDOM_SAMPLE(excluded, num_polling_devices)   ▷ Select polling sample
    for each device IN polling_devices do
        feedback ← COLLECT_FEEDBACK(device, trustee)         ▷ Poll device about trustee
        if trustee NOT IN polling_aggregation then
            polling_aggregation[trustee] ← []                ▷ Initialize trustee entry
        end if
        polling_aggregation[trustee].APPEND(feedback)        ▷ Store poll response
    end for
    total_polls ← LENGTH(polling_aggregation[trustee])
    count_distrust ← COUNT(polling_aggregation[trustee], 0)  ▷ Count distrust votes
    if count_distrust / total_polls > 0.5 then               ▷ Majority distrust confirmation
        MaliciousList.APPEND(trustee)                        ▷ Flag as malicious
    end if
end for
    return MaliciousList
```

4.3.3 Plan

When the $RemoveList$ or $MaliciousList$ from the *Analyze* component contains entries, the *Plan* component triggers and applies predefined countermeasures—such as blocking, isolating, flagging, or suspension—to the listed devices according to policies in the *Knowledge* component. The proposed trust model's decision engine (*Plan* component) implements a blocking mechanism for identified malicious devices and removes records generated by devices in the $RemoveList$ from the local training dataset. This serves as a solution to mitigate the impact of erroneous feedback values on ML models. This decision is crucial for preventing skewed data from influencing the ML models during the model update process. Finally, the *Plan* component communicates these decisions to the *Execute* component.

4.3.4 Execute

The *Execute* component operationalizes the *Plan* component's decisions through two key enforcement actions: (1) exclusion of all devices in the $MaliciousList$ from service provider selection, and (2) purging of all data contributions from devices in the $RemoveList$ from local training datasets.

4.3.5 Knowledge

The *Knowledge* component of the MAPE-K loop is crucial for informing each component. It encompasses historical feedback data and prior classifications of devices as malicious or benign. This knowledge base is continually updated as new feedback is collected, enabling the system to refine its monitoring, analysis, planning, and execution processes over time.

Integrating the MAPE-K loop within the trust model provides a structured approach to managing device interactions and ensuring network integrity. Through continuous monitoring, adaptive analysis, dynamic planning, and effective execution, the system enhances its ability to respond to malicious behavior while maintaining trust among benign devices.

5. EVALUATION

In this section, we present and analyze the simulated results obtained from extensive experimental evaluations of the proposed trust model.

5.1 Experimental Setup

The simulations were conducted using Python 3.9 on a system with an Intel Core i7 processor and 16GB of RAM. Our network model leverages the comprehensive dataset introduced by Marche et al. [48], which documents device specifications, services, and social relationships for a large-scale IoT ecosystem. The geographic distribution of nodes follows the SmartSantander [49] topology, representing a clustered, multi-region urban environment (Santander, Spain). Node mobility and encounter patterns are governed by the Small World In Motion (SWIM) [50] model to ensure realistic connectivity dynamics.

To evaluate performance under realistic constraints, we sampled a connected sub-network of 250 devices and 70 users, ensuring high interaction probability. We stratified these devices into three tiers based on emulated hardware specifications and energy capabilities:

- CEC-0 (Low-Resource): Modeled after 8-bit/16-bit MCUs (e.g., ATMega series) with 32KB RAM, 50% service availability, and 0.7–1 ms latency.
- CEC-1 (Mid-Range): Modeled after ARM Cortex-M0/M3 MCUs with 128KB RAM, 75% availability, and 0.3–0.7 ms latency.
- CEC-2 (High-End): Modeled after ARM Cortex-M4 or ESP32 MCUs with 512KB RAM, 95% availability, and <0.3 ms latency.

Fog nodes were emulated as edge servers equipped with Quad-core 2.4GHz CPUs and 8GB of RAM, providing the computational power required for ML-based trust aggregation.

Transaction types in this simulation consist of a service request from a requester to a provider, a service response, and a subsequent feedback packet sent to the fog node. To evaluate the model's efficacy, Ground Truth labeling is established at the simulation's onset by assigning each node an inherent nature (Cooperative or Malicious). These labels dictate the following behavioral patterns:

Cooperative devices demonstrated altruistic behavior by consistently delivering high-quality services proportional to their computational resources and energy capacity, while reliably submitting accurate feedback to edge servers. In contrast, malicious devices engaged in various adversarial strategies modeled through distinct service quality patterns. Malicious entities (ME) persistently provided substandard service, while DA attackers selectively maintained high service quality for their friends while degrading performance for others. OOA attackers alternated unpredictably between high and low service quality, whereas OSA attackers initially delivered excellent service to establish a good reputation before abruptly switching to poor performance. WA attackers employed a more sophisticated strategy, initially behaving as ME before systematically abandoning the network after 2,000 rounds and rejoining with new identities to erase their malicious history. Devices that engage in recommendation attacks and are categorized as BMA provide a recommendation value of 0 for those deemed trustworthy. In contrast, devices classified as BSA assign a recommendation value of 1 to devices with low trustworthiness. It is assumed that devices can accurately assess service quality and generate feedback values, but some malevolent devices may provide incorrect feedback to mislead the edge server's ML model. These malicious devices report inverted feedback values, assigning 0 for high-quality providers and 1 for low-quality providers.

While network and communication security fall outside the scope of this investigation, we operate under the assumption of stable node interconnections with deterministic transmission latency. Within the simulated SIoT ecosystem, although fog nodes could theoretically deploy heterogeneous ML architectures, we maintain implementation uniformity by adopting identical local models across all nodes. Furthermore, we presume all fog nodes execute their designated computational tasks with complete integrity, abstracting away potential adversarial behaviors in this layer.

The MLP architecture was selected as the foundation for our ML framework, owing to its demonstrated efficacy in feature extraction and pattern recognition. Each fog node in our system implements a local MLP model constructed using the TensorFlow framework [51] with Keras API integration [52], ensuring both computational efficiency and development consistency. The hyperparameters of these MLP implementations are detailed in Table 3.

Table 3. Details of the MLP Model

Parameter	Value
Number of Layers	3 Fully Connected (Dense) Layers
Input Dimension	8 (Number of Trust Metrics)
Layer 1	32 Neurons, ReLU Activation
Layer 2	16 Neurons, ReLU Activation
Layer 3	4 Neurons, Softmax Activation
Loss Function	Categorical Cross-Entropy
Optimizer	Stochastic Gradient Descent (SGD)
Learning Rate	0.01
Epochs	50
Batch Size	32
Momentum	0.9
K-Fold Cross-Validation	10

The experimental simulation spanned 5,000 discrete operational rounds, with each round representing a complete service transaction cycle.

The MLP models were trained on transactional data collected during an initial 500-round period, with each transaction labeled according to the provider's behavior and operational constraints. Transactions involving malicious providers were categorically labeled as DT, while non-malicious providers received differentiated labels based on service quality: unavailable nodes were marked as UT, available nodes with response times exceeding 0.5 ms as T, and those with sub-0.5 ms responses as HT.

The deployed MLP models subsequently predict service providers' trust levels during operational rounds. Through gossip learning, the models undergo iterative refinement every 500 rounds. This decentralized learning process strategically partitions the dataset, allocating 70% of transactions for model training while reserving 30% for performance validation. To optimize model robustness, we implement 10-fold cross-validation during training, systematically evaluating configurations to identify the topology demonstrating optimal mean accuracy across all folds.

Each device maintains bounded interaction histories to optimize resource utilization while ensuring data relevance. For service requests, devices evaluate up to five candidate providers per query. Recent service feedback is cached with a FIFO buffer limited to 20 entries per provider, while recommendation histories preserve the latest 10 endorsements per service-recommender pair. Complete configuration details are enumerated in Table 4.

Table 4. Description of simulation parameters

Parameter	Description	Value
α	Parameter to increase Perseverance metric	0.1
β	Parameter to decrease Perseverance metric	0.2
γ	Minimum transactions for direct trust computation	5
ρ	Parameter to prioritizes feedbacks	0.4
R_{max}	Maximum valid feedback rounds	1000
k	Number of selected peers in Gossip learning	2

5.2 Evaluation Metrics

To assess the effectiveness of the proposed trust model, several key metrics are considered: Success Rate, Accuracy, F1-Measure, and Loss.

Success Rate captures how often services meet user expectations, showing the percentage of transactions that received favorable feedback out of all completed interactions. It serves as a direct indicator of service provider reliability from the user's perspective. Accuracy gauges how precisely the system predicts trust levels, measuring the percentage of cases where the forecasted trust classification matches the actual observed outcome after transaction completion. This metric reflects the model's ability to correctly anticipate service quality. F1-Measure is a metric that balances Precision and Recall, computed as the harmonic mean of the two. Precision is defined as the ratio of true positive predictions to the total number of positive predictions made by the model, while Recall is the ratio of true positives to the total number of actual positive instances. Given the class imbalance, we utilized the micro-averaged F-Measure for evaluation. Lastly, Loss denotes the mean error of the predicted trust levels across all transactions. The efficacy of a trust model is demonstrated by strong performance across key metrics: elevated Accuracy and F1-Measure scores reflect precise trust predictions, while a high Success Rate indicates consistent service satisfaction. Concurrently, diminishing Loss values signify improved model convergence and reliability. Together, these metrics provide a comprehensive assessment of trust management effectiveness.

5.3 Experimental Results

This section presents a comprehensive performance evaluation of the proposed trust model, including rigorous benchmarking against two state-of-the-art approaches in the field. Through systematic analysis, we demonstrate the model's effectiveness across key metrics while highlighting its comparative advantages.

Resilience To Trust-Based Attacks: In the first phase, the resilience of the proposed trust model is assessed against different trust-based attacks. For this experiment, 25% of the devices were deliberately configured as malicious. Each attack type (namely, ME, DA, OOA, OSA, and WA) was evaluated separately. Metrics such as accuracy, F-Measure, Success Rate, and loss were recorded and plotted every 500 rounds, as illustrated in Figure 3.

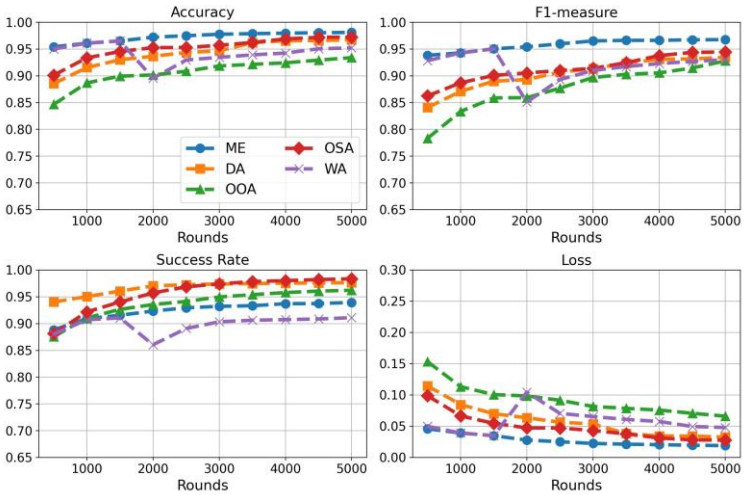

Figure 3. Accuracy, F-Measure, Success Rate, and Loss rates in the presence of 25% malicious devices simulating ME, DA, OOA, OSA, and WA

The proposed trust model demonstrates strong resilience against all these attacks, achieving an Accuracy, F-Measure, and Success Rate of over 0.9, while Loss stays below 0.07 for each type of attack. The predictable behavioral patterns of ME providers facilitate reliable detection, resulting in the best performance metrics for Accuracy, F-Measure, and Loss in this category. However, ME and WA record the lowest Success Rate. This is attributed to the fact that ME attackers consistently engage in malicious activities, while other attack types display malicious behavior either randomly or under specific conditions. In the WA scenario, the departure and subsequent re-entry of malicious devices with new identities at round 2,000 leads to a

sharp decline in Accuracy, F-Measure, and Success Rate, along with an increase in Loss. However, this drop is later mitigated by the successful identification of these malicious devices.

Resilience To Recommendation Attack: In addition to the previously discussed attacks, recommendation-driven exploits (including BMA, BSA, and SPA) pose additional threats to trust management systems. While SPA involves malicious devices artificially inflating their trustworthiness through self-recommendations, our model inherently neutralizes this strategy by disregarding all self-originating endorsements during trust computation. To evaluate robustness, we conducted comparative simulations under three adversarial conditions:

- Baseline: 25% ME attackers
- Scenario A: 25% ME + 25% BMA attackers
- Scenario B: 25% ME + 25% BSA attackers

As demonstrated in Figure 4, the results indicate minimal differences in Accuracy, F-Measure, Success Rate, and Loss across all three scenarios. This resilience stems from the recommendation filtering protocol detailed in Section 4.1, which systematically excludes untrustworthy inputs before they influence trust calculations.

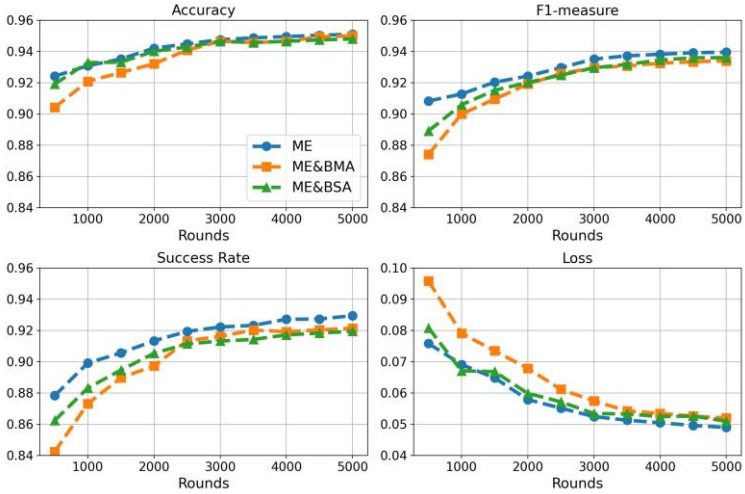

Figure 4. Accuracy, F-Measure, Success Rate, and Loss rates in the presence of 25% malicious devices simulating ME, ME and BMA, and ME and BSA

Resilience to SA: Although SA is not explicitly modeled as a standalone attack, the BMA and BSA scenarios effectively simulate a coordinated collusion attack, which is the primary objective of a Sybil swarm in a trust system. The model consistently maintains high accuracy and F-Measure in these scenarios demonstrate that the credibility-weighted recommendation filtering (Section 4.1) successfully neutralizes the impact of coordinated, fraudulent inputs. This result suggests inherent robustness against SAs that target the recommendation subsystem, as long as the Sybil nodes cannot quickly establish high credibility scores, which is inherently difficult under the social relationship constraints of SIoT. Furthermore, the following features of the proposed trust model significantly increase the cost and complexity of mounting a successful SA:

1. Social Relationship Barrier: The requirement for devices to establish formal SIoT relationships (OOR, POR, CLOR, etc.) acts as a natural barrier. Forging believable, verifiable relationships (e.g., co-location or co-ownership) for a large number of Sybils is non-trivial and resource-intensive.
2. Credibility-Weighted Defense: The recommendation metric inherently mitigates Sybil-based collusion by weighting opinions based on the historical accuracy of the recommender. A swarm of new, unproven Sybil nodes would have low credibility weights, marginalizing their collective impact.
3. MAPE-K Anomaly Detection: The Analyze component's IQR-based outlier detection (Algorithm 1) and behavioral polling (Algorithms 2 & 3) can flag coordinated anomalies. For example, a sudden influx of new nodes or a block of nodes providing identical, outlier feedback could be flagged for deeper inspection, potentially revealing Sybil behavior.
4. Resource-Aware Context: The CEC metric provides context. A sudden appearance of many high-capability (CEC-2) nodes might be statistically anomalous and trigger scrutiny.

While these features significantly strengthen the model's security posture against SAs, future work could be further enhanced to provide even more specialized resistance.

Resilience to Increasing Malicious Devices: We evaluated the proposed trust model's resilience by systematically increasing the proportion of ME devices from 10% to 90% in 10% increments. As shown in Figure 5, the model demonstrates remarkable robustness across all tested scenarios.

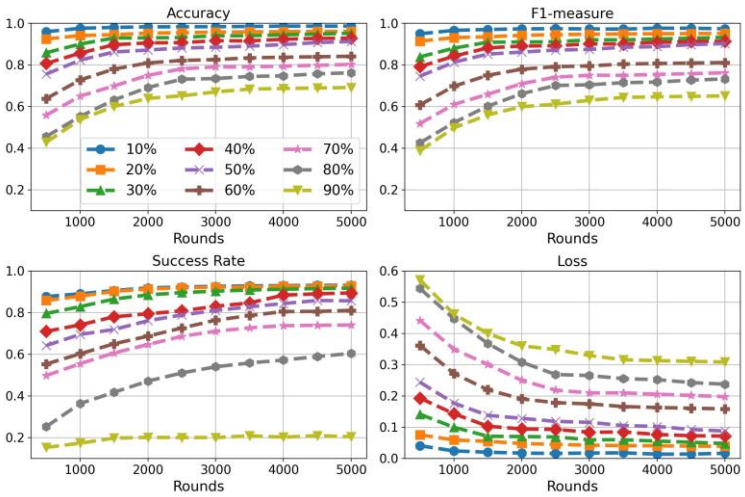

Figure 5. Comparison of Accuracy, F-Measure, Success Rate, and Loss as the percentage of malicious devices increases

Key performance metrics maintain strong thresholds even under extreme adversarial conditions: Accuracy consistently exceeds 0.85, F-Measure remains above 0.8, and Loss stays below 0.32. Notably, the Success Rate preserves a minimum value of 0.7 despite 70% of devices being malicious, confirming the model's ability to sustain reliable performance and maintain convergence even when facing overwhelming adversarial presence.

Feedback Validation Mechanism: The next simulation aims to evaluate the effectiveness of the feedback validation mechanism in identifying and removing incorrect feedback submitted by malicious devices from the learning dataset. This simulation involves a scenario with 25% ME attackers and 25% of attackers providing false feedback (positive ratings for malicious devices and negative ratings for benign ones). The evaluation is conducted twice: once with the detection and removal of erroneous feedback and once without it. The results, illustrated in Figure 6, demonstrate that feedback validation significantly enhances the trust model across all four metrics.

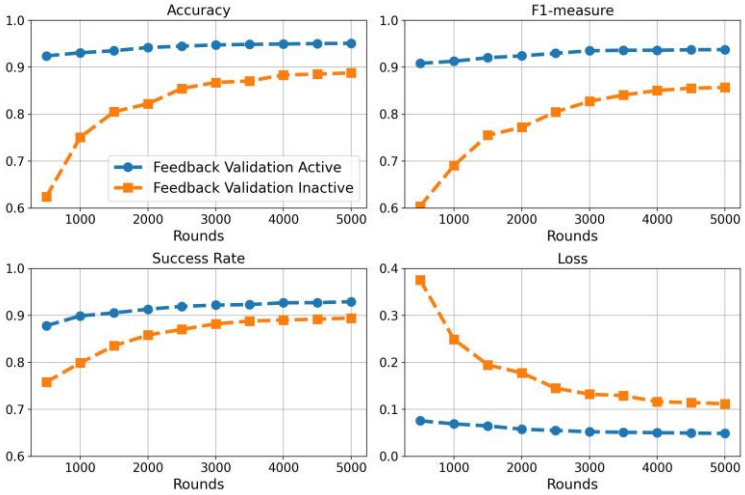

Figure 6. Comparison of Accuracy, F-Measure, Success Rate, and Loss with Feedback Validation Active versus Inactive

Comparison with Alternative Versions: To demonstrate the effectiveness of the MAPE-K loop and gossip learning within the proposed trust model in mitigating trust attacks, we compare the proposed trust model (PTM) with three alternative versions. PTM-v1 represents the proposed trust model implemented without gossip learning, meaning local MLP models do not collaborate. PTM-v2 is the model that operates without the detection and removal of malicious devices and erroneous feedback through the MAPE-K loop. PTM-v3 is the variant that lacks both gossip learning and the MAPE-K control loop. Our experimental evaluation encompassed five distinct attack scenarios (ME, DA, OOA, WA, and OSA) with 25% of devices providing false feedback in each case. The results are illustrated in Figure 7.

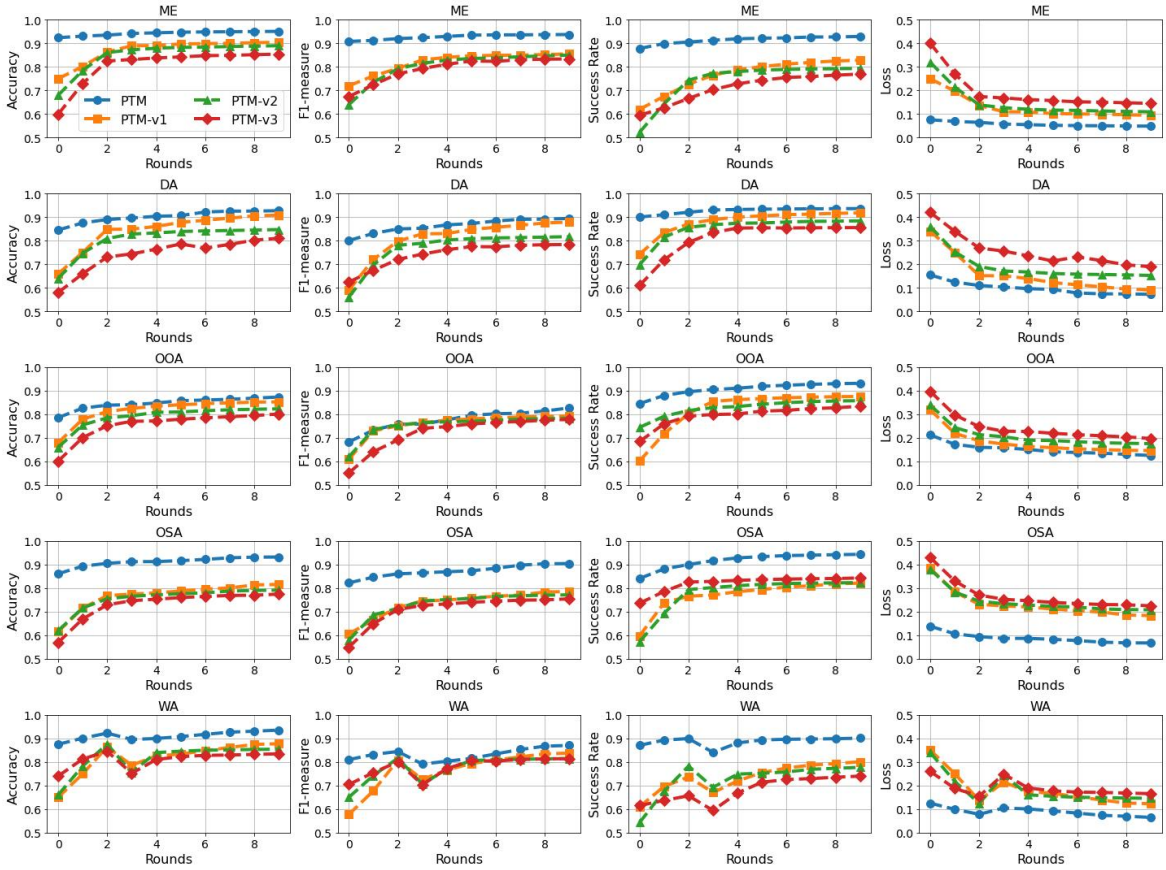

Figure 7. Performance comparison of the proposed trust model (PTM) and its variants (PTM-v1, PTM-v2, PTM-v3) for different trust attacks

The experimental results demonstrate that integrating gossip learning with the MAPE-K control loop significantly enhances the PTM's performance across all evaluated trust attacks. Compared to PTM-v1, the gossip learning mechanism yields an average 8% improvement in both Accuracy and F-Measure, along with an 11% higher Success Rate and 8% reduced Loss, attributable to its distributed knowledge aggregation from multiple fog nodes. Further analysis against PTM-v2 reveals additional gains from the MAPE-K framework, including 9% greater Accuracy, 8% improved F-Measure, 11% enhanced Success Rate, and 9% lower Loss. These improvements stem from two synergistic mechanisms: the control loop's dynamic blocking of malicious devices, which reduces untrustworthy transactions, coupled with its feedback purification that removes deceptive inputs to refine ML model predictions. Thus, the MAPE-K control loop plays a crucial role in enhancing the overall effectiveness of the proposed trust model. Overall, the PTM shows a 13% increase in Accuracy, a 10% rise in F-Measure, a 14% improvement in the overall Success Rate, and a 12% reduction in Loss compared to PTM-v3. This underscores the effectiveness of both gossip learning and the MAPE-K control loop in enhancing the robustness of the PTM against trust attacks.

Comparison with State-of-the-Art Models: The final set of simulations seeks to assess how the proposed trust model performs when 25% of nodes participate in trust attacks, comparing its effectiveness with two established models introduced by Marche et al. [20] and Moeinaddini et al. [14]. This analysis specifically contrasts our approach with two state-of-the-art alternatives: the incremental SVM-based framework by Marche et al. [20] and the decentralized trust model by Moeinaddini et al. [14], both employing ML techniques for SIoT trust computation.

Marche et al.'s [20] methodology implements a device-wide SVM architecture that dynamically assesses trustworthiness through continuous evaluation of three key dimensions: goodness, usefulness, and behavioral perseverance. The trust computation mechanism in this model is distributed, occurring directly on the devices. On the other hand, Moeinaddini et al. [14] introduced a decentralized trust model that employs local MLP models within fog nodes to predict trust levels. These local models are periodically updated via federated learning based on transaction results reported by devices, with malicious devices being blocked using an adaptive threshold.

To establish rigorous benchmarking conditions, we meticulously reimplemented both baseline architectures in strict adherence to their original specifications while maintaining consistent simulation parameters across all evaluations. The experimental framework preserves identical services, social object relationships, and request sequences to enable direct model comparison. Our quantitative assessment employs four principal metrics (Accuracy, F-Measure, Success Rate, and Loss) measured over 5,000 operational rounds across a 250-device network topology. We configured all three trust models under identical attack conditions, with 25% of nodes designated as malicious. The assessment examines five distinct attacks: ME, DA, OOA, OSA, and WA. Table 5 presents the mean values from 30 runs of this experiment, with the p-values computed between the proposed trust model and the other two models.

Table 5. Robustness analysis of the proposed trust model in comparison with state-of-the-art approaches across different attack types.

Attack type	Paper	Accuracy	P-value	F- Measure	P-value	Success Rate	P-value	Loss	P-value
ME	Marche et al. [20]	0.851	<.001	0.824	<.001	0.798	<.001	0.149	<.001
	Moeinaddini et al. [14]	0.939	<.01	0.898	<.001	0.844	<.001	0.061	<.01
	Proposed	**0.951**	-	**0.938**	-	**0.929**	-	**0.049**	-
DA	Marche et al. [20]	0.822	<.001	0.791	<.001	0.815	<.001	0.178	<.001
	Moeinaddini et al. [14]	0.910	<.01	0.854	<.001	0.915	<.01	0.090	<.01
	Proposed	**0.927**	-	**0.895**	-	**0.936**	-	**0.073**	-
OOA	Marche et al. [20]	0.811	<.001	0.783	<.001	0.853	<.001	0.189	<.001
	Moeinaddini et al. [14]	0.832	<.001	0.805	<.01	0.908	<.001	0.168	<.001
	Proposed	**0.874**	-	**0.828**	-	**0.932**	-	**0.126**	-
OSA	Marche et al. [20]	0.861	<.001	0.814	<.001	0.853	<.001	0.139	<.001
	Moeinaddini et al. [14]	0.912	<.01	0.866	<.001	0.892	<.001	0.088	<.01
	Proposed	**0.932**	-	**0.904**	-	**0.944**	-	**0.068**	-
WA	Marche et al. [20]	0.833	<.001	0.854	<.001	0.774	<.001	0.167	<.001
	Moeinaddini et al. [14]	0.900	<.001	0.858	<.01	0.864	<.001	0.100	<.001
	Proposed	**0.935**	-	**0.870**	-	**0.901**	-	**0.065**	-

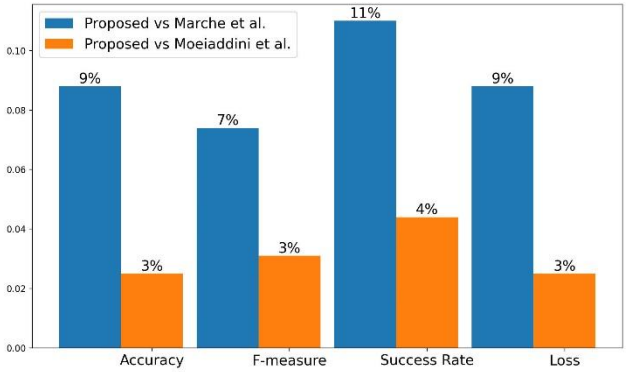

Figure 8. Comparison of the Mean Improvement of the Proposed Trust Model Against State-of-the-Art Models

The comparison revealed that the proposed model outperformed the other two models, yielding a p-value of less than 0.01 in the worst case, indicating that the differences in mean values are statistically significant. These low p-values strongly indicate that the proposed method delivers statistically superior results compared to both alternative models. Based on the results, the proposed model consistently outperforms the others across all evaluation parameters. The mean improvements for each metric are illustrated in Figure 8. When compared to Marche et al. [20], the proposed model shows an average increase of 9% in Accuracy, 7% in F-Measure, 11% in Success Rate, and an 8% reduction in Loss across all attack types. In comparison to Moeinaddini et al. [14], there is an average increase of 3% in Accuracy and F-Measure, a 4% increase in Success Rate, and a 3% reduction in Loss. This superior performance is achieved by combining the MAPE-K loop's proactive monitoring with the collaborative nature of gossip learning. This synergy enables fog nodes to autonomously detect attacks and erroneous data while collectively refining trust knowledge, ensuring seamless adaptation to dynamic network environments without the need for centralized coordination.

5.4 Discussion: Scalability, Efficiency, and Limitations

To assess the practical deployability of the proposed trust model, we examine its parameter sensitivity, scalability characteristics, communication overhead, and remaining limitations.

5.4.1 Sensitivity Analysis of Gossip Learning Parameters

To evaluate the robustness of the proposed trust model to variations in gossip interval and peer selection size (k), we conducted a sensitivity analysis under the 25% WA attackers combined with 25% devices providing false feedback. WA is selected because its abrupt departure and re-entry at round 2,000 creates a sudden accuracy drop, providing the strict test of gossip learning's adaptation and convergence speed. The gossip interval was varied over {200, 500, 1000} rounds, and k over {2, 4, 6}. Table 6 reports the steady-state Accuracy and Loss averaged over the 5,000-round simulation. Communication overhead is normalized to our baseline configuration (500 rounds, $k = 2$). Convergence rounds are measured as the number of simulation rounds required to reach 90% of final steady-state accuracy after the WA attack's re-entry event at round 2,000. All values are averages over 30 independent simulation runs.

Table 6. Impact of gossip interval and peer selection size k on model performance under WA attack with 25% malicious devices and 25% false feedback

Gossip Interval (Rounds)	Peer Selection Size (k)	Accuracy	Loss	Communication Overhead (Relative)	Convergence Rounds (to 90% of accuracy)
200	2	0.946	0.054	2.5x	480
200	4	0.948	0.052	5.0x	460
200	6	0.949	0.051	7.5x	455
500	2	0.935	0.065	1.0x (baseline)	1,020
500	4	0.937	0.063	2.0x	980
500	6	0.938	0.062	3.0x	960
1000	2	0.921	0.079	0.5x	1,850
1000	4	0.924	0.076	1.0x	1,780
1000	6	0.925	0.075	1.5x	1,750

The results demonstrate that the proposed trust model maintains Accuracy ≥ 0.921 and Loss ≤ 0.079 across all parameter combinations, confirming that performance is not brittle to gossip hyperparameter choices. Decreasing the gossip interval from 500 to 200 rounds yields only a marginal improvement (+1.1% accuracy) but increases communication overhead by 150%. Conversely, increasing the interval to 1,000 rounds reduces overhead by 50% but degrades accuracy by −1.4% and delays post-attack recovery (convergence rounds increase from ~1,020 to ~1,850). Regarding peer selection size, increasing k from 2 to 4 provides modest gains (<0.3% accuracy improvement) at the cost of linearly increasing overhead, with diminishing returns beyond k = 4. Based on this sensitivity analysis, the original configuration (500 rounds, k = 2) represents a balanced trade-off between accuracy, resilience, and communication efficiency.

5.4.2 **Scalability Mechanisms**

The proposed trust model addresses scalability and efficiency through two primary mechanisms:

- **Communication Overhead of Gossip Learning:** To mitigate communication overhead, the gossip protocol avoids global synchronization by employing a randomized peer-selection strategy, communicating with only a subset of k peers. The gossip learning process is triggered periodically (e.g., every 500 rounds) rather than per transaction, which significantly reduces bandwidth consumption. Consequently, the communication overhead scales as $O(k \cdot M)$, where M is the number of fog nodes, rather than $O(M^2)$, ensuring viability in large-scale deployments. To provide concrete quantification, each fog node's MLP model comprises 884 trainable parameters (Layer 1: 8×32 + 32 = 288; Layer 2: 32×16 + 16 = 528; Layer 3: 16×4 + 4 = 68). Using 32-bit floating-point representation, the raw model size is approximately 3.5 KB (884 × 4 bytes). With k = 2 peers selected per gossip round, each fog node transmits 7.0 KB and receives 7.0 KB, totaling 14.0 KB per gossip round. Over a 5,000-round simulation (10 gossip rounds), each fog node incurs approximately 140 KB of communication overhead. This overhead is negligible relative to typical fog node memory and bandwidth capacities, confirming the efficiency of the gossip-based collaborative learning mechanism.
- **Distributed Processing and Resource Management**: The workload for trust calculation and MAPE-K loops is distributed across M fog nodes, each managing a localized cluster. This keeps the complexity of IQR-based outlier detection within the hardware limits of typical fog gateways. Furthermore, the use of incremental learning minimizes computational strain by refining the ML model with new data points instead of full retraining.

5.4.3 Other Limitations and Future Work

Despite these architectural strengths, certain constraints warrant further investigation. Future research will address stochastic network latency, extreme device churn, cross-domain trust federation, and advanced Sybil-resistance mechanisms. Additionally, moving beyond uniform weighting in gossip aggregation toward reputation-based weighting represents a promising direction for improved robustness.

6. CONCLUSION

This paper presented GossipTrust, a novel decentralized trust model for social IoT that addresses the critical challenges of scalability, dynamic trust evaluation, and resource constraints. By leveraging fog-based ML models, our trust model distributes trust computation efficiently, eliminating single points of failure while minimizing overhead on IoT devices. The integration of a MAPE-K control loop enables real-time detection of trust attacks and erroneous data through continuous monitoring and adaptive response. Furthermore, gossip learning allows fog nodes to collaboratively refine trust knowledge without centralized coordination, ensuring adaptability to evolving network conditions. Experimental results demonstrate GossipTrust's superiority over baseline models, with 13% higher accuracy, 10% improved F-measure, 14% greater success rate, and 12% reduced loss. These gains validate its effectiveness in balancing security and efficiency. To the best of our knowledge, this is the first work to combine MAPE-K-driven adaptation with gossip learning for SIoT trust management, offering a scalable and dynamic solution for real-world deployments. Future research will focus on establishing cross-domain trust federation, and developing more robust mechanisms for Sybil-resistance. Additionally, we plan to enhance the aggregation process of gossip learning beyond the uniform weighting approach to further improve model robustness and accuracy in gossip-based trust propagation.

7. REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] P. Selvakumar, S. Geetha, N. Kaya, P. S. Chandel, and P. Srivastava, "Social Internet of Things (SIoT)," in *Analyzing Privacy and Security Difficulties in Social Media: New Challenges and Solutions*, IGI Global Scientific Publishing, 2025, pp. 39–62.

[3] M. Nitti, R. Girau, and L. Atzori, "Trustworthiness management in the social internet of things," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 5, pp. 1253–1266, 2013.

[4] L. Wei, J. Wu, C. Long, and B. Li, "On designing context-aware trust model and service delegation for social internet of things," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4775–4787, 2020.

[5] J. Wang, X. Jing, Z. Yan, Y. Fu, W. Pedrycz, and L. T. Yang, "A survey on trust evaluation based on machine learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 5, pp. 1–36, 2020.

[6] I. Singhal, K. Chavan, A. Mane, S. Kelkar, and S. Deshpande, "Mitigating SIoT attacks in Smart Medical Systems: A Machine Learning based Approach," in *2022 IEEE Global Conference on Computing, Power and Communication Technologies (GlobConPT)*, IEEE, 2022, pp. 1–6.

[7] S. Sagar, A. Mahmood, M. Sheng, M. Zaib, and W. Zhang, "Towards a machine learning-driven trust evaluation model for social internet of things: A time-aware approach," in *MobiQuitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2020, pp. 283–290.

[8] Y. Wen, Z. Xu, R. Zhi, and J. Chen, "Trust Prediction Model Based on Deep Learning in Social Internet of Things," in *IoT as a Service: 6th EAI International Conference, IoTaaS 2020, Xi'an, China, November 19–20, 2020, Proceedings 6*, Xi'an, China: Springer, 2021, pp. 557–570.

[9] S. Sagar, A. Mahmood, K. Wang, Q. Z. Sheng, J. K. Pabani, and W. E. Zhang, "Trust–SIoT: Toward Trustworthy Object Classification in the Social Internet of Things," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1210–1223, 2023, doi: 10.1109/TNSM.2023.3247831.

[10] W. Abdelghani, C. A. Zayani, I. Amous, and F. Sèdes, "Trust evaluation model for attack detection in social internet of things," in *Risks and Security of Internet and Systems: 13th International Conference, CRiSIS 2018, Arcachon, France, October 16–18, 2018, Revised Selected Papers 13*, Springer, 2019, pp. 48–64.

[11] W. Abdelghani, I. Amous, C. A. Zayani, F. Sèdes, and G. Roman-Jimenez, "Dynamic and scalable multi-level trust management model for Social Internet of Things," *J. Supercomput.*, vol. 78, no. 6, pp. 8137–8193, 2022.

[12] B. Jafarian, N. Yazdani, and M. S. Haghighi, "Discrimination-aware trust management for social internet of things," *Computer Networks*, vol. 178, p. 107254, 2020.

[13] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer (Long. Beach. Calif).*, vol. 36, no. 1, pp. 41–50, 2003.

[14] E. Moeinaddini, E. Nazemi, and A. Shahraki, "A new approach on self-adaptive trust management for social Internet of Things," *Computer Networks*, vol. 263, p. 111187, 2025.

[15] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *IFIP International Conference on Distributed Applications and Interoperable Systems*, 2019, pp. 74–90.

[16] L. Wulfert, N. Asadi, W.-Y. Chung, C. Wiede, and A. Grabmaier, "Adaptive decentralized federated gossip learning for resource-constrained iot devices," in *Proceedings of the 4th International Workshop on Distributed Machine Learning*, 2023, pp. 27–33.

[17] Z. Lin and L. Dong, "Clarifying trust in social internet of things," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 2, pp. 234–248, 2017.

[18] R. M.S., S. Pattar, R. Buyya, V. K.R., S. S. Iyengar, and L. M. Patnaik, "Social Internet of Things (SIoT): Foundations, thrust areas, systematic review and future directions," *Comput. Commun.*, vol. 139, pp. 32–57, 2019.

[19] Y. Saleem, N. Crespi, M. H. Rehmani, R. Copeland, D. Hussein, and E. Bertin, "Exploitation of social IoT for recommendation services," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016, pp. 359–364.

[20] C. Marche and M. Nitti, "Trust-related attacks and their detection: A trust management model for the social IoT," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3297–3308, 2020.

[21] C. A. and A. I. and S. F. Abdelghani Wafa and Zayani, "Trust Management in Social Internet of Things: A Survey," in *Social Media: The Good, the Bad, and the Ugly*, Swansea, UK: Springer International Publishing, 2016, pp. 430–441.

[22] R. K. Chahal, N. Kumar, and S. Batra, "Trust management in social Internet of Things: A taxonomy, open issues, and challenges," *Comput. Commun.*, vol. 150, pp. 13–46, 2020.

[23] A. Rajan, J. Jithish, and S. Sankaran, "Sybil attack in IOT: Modelling and defenses," in *2017 International conference on advances in computing, communications and informatics (ICACCI)*, IEEE, 2017, pp. 2323–2327.

[24] Y. Brun *et al.*, "Engineering self-adaptive systems through feedback loops," *Software engineering for self-adaptive systems*, pp. 48–70, 2009.

[25] D. Weyns, "Software engineering of self-adaptive systems," in *Handbook of Software Engineering*, Cham, Switzerland: Springer International Publishing, 2019.

[26] S. Asiri and A. Miri, "An IoT trust and reputation model based on recommender systems," in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, IEEE, 2016, pp. 561–568.

[27] R. Chen, F. Bao, and J. Guo, "Trust-based service management for social internet of things systems," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 6, pp. 684–696, 2015.

[28] G. Chen, F. Zeng, J. Zhang, T. Lu, J. Shen, and W. Shu, "An adaptive trust model based on recommendation filtering algorithm for the Internet of Things systems," *Computer Networks*, vol. 190, p. 107952, 2021.

[29] M. Bahutair, A. Bougeuttaya, and A. G. Neiat, "Adaptive trust: Usage-based trust in crowdsourced IoT services," in *2019 IEEE international conference on web services (ICWS)*, Milan, Italy, 2019, pp. 172–179.

[30] R. Ormándi, I. Hegedűs, and M. Jelasity, "Gossip learning with linear models on fully distributed data," *Concurr. Comput.*, vol. 25, no. 4, pp. 556–571, 2013.

[31] A. Tundo, F. Filippini, F. Regonesi, M. Ciavotta, and M. Savi, "Decentralized Edge Workload Forecasting With Gossip Learning," *IEEE Transactions on Network and Service Management*, 2025.

[32] C. C. Sobin, "A survey on architecture, protocols and challenges in IoT," *Wirel. Pers. Commun.*, vol. 112, no. 3, pp. 1383–1429, 2020.

[33] C. Goswami, R. Raman, B. G. Pillai, R. Singh, B. Dhanne, and D. Kapila, "Implementation of a Machine Learning-based Trust Management System in Social Internet of Things," in *2022 5th International Conference on Contemporary Computing and Informatics (IC3I)*, 2022, pp. 1586–1590.
[34] M. Masmoudi, W. Abdelghani, I. Amous, and F. Sèdes, "Deep learning for trust-related attacks detection in social internet of things," in *Advances in E-Business Engineering for Ubiquitous Computing: Proceedings of the 16th International Conference on e-Business Engineering (ICEBE 2019)*, Springer, 2020, pp. 389–404.
[35] S. Sagar, A. Mahmood, Q. Z. Sheng, and W. E. Zhang, "Trust computational heuristic for social Internet of Things: A machine learning-based approach," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, IEEE, 2020, pp. 1–6.
[36] R. Magdich, H. Jemal, C. Nakti, and M. Ben Ayed, "An efficient Trust Related Attack Detection Model based on Machine Learning for Social Internet of Things," in *2021 International Wireless Communications and Mobile Computing (IWCMC)*, Harbin City, China, 2021, pp. 1465–1470.
[37] R. Magdich, H. Jemal, and M. Ben Ayed, "Context-awareness trust management model for trustworthy communications in the social Internet of Things," *Neural Comput. Appl.*, pp. 1–26, 2022.
[38] S. Dramé-Maigné, M. Laurent, L. Castillo, and H. Ganem, "Centralized, distributed, and everything in between: Reviewing access control solutions for the IoT," *ACM Computing Surveys (CSUR)*, vol. 54, no. 7, pp. 1–34, 2021.
[39] M. Amiri-Zarandi, R. A. Dara, and X. Lin, "SIDS: A federated learning approach for intrusion detection in IoT using Social Internet of Things," *Computer Networks*, vol. 236, p. 110005, 2023.
[40] G. Rjoub, O. A. Wahab, J. Bentahar, and A. Bataineh, "Trust-driven reinforcement selection strategy for federated learning on IoT devices," *Computing*, vol. 106, no. 4, pp. 1273–1295, 2024.
[41] L. Bi, T. Muazu, and O. Samuel, "IoT: A Decentralized Trust Management System Using Blockchain-Empowered Federated Learning," *Sustainability*, vol. 15, no. 1, 2023.
[42] K. A. Awan, I. Ud Din, M. Zareei, A. Almogren, B. Seo-Kim, and J. A. Pérez-Díaz, "Securing IoT With Deep Federated Learning: A Trust-Based Malicious Node Identification Approach," *IEEE Access*, vol. 11, pp. 58901–58914, 2023.
[43] M. H. ur Rehman, A. M. Dirir, K. Salah, E. Damiani, and D. Svetinovic, "TrustFed: A Framework for Fair and Trustworthy Cross-Device Federated Learning in IIoT," *IEEE Trans. Industr. Inform.*, vol. 17, no. 12, pp. 8485–8494, 2021.
[44] E. Baccarelli, M. Scarpiniti, P. G. V. Naranjo, and L. Vaca-Cardenas, "Fog of social IoT: When the fog becomes social," *IEEE Netw.*, vol. 32, no. 4, pp. 68–80, 2018.
[45] U. Jayasinghe, G. M. Lee, T.-W. Um, and Q. Shi, "Machine learning based trust computational model for IoT services," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 39–52, 2018.
[46] C. Bormann, M. Ersue, and A. Keranen, "Terminology for constrained-node networks," 2014.
[47] Y. Dodge, "Interquartile Range," in *The Concise Encyclopedia of Statistics*, New York, NY: Springer New York, 2008, pp. 266–267.
[48] C. Marche, L. Atzori, and M. Nitti, "A dataset for performance analysis of the social internet of things," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, IEEE, 2018, pp. 1–5.
[49] L. Sanchez et al., "SmartSantander: IoT experimentation over a smart city testbed, " *Computer networks*., vol. 61, pp. 217-238, 2014.
[50] S. Kosta, A. Mei, and J. Stefa, "Small world in motion (SWIM): Modeling communities in ad-hoc mobile networking," in *2010 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2010, pp. 1–9.
[51] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, Savannah, GA, 2016, pp. 265–283.
[52] A. Gulli and S. Pal, *Deep Learning with Keras*. Birmingham,UK: Packt Publishing, 2017.