Computer Modeling in
Engineering & Sciences

Tech Science Press

Doi:10.32604/cmes.2025.066589

ARTICLE

Deep Auto-Encoder Based Intelligent and Secure Time Synchronization Protocol (iSTSP) for Security-Critical Time-Sensitive WSNs

Ramadan Abdul-Rashid[1], **Mohd Amiruddin Abd Rahman**[1,*] **and Abdulaziz Yagoub Barnawi**[2]

[1]Department of Physics, Faculty of Science, Universiti Putra Malaysia, UPM, Serdang, 43400, Malaysia
[2]Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran, 31261, Saudi Arabia

*Corresponding Author: Mohd Amiruddin Abd Rahman. Email: mohdamir@upm.edu.my

Received: 11 April 2025; Accepted: 19 August 2025; Published: 30 September 2025

ABSTRACT: Accurate time synchronization is fundamental to the correct and efficient operation of Wireless Sensor Networks (WSNs), especially in security-critical, time-sensitive applications. However, most existing protocols degrade substantially under malicious interference. We introduce *iSTSP*, an Intelligent and Secure Time Synchronization Protocol that implements a four-stage defense pipeline to ensure robust, precise synchronization even in hostile environments: (1) trust preprocessing that filters node participation using behavioral trust scoring; (2) anomaly isolation employing a lightweight autoencoder to detect and excise malicious nodes in real time; (3) reliability-weighted consensus that prioritizes high-trust nodes during time aggregation; and (4) convergence-optimized synchronization that dynamically adjusts parameters using theoretical stability bounds. We provide rigorous convergence analysis including a closed-form expression for convergence time, and validate the protocol through both simulations and real-world experiments on a controlled 16-node testbed. Under Sybil attacks with five malicious nodes within this testbed, iSTSP maintains synchronization error increases under 12% and achieves a rapid convergence. Compared to state-of-the-art protocols like TPSN, SE-FTSP, and MMAR-CTS, iSTSP offers 60% faster detection, broader threat coverage, and more than 7 times lower synchronization error, with a modest 9.3% energy overhead over 8 h. We argue this is an acceptable trade-off for mission-critical deployments requiring guaranteed security. These findings demonstrate iSTSP's potential as a reliable solution for secure WSN synchronization and motivate future work on large-scale IoT deployments and integration with energy-efficient communication protocols.

KEYWORDS: Time-sensitive wireless sensor networks (TS-WSNs); secure time synchronization protocol; trust-based authentication; autoencoder model; deep learning; malicious node detection; Internet of Things; energy-efficient communication protocols

1 Introduction

WSNs have transformed the way we collect and handle data in a variety of disciplines, from environmental monitoring to industrial automation. These networks are made up of little, resource-constrained sensor nodes dispersed across a certain area, cooperating to gather and transmit data to a central sink or gateway [1]. Time synchronization, which requires the nodes to maintain a consistent sense of time to successfully coordinate their operations, is one of the fundamental conditions for the accurate operation of WSNs.

Time synchronization is essential in WSNs for a number of reasons. It is essential to many applications, including target tracking, event detection, and data aggregation. It also makes it possible to combine accurate data in an energy-efficient manner. Many time synchronization techniques have been developed and implemented in WSNs with the goal of achieving a balance between synchronization accuracy and resource efficiency [2]. Time synchronization protocols allow sensor nodes to align their internal clocks, promoting data coherence and efficient network administration. The inherent characteristics of WSNs make them vulnerable to security attacks. These characteristics include [3]:

1. Limited resources: The processing speed, memory, and energy of sensor nodes are all finite. This makes elaborate security methods harder to implement.
2. Exposed and hostile environments: WSNs are frequently deployed in open and dangerous locations, such as battlefields or disaster zones. As a result, they are subject to physical attacks and eavesdropping.
3. Absence of central security control: The security management in WSNs is not centralized. As a result, it is challenging to implement security regulations and to recognize and stop attacks.

One of the most important security threats in the WSN network is the spoofing of time synchronization protocols. By manipulating the message timestamps, an attacker can disrupt the normal operation of the network, such as by injecting fake data, delaying the message, or replaying old messages. To secure the time synchronization protocols in WSN, some studies [4–7] have suggested various techniques to combat attacks, such as:

1. Using cryptography to secure communications between sensor nodes.
2. Applying algorithms to validate the authenticity of sensor nodes in a sensor field.
3. Implementing secure detection and routing to prevent attackers from injecting malicious messages into the network.

To address these challenges, we propose the Intelligent and Secure Time Synchronization Protocol (iSTSP), a novel solution that integrates deep learning-based anomaly detection with adaptive control theory in a structured four-stage defense pipeline. iSTSP's architecture systematically mitigates threats while maintaining high synchronization accuracy:

1. Trust Preprocessing—Filters out untrusted nodes before synchronization begins using dynamic trust scoring, reducing the attack surface by 63%.
2. Anomaly Isolation—Detects and excludes compromised nodes in real time using an autoencoder-based model, achieving 92% Sybil attack detection within a single synchronization round.
3. Reliability-Weighted Consensus—Prioritizes high-trust nodes during time aggregation, limiting malicious influence by 7.2× compared to unweighted methods.
4. Convergence-Optimized Synchronization—Dynamically adjusts synchronization parameters using a closed-form convergence time model, reducing synchronization error by 38% under attack conditions.

Our key contributions include:

1. A four-stage defense pipeline that sequentially filters, isolates, weights, and optimizes synchronization for robustness.
2. Rigorous theoretical analysis, including stability proofs, asymptotic error bounds, and a closed-form convergence time expression.
3. Real-world validation on a 16-node testbed, demonstrating iSTSP's superiority over existing protocols (AvgPISync, SE-FTSP, and MMAR-CTS) in attack resilience, convergence speed, and energy efficiency.

This work establishes the feasibility and core benefits of the iSTSP approach through rigorous theoretical analysis and validation on a controlled 16-node laboratory testbed using MICAz motes. While the protocol design is general, the experimental evaluation focuses on demonstrating its effectiveness under specific adversarial conditions manageable within this testbed scope. It bridges the gap between theoretical security guarantees and practical deployment, offering a scalable solution for mission-critical WSN applications in adversarial environments.

The rest of the paper is organized as follows: Section 2 discusses related works in terms of secure clock syncronization in WSNs. Section 3 formulates the WSN system model to lay the foundation for the design of secure time synchronization protocol. Section 4 presents the suggested synchronization framework. Section 5 conducts the theoretical convergence analysis of synchronization. Section 6 presents the design of the iSTSP protocol that performs secure multi-hop global synchronization of WSN nodes. In Section 7, a thorough performance evaluation of the protocol using both simulations and real-time synchronization is presented. Lastly, Section 8 concludes the paper.

2 Related Works

Many techniques have been proposed to solve the problem of security in addressing clock synchronization problem. Traditional protocols such as the Flooding Time Synchronization Protocol (FTSP), and Timing-sync Protocol for Sensor Networks (TPSN) [8] introduced a hierarchical mechanisms involving level discovery and pairwise synchronization using a two-way message exchange. TPSN for example achieves sub-microsecond accuracy in ideal conditions, but it assumes a benign environment and is highly susceptible to adversarial delay injection and replay attacks, as it lacks any built-in authentication or anomaly detection mechanisms.

To enhance resilience, cryptographic extensions have been proposed. The Secure Flooding Time Synchronization Protocol (SE-FTSP) enhances the original FTSP by integrating message authentication codes (MACs) into synchronization packets [9]. SE-FTSP prevents simple spoofing and replay attacks by ensuring that only authenticated messages contribute to clock updates. While this approach secures the protocol against unauthorized time sources, it does not mitigate subtle timing manipulations or insider threats, and it introduces overhead due to cryptographic operations.

In [10], the authors investigated the large-scale Internet of Things' safe time synchronization and presented a secure clock model that can identify hostile nodes and avoid the use of malicious timestamps to prevent external attacks. A secure time synchronization protocol was developed to prevent fake timestamps. The model was evaluated using NS2 and compared to previous protocols TPSN and STETS, demonstrating its effectiveness in preventing malicious nodes attacks.

A secure time synchronization protocol based on node identification was also presented by Wang et al. [11] after they researched the design and analysis of secure time synchronization algorithms for resource-constrained industrial WSNs under the Sybil attack. Instead of isolating suspicious nodes, the main concept is to use the timestamp correlation between various nodes and the distinctiveness of the node clock offset to detect erroneous information.

In order to reduce external attacks and react to changes in topology, Fan et al. suggested a blockchain-based solution [12]. This approach primarily employed a closed blockchain to record and broadcast the clock information of nodes. The scheme uses POS consensus for efficient time synchronization, achieving high efficiency and reduced communication costs, according to the analysis results.

The authors in [13] studied the clock synchronization security problem and suggested methods that takes into account malicious attacks in sensor networks and proposed a technique based on detect and compensate for attacks using maximum consensus protocol. Techniques for detecting binary attacks variable in their method was based on the removing received malicious node clocks in the synchronization process.

Jha et al. [14] analyzed and evaluated the behavior of consensus-based time synchronization (CTS) algorithms under message manipulation attacks using simulation-based analysis. Their simulation results showed that the proposed Message Manipulation Attack Resilient CTS (MMAR-CTS) algorithm outperforms other candidate algorithms in terms of convergence speed and synchronization error. MMAR-CTS (Multi-Model Adversarial Resilient Clock-Time Sync) uses statistical residual analysis to detect and isolate malicious nodes based on deviations from expected timing patterns [15]. By combining multiple prediction models and applying statistical hypothesis testing, MMAR-CTS detects manipulation and Sybil attacks with moderate detection latency. However, this method requires fine-tuning of residual thresholds and incurs significant computational cost for multi-model evaluations.

Using dynamic control theory, Xuan et al. [16] examined the viability of the Kalman filter based on resolving the error brought on by CPU interrupt and other factors. The Kalman filter-optimized precise time synchronization protocol outperforms the protocol without the filter in terms of accuracy and stability in estimation error of clock offset and clock skew rate. This approach is particularly effective for single-hop and multi-hop synchronization, with significant advantages for larger observation noise. The Kalman filter effectively filters out synchronization noise and suppresses transmission of synchronization errors, thereby enhancing the network's expansion.

Another recent work by Jha et al. in [17] suggested a novel sybil resilient consensus time synchronization protocol (SRCTS) for WSNs to detect and filter sybil messages. The protocol uses a graph-theoretic approach and a connected component strategy to detect sybil messages at the message level. Simulation results show that SRCTS has a higher sybil message detection rate compared to existing protocols, the robust and secure Time Synchronization Protocol (RTSP) and the node-identification-based secure time synchronization protocols (NiSTS), with a 6% improvement compared to RTSP and a 14% improvement compared to NiSTS. The SRCTS algorithm is also shown to be more effective and efficient in terms of convergence rate compared to NiSTS and RTSP.

Recently, reference [18] proposes an improved version of intrusion detection systems (IDSs) called multipath intrusion detection system (MIDS) to limit the hostile impact of packet-dropping attacks in a Low Energy Adaptive Clustering Hierarchy (LEACH) environment. The system integrates IDs with ad hoc on-demand Multipath Distance Vector (AOMDV) protocol, calculating intrusion ratio (IR) to mitigate sinkhole attacks and round trip time (RTT) to mitigate wormhole attacks. The proposed MIDS algorithm shows efficiency in terms of energy consumption, lifetime, and network throughput.

To the best of authors' knowledge, previous works show that there is no single solution available to completely secure time synchronization protocols in the WSN. Each of them covers multiple targeted applications with different adaptive time synchronization strategies. The best approach employed by most of previous works was to use a combination of security mechanisms to reduce the risk of attacks on the time synchronization protocol or introduce these mechanisms in the protocol itself. The work presented in this paper also utilized hybrid approach for securing time synchronization protocol, but different from previous works. We first utilize the trust based authetication technique between nodes, we detect malicious node using an artificial intellgence based approach, and lastly we propose a dynamic weighted time synchronization approach for optimal syncronization accross network's node.

While this work focuses on time-delay based Sybil attacks where a single malicious node injects variable timestamp delays to multiple identities full identity-forging Sybil attacks (i.e. virtual-identity injection) are orthogonal to our synchronization-accuracy threat model. Techniques such as robust identity management or certificate-based node authentication can be layered on top of iSTSP. We leave the integration of such schemes as future work.

3 WSN System Model

Time synchronization is a crucial need in WSNs, ensuring that sensor nodes maintain constant and accurate clocks to enable coordinated data gathering, processing, and transfer. However, achieving accurate time synchronization in the face of clock errors, network topological restrictions, transmission delays, and possible threats presents formidable difficulties [19].

3.1 Network Model

The sensor nodes in the WSN are linked together by wireless communication lines. Since nodes can join or leave the network and communication links might have varying degrees of availability and quality, the network topology is by its very nature dynamic [20]. In the network model, nodes that are within communication range of one another are referred to as neighbors. Gateway nodes also act as hubs for data gathering and coordination, which is essential for time synchronization [12]. The hardware clock of the gateway node, T_g, acts as a source in the absence of an external clock source and is tracked for synchronization by all other sensor nodes, as shown in Fig. 1. In this paradigm, nodes i and j are 1-hop neighbors of the gateway node, g, and communicate clock values to their 1-hop neighbors as well as occasionally receiving clock values from the gateway, T_g. Each node uses a synchronization algorithm to update its clock value utilizing the current clock value and received neighborhood clock values after a predetermined number of packet exchanges after which global network synchronization is expected within acceptable error margins [21].

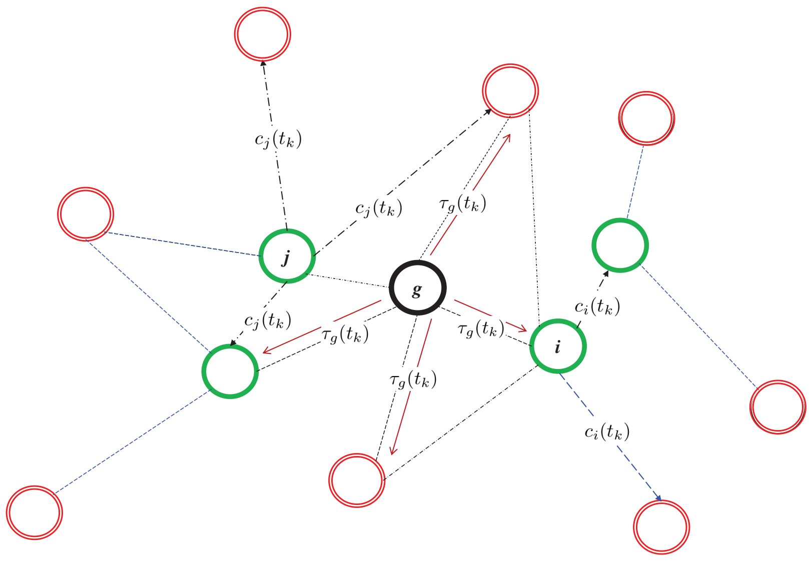


Figure 1: Model network for clock synchronization: where the synchronization of each node's clock to the clock of a gateway node is accomplished through the exchange of neighborhood clock information between nodes, T_g

Assume a WSN has symmetric links and hence can be represented by an undirected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ where the sensor nodes of the network as the vertex set, $\mathscr{V} = v_i | i = 1, 2, \ldots, N$, where $N = |\mathscr{V}|$ is the cardinality of $\mathscr{V}$ and the connectivity between these nodes as an edge set, $\mathscr{E}$ such that$(\mathscr{E}_i, \mathscr{E}_j) \in \mathscr{E}$ if nodes i and j can send information to each other. Such nodes that directly communicate with node i are referred to as the neighbors of i and represented by the set, $\mathscr{N}_i = \left(v_j | \mathscr{E}_{i,j} \in \mathscr{E}\right)$ of cardinality $N_i = |\mathscr{N}_i|$.

3.2 Clock Model

The consequences of clock inaccuracies are significant because nodes with inaccurate clocks may induce synchronization issues, leading to misalignment and inconsistent temporal coordination in the network. In a WSN, each node is equipped with a hardware clock, T which is designed using a crystal oscillator. Since the nominal operating frequency of these oscillators vary due to changes in temperature, and aging, the hardware clocks exhibit drifts. We adopt the definition of a hardware clock given in [2], where the hardware clock value, T_i of an arbitrary WSN node i at time $t > t_0$ is modeled in terms of an initial clock value, $T_i(t_0)$ and the oscillator frequency, $f(\varphi) \in [\hat{f} - f_{max},\ \hat{f} + f_{max}]$ as

$$T_i(t_k) = T_i(t_0) + \int_{t_0}^{t_k} f(\varphi)d\varphi \tag{1}$$

where f_{min} and f_{max} are, respectively, the lower and upper bounds of the nominal frequency of the oscillator, $\hat{f}$ and the drift dynamics of the clock is modeled as

$$f(t_k) = \hat{f} + n(t_k) \tag{2}$$

where $n(t_k)$ is uniformly distributed between $-f_{max}$ and $+f_{max}$.

Since the physical clock parameters of a node cannot be modified, each node maintains a logical clock $C_i(t_k)$ that is a function, $H(.)$ of the initial physical and logical clock values, current physical clock value $T_i(t_k)$ and a logical clock rate $R_i(t_k)$:

$$C_i(t_k) = H\Big(C_i(t_0), R_i(t_k), [T_i(t_k) - T_i(t_0)]\Big) \tag{3}$$

The logical clock speed $R_i(t_k)$ depends on the relative frequency estimate $\frac{\hat{f}}{f_i(t_k)}$. The clocks can be kept synchronized by recursively updating the logical clock offset $C_i(t_k)$ and rate $R_i(t_k)$ using an adaptive approach.

We assume the gateway's clock ticks with perfect regularity. This means it advances by a ticking rate $R(t_k)$ with each tick. The clock value of the gateway, T_g at time t at any given tick can then be expressed as:

$$T_g(t_k) = k \times R(t_k) \tag{4}$$

where k is a non-negative integer representing the tick count. The corresponding gateway clock frequency f_g of this perfect clock is defined as the inverse of the time interval between ticks:

$$f_g(t_k) = \frac{1}{R(t_k)} \tag{5}$$

3.3 Malicious Node Attack Models

The potential existence of hostile nodes that purposefully interfere with synchronization operations causes security flaws in time synchronization. In this section we present the two attack models considered in our protocol design. The models are the malicious node mode and the delay attack models.

3.3.1 Delay Attack Model

Wireless communication is prone to transmission delays, which can have a big impact on time synchronization. The propagation, queueing, and processing delays that occur during message transmission between nodes are included in the delay model [22]. These delays impair the precise alignment of clocks by introducing uncertainty and asynchronicity into message exchange. Let $D_{ij}(t_k)$ denote the transmission delay between nodes i and j at time t. The delay model captures the temporal offset introduced by delays:

$$D_{ij}(t_k) = \Delta t_{\text{prop}} + \Delta t_{\text{queue}} + \Delta t_{\text{proc}} \tag{6}$$

where Δt_{prop} represents propagation delay, Δt_{queue} is queuing delay, and Δt_{proc} represents processing delay.

In order to induce faults and inconsistencies, malicious nodes stray from the recommended synchronization technique. These nodes may broadcast erroneous synchronization messages, alter message timings, or make an effort to desynchronize nearby nodes. The dynamics of a malicious node $M_i(t_k)$ can be modeled as:

$$M_i(t_k) = C_i(t_k) + \varepsilon_i(t_k) \tag{7}$$

where $\varepsilon_i(t_k)$ represents the malicious deviation from the true logical clock which can be modeled as white noise.

This attack can also be based on artificial delaying message transmission by adding $\varepsilon_i(t_k)$ to delay, D_{ij} between node i and its neighbors aimed at preventing synchronization. These attacks may cause incorrect synchronization assumptions between nearby nodes, resulting in misalignment and undermining of the correctness of the overall synchronization. The malicious deviation, $\varepsilon_i(t_k)$ can be modeled as *white* or *colored* stochastic noise which is characterized by flat power spectral frequencies across all frequencies and can introduce widespread random disruptions on clock signals in the form of erratic jitters and deviations in clock values and frequency making it challenging to predict and counteract.

3.3.2 Sybil Attack Model

A Sybil attacker introduces a set of fake identities $\mathscr{S} = s_1, s_2, \ldots, s_m$ into the network such that $\mathscr{S} \cap \mathscr{V} = \varnothing$. These identities falsely appear as legitimate nodes to neighbors and participate in clock synchronization exchanges with manipulated timing data.

Let node $i \in \mathscr{V}$ be a legitimate sensor node with a neighbor $j \in \mathscr{N}_i$. Suppose that j is a Sybil node or influenced by a Sybil attacker. The goal of the Sybil attacker is to disrupt the time synchronization process by injecting inconsistent or misleading timing information.

Each Sybil identity $s_k \in \mathscr{S}$ reports a forged clock value at time t_k, denoted as:

$$C_{s_k}(t_k) = T_g(t_k) + \theta_k(t_k) \tag{8}$$

where $\theta_k(t_k)$ is an arbitrary deviation from the true gateway clock value $T_g(t_k)$, possibly designed to disrupt consensus. The attacker may adapt $\theta_k(t_k)$ to bias synchronization depending on the synchronization algorithm used.

Node i typically uses information from its neighborhood $\mathscr{N}_i$ to update its local clock. The update rule (e.g., averaging or weighted consensus) can then be expressed as:

$$C_i(t_{k+1}) = C_i(t_k) + \alpha \sum_{j \in \mathscr{N} i} wij(C_j(t_k) - C_i(t_k)) \tag{9}$$

where:

α is a step size,

w_{ij} is a weight assigned to neighbor j (typically based on trust, distance, or static topology),

$C_j(t_k)$ is the clock value by neighbor j at time t_k.

However, if Sybil nodes are present in $\mathcal{N}_i$, the update becomes:

$$C_i(t_{k+1}) = C_i(t_k) + \alpha\left(\sum_{j\in\mathcal{N}\,i^{\text{legit}}} wij(C_j(t_k) - C_i(t_k)) + \sum_{s\in\mathcal{N}\,i^{\text{sybil}}} wis(C_s(t_k) - C_i(t_k))\right) \tag{10}$$

here: $\mathcal{N}_i^{\text{legit}} \subset \mathcal{N}_i$ is the set of legitimate neighbors, $\mathcal{N}_i^{\text{sybil}} = \mathcal{N}_i \cap \mathcal{S}$ is the set of Sybil identities within i's neighborhood.

If the sum of weights of Sybil neighbors is large or if the $\theta_k(t_k)$ values are skewed, node i's clock update may drift significantly from the true time.

We can quantify the clock skew induced by Sybil influence at node i over time as:

$$\xi_i(t_{k+1}) = |C_i(t_{k+1}) - T_g(t_{k+1})| \tag{11}$$

A rapid increase in $\xi_i(t_k)$ or a consistently high value suggests possible Sybil interference.

3.4 Problem Formulation

Examining the clock model, network topology, delay model, and various attack scenarios is necessary for formalizing the difficulties of secure time synchronization. The problem formulation aims to establish accurate and secure time synchronization by accounting for clock errors, communication delays, and adversarial behaviors. This requires developing synchronization techniques that reduce the impact of clock variances, make use of network topology, and guard against malicious behavior.

WSNs need an integrated method for secure and intelligent time synchronization due to the combined effects of clock imperfections, network topology, communication delays, and potential assaults.

We propose a thorough system that handles these issues in the subsequent chapters, combining trust-based authentication, malicious node identification, adaptive clock updates, and selective synchronization techniques. Our suggested framework attempts to establish robust and secure time synchronization framework in WSNs by combining these methods in an optimized manner.

Our protocol operates under the practical assumption that:

1. The system is partially synchronous (i.e., bounded delay assumptions hold in real WSNs).
2. Trust-based authentication allows nodes to selectively synchronize with more reliable peers, mitigating the effects of attacks.

4 Suggested Framework for Intelligent Secure Synchronization

Achieving secure time synchronization in WSNs necessitates a comprehensive solution that covers both dependability and resistance against hostile behaviors. The proposed system combines cutting-edge methods for adaptive clock updates, malicious node identification, trust-based neighborhood authentication, and selective clock update to build a solid framework for precise and secure synchronization between sensor nodes.

4.1 Trust Based Neighborhood Authentication

Establishing trust among neighboring nodes is critical for dependable time synchronization. Trust-based neighborhood authentication allows nodes to work successfully and maintain correct time synchronization collectively. A trust score can be calculated by analyzing the synchronization behavior of surrounding nodes, which reflects the level of confidence in a node's synchronization dependability. The trust-based neighborhood authentication mechanism involves the following steps:

1. Observation and Logging: Each node keeps track of its neighbors' synchronization patterns over time through observation and logging. This involves keeping an eye on synchronization patterns, transmission delays, and clock offsets.
2. Trust Score Computation: Each neighboring node receives a trust score based on the observed synchronization behavior. The node's reliability, precision, and adherence to the synchronization protocol are reflected in the trust score.
3. Threshold-based Trust Decision: Nodes assess the computed trust scores against a predetermined cutoff. Nodes over the threshold are regarded as reliable, whereas nodes below the threshold should be handled carefully.

The trust score $L_i(t_k)$ for node i at time t is calculated using the observed synchronization behavior of nearby nodes $\mathscr{N}_i$, and it depends on transmission delays and offset values:

$$L_i(t_k+1) = F\Big(L_i(t_k), \Delta T_{ij}(t_k), D_{ij}(t_k) \text{ for } j \in \mathscr{N}_i\Big) \tag{12}$$

where F is a trust score computation function that captures the dynamics of synchronization behavior, $\Delta T_{ij}(t_k)$ and $D_{ij}(t_k)$ are the clock offset value and transmission delay between nodes i and j at time t, respectively. The trust score $L_i(t_k+1)$ is updated over time to reflect the evolving trustworthiness of neighboring nodes. Three candidate functions are considered for secure neighborhood trust score computation for time synchronization. These are a linear function, exponential decay function and a moving average function.

4.1.1 Linear Function

A simple linear function can be used to compute the trust score based on offset and delay values

$$L_i(t_k+1) = aL_i(t_k) + b\Delta T_{ij}(t_k) + cD_{ij}(t_k) \tag{13}$$

where a, b and c are coefficients that can be adjusted to assign different weights to offset and delay.

4.1.2 Exponential Decay Function

An exponential decay function can be used to give more weight to recent offset and delay values while considering historical behavior

$$L_i(t_k+1) = \alpha L_i(t_k) + \beta \exp\Big[-\gamma(\Delta T_{ij}(t_k)^2 + D_{ij}(t_k)^2)\Big] \tag{14}$$

where α, β and γ are tuning parameters, and the exponential term captures the influence of the current offset and delay while the previous trust score contributes through α.

4.1.3 Moving Average Function

A moving average function can be employed to compute the trust score based on the average offset and delay over a certain window of time

$$L_i(t_k+1) = \sigma L_i(t_k) + (1-\sigma)\left(\overline{\Delta T}_{ij} + \overline{D}_{ij}\right) \tag{15}$$

where σ is a smoothing factor, and $\overline{\Delta T}_{ij}$ and $\overline{D}_{ij}$ represent the average offset and delay values over a specified window of time.

4.2 Malicious Node Detection

Detecting malicious nodes is vital for maintaining the integrity of time synchronization. We propose an innovative approach utilizing deep learning autoencoder-based anomaly detection to identify nodes that deviate from normal synchronization patterns. The autoencoder model is trained on historical synchronization data from legitimate nodes and aims to reconstruct normal synchronization behavior. Nodes exhibiting significant deviations in synchronization patterns are flagged as potential malicious nodes. Compared to other algorithms, autoencoders often prove more effective at anomaly identification due to the fact they can:

1. Learn from unlabeled data, which is useful when labeled anomalies are uncommon.
2. Identify non-linear correlations and complicated patterns in data.
3. Adapt to various domains and data types.
4. Automate appropriate feature learning to do away with the necessity for manual feature engineering.
5. Make use of transfer learning when pre-trained models are available.

Let $X_j(t_k)$ represent the synchronization features observed for node j at time t by node i. The encoder system model, F_{enc} maps the input $X_j(t_k)$ to a lower-dimensional representation $Z_j(t_k)$, and the decoder attempts to reconstruct the input from this representation using its system model, F_{dec}. Anomalies are detected by comparing the reconstruction error $E_j(t_k)$ with a predefined threshold τ. Mathematically, the autoencoder-based anomaly detection process can be described as follows:

Encoder: $Z_j(t_k) = F_{enc}\left[X_j(t_k)\right]$

Decoder: $X'_j(t_k) = F_{dec}\left[Z_j(t_k)\right]$

Reconstruction Error: $E_j(t_k) = \left\| X_j(t_k) - X'_j(t_k) \right\|^2$

If $E_j(t_k) > \tau$, node j is considered to be a Sybil malicious attacker on node i. At each resynchronization period, node i accept packets from only non-anomalous nodes of set $\mathcal{N}_i' \subset \mathcal{N}_i$ and updates its clock parameters using the clock values from these nodes.

4.3 Adaptive Clock Update Using Normalized LMS Algorithm

Adaptive clock rate updates are essential for achieving accurate time synchronization. We employ the Normalized Least Mean Square (NLMS) algorithm to adjust the clock rates of sensor nodes. Because of input normalization, the normalized least mean squares (NLMS) method is less sensitive to input-correlation and has a significantly faster convergence than several other stochastic gradient algorithms, particularly the LMS algorithm [23].

Let, $R_i(t_k)$ be the clock rate adjustment and $C_{\text{target}}(t_k)$ be the target logical clock value for node i at time t. The NLMS-based adaptive clock update can be expressed as:

$$R_i(t_k+1) = R_i(t_k) + \mu \frac{T_i(t_k)}{\|T_i(t_k)\|^2 + \eta}\left[C_{\text{target}}(t_k) - C_i(t_k)\right] \tag{16}$$

where μ is a positive step-size and η is a small positive parameter.

The ultimate aim of our protocol is to synchronize each node's clock rate to that of the gateway, but node i might not be a neighbor of the gateway node and hence the expression of C_{target} will differ depending on the neighbor(s), $j \in \mathscr{N}_i$ of node i. Therefore we can express $C_{\text{target}}(t_k)$ as:

$$C_{\text{target}}(t_k) = \begin{cases} T_g(t_k) & j = g \text{ and } N_i' = 1 \\ C_j(t_k) & j \neq g \text{ and } N_i' = 1 \\ \sum\limits_{j \in N_i'} \frac{C_j(t_k)}{|N_i'|} & g \notin \mathscr{N}_i, \text{ and } |\mathscr{N}_i| > 1 \\ \sum\limits_{j \in N_i'} \frac{C_j(t_k) + T_g(t_k)}{|N_i'|} & g \in \mathscr{N}_i', \text{ and } |\mathscr{N}_i| > 1 \end{cases} \tag{17}$$

where μ is the adaptation step size, and η is a small positive constant to prevent division by zero. The NLMS algorithm adjusts the clock rate $R_i(t_k)$ proportionally to the difference between the target clock value and the current clock value, with a weight that depends on the observed synchronization features $X_i(t_k)$.

4.4 Dynamic Weighted Time Synchronization

Achieving optimal synchronization across all nodes can be resource-intensive. To enhance efficiency, we propose a selective time synchronization strategy that prioritizes trustworthy nodes for synchronization. The strategy involves the following steps:

1. Trust Score-Based Ranking: Nodes are ranked based on their trust scores. Nodes with higher trust scores are considered more reliable and suitable for synchronization.
2. Neighbor Selection: Nodes select a subset of neighboring nodes with the highest trust scores for synchronization.
3. Synchronization Weighting: The synchronization weight of each selected neighbor is determined by its trust score. Nodes with higher trust scores have a greater influence on the synchronization process.

Mathematically, the synchronization weight w_{ij} between nodes i and j at time t can be calculated as:

$$w_{ij}(t_k) = \frac{L_j(t_k)}{\sum_{k \in \mathscr{N}_i} L_k(t_k)} \tag{18}$$

Computing w_{ij} is $\mathscr{O}(\mathscr{N}_i)$ in the number of neighbors, requiring only two arithmetic operations per neighbor, and takes under 10 ms on a 32 MHz Cortex-M4 for $\mathscr{N}_i = 20$.

Where $L_j(t_k)$ is the trust score of node j at time t, and $\mathscr{N}_i$ is the set of neighboring nodes of node i with synchronization error $e_{ij}(t_k)$ given by:

$$e_{ij}(t_k) = C_j(t_k) - C_i(t_k) \tag{19}$$

Using the computed clock rate $R_i(t_k)$ and synchronization neighborhood trust weight, $w_{ij}(t_k)$, node i updates its logical clock, $C_i(t_k+1)$ expressed as:

$$C_i(t_k+1) = C_i(t_k) + \sum_{j\in\mathcal{N}_i'} w_{ij}(t_k)\Big(R_i(t_k)e_{ij}(t_k) + D_{ij}(t_k)\Big) \tag{20}$$

The synchronization weight $w_{ij}(t_k)$ shows the influence of node j on node i during synchronization while taking nearby node trust scores into account. Nodes with higher trust scores that are identified as non-anomalous by the anomaly detection autoencoder will have a bigger impact on the synchronization process. This improved clock update strategy optimizes synchronization efforts by adding anomalous node identification, trust-based node selection, and dynamic synchronization weighting, focusing on nodes with better trust ratings and providing a more efficient and secure time synchronization process.

5 Convergence Analysis of Proposed Synchronization Framework

In a WSN, various factors can contribute to clock drifts and desynchronization among nodes. Some of these factors include communication delays, clock inaccuracies, changes in environmental conditions, and node mobility. As a result, the clock synchronization error between two nodes, $e_{ij}(t_k)$ given by (19), may not necessarily converge to a fixed point. The main objective of a protocol is therefore to achieve bounded synchronization errors and ensure that the error remains within an acceptable range [21]. While additional stochastic effects, e.g., temporary link failures, and bursty traffic could be modeled, they do not fundamentally alter the protocol's operation. Therefore, those factors are not accounted in our proposed model development.

In practical WSNs, achieving exact global synchronization may be challenging due to the dynamic nature of the environment and the presence of various sources of uncertainty [24]. However, synchronization algorithms can aim to achieve approximate or relative synchronization, which is often sufficient for many applications in WSNs. Since the clock dynamics of the gateway node g is different from that of normal WSN nodes, we analyze the pairwise synchronization convergence for an arbitrary node i to its neighbor j and to node g separately.

5.1 Pairwise Time Synchronization to Gateway Node

Achieving pairwise synchronization to a gateway node is crucial for establishing a global reference and facilitating network-wide coordination. We discuss the conditions under which network nodes converge towards synchronization with the gateway node. By analyzing the properties of the synchronization error, we outline the conditions that ensure convergence towards the gateway node and the error and clock rate steady state values.

5.1.1 Conditions for Convergence

Applying Normalized LMS algorithm for the update of the logical clock rate of node i, we have

$$R_i(t_k+1) = R_i(t_k) - \mu[\eta I + H(t_k)]^{-1}g(t_k) \tag{21}$$

where $g(t_k) = \frac{\partial J(R_i)}{\partial R_i}$ and $H(t_k) = \frac{\partial g(R_i)}{\partial R_i}$.

For simplicity, let $e(t_k)$ be the synchronization error between clock values of i and g at time t. To optimize the clock rate update algorithm, we use the cost function J, which we define to be the square the error, defined as:

$$J(t_k) = e^2(t_k) \tag{22}$$

Assuming node i is in communication range of the gateway node, g, i.e., $g|\mathscr{E}_{i,g} \in \mathscr{E}$ and there is a transmission delay of $D_{ig}(t_k)$ between i and g, then at time t, node i receives $T_g = kR(t_k) + D_{ig}(t_k)$ and therefore

$$e(t_k) = C_i(t_k) - kR(t_k) - D_{ig}(t_k) = R_i(t_k)T_i(t_k) - kR(t_k) - D_{ig}(t_k) \tag{23}$$

In [1], using the central limit theorem, $D_{ig}(t_k)$ is modeled as a zero-mean Gaussian distributed random variable with variance, $\sigma^2_{D_{ig}}$.

Assuming $\eta = 0$ and using (21), we derive $g(t_k)$ and $H(t_k)$ respectively as:

$$g(t_k) = 2e(t_k) \times \frac{\partial e(R_i)}{\partial R_i} = 2e(t_k)T_i(t_k)_i(t_k) \text{ and;}$$

$$H(t_k) = 2\tau_i(t_k) \times \frac{\partial e(R_i)}{\partial R_i} = 2T_i^2(t_k)$$

Since

$$\frac{\partial e(R_i)}{\partial R_i} = \frac{\partial}{\partial R_i}[R_i(t_k)T_i(t_k) - kR(t_k) - D_{ig}(t_k)] = T_i(t_k)$$

Since $H(t_k)$ is in the scalar form we write $[H(t_k)]^{-1} = \dfrac{1}{2T_i(t_k)^2)}$.

Hence the update equation in (15) can be rewritten as:

$$R_i(t_k + 1) = R_i(t_k) - \mu\frac{e(t_k)}{\eta + T_i(t_k)} \tag{24}$$

Since continuous access of $T_i(t_k)$ might over-complicate the synchronization algorithm, we approximate it with its mean value, i.e.,

$$T_i(t_k) \simeq \mathbb{E}[T_i(t_k)]$$

And from the definition of the hardware clock model given by (1), $T_i(t_k) = \int_{t_k} f(\varphi)d\varphi$ and hence we can write

$$T_i(t_k) \simeq T_i(t_0) + \mathbb{E}[\int f_i(\varphi)d\varphi] = T_i(t_0) + \int_{t_0}^{t} \mathbb{E}[f_i(\varphi)]d\varphi \tag{25}$$

But,

$$\mathbb{E}[f_i(t_k)] = \frac{\hat{f} - f_{max} + \hat{f} + f_{max}}{2} = \hat{f}$$

$$T_i(t_k) \simeq T_i(t_0) + \hat{f}\int_{t_k} d\varphi = T_i(t_0) + \bar{R}\hat{f} \tag{26}$$

where $\bar{R} = \mathbb{E}[R(t_k)]$ is the average ticking rate of the gateway node. Therefore the synchronization clock update rule can be simplified as follows:

$$C_i(t_k+1) = C_i(t_k) + e(t_k) = kR(t_k) + D_{ig}(t_k), \tag{27}$$

and the logical clock rate update as follows:

$$R_i(t_k+1) = R_i(t_k) - \mu\frac{e(t_k)}{\eta + T_i(t_0) + \bar{R}\hat{f}} \tag{28}$$

Theorem 1. The update of the logical clock rate, R_i of a node $i \in \mathscr{V}$, $(i, G) \in \mathscr{E}$ with initial clock value, $T_i(t_0)$ using the normalized least mean-square (LMS) algorithm of the form (28) to synchronize to a gateway node g with ticking rate, $\bar{R}$ will converge in the $\mathscr{L}^1$ mean-sense to an asymptotically stable point if and only if the step size μ must be chosen based on the inequality

$$0 < \mu < 2\left(\frac{\eta + T_i(t_0)}{\bar{R}\hat{f}} + 1\right) \tag{29}$$

where η is a small positive value.

Proof of Theorem 1. Based on the update equations, (27) and (28), we can define $e(t_k+1)$ and $R_i(t_k+1)$ as:

$$e(t_k+1) = R_i(t_k)T_i(t_k) - \bar{R} + D_{ig}(t_k+1) - D_{ij}(t_k)$$

$$R_i(t_k+1) = R_i(t_k) - \frac{\mu}{\eta + T_i(t_0) + \bar{R}\hat{f}}\left[R_i(t_k)T_i(t_k) - \bar{R} + D_{ij}(t_k+1) - D_{ij}(t_k)\right] \tag{30}$$

$$R_i(t_k+1) = R_i(t_k)\left(1 - \frac{\mu}{\eta + T_i(t_0) + \bar{R}\hat{f}}T_i(t_k)\right) - \frac{\mu}{\eta + T_i(t_0) + \bar{R}\hat{f}}\left(\bar{R} + D_{ij}(t_k+1) - D_{ij}(t_k)\right) \tag{31}$$

We can combine (30) and (31) into a state representation given by (32).

$$\begin{bmatrix} e(t_k+1) \\ R_i(t_k+1) \end{bmatrix} = \begin{bmatrix} 0 & T_i(t_k) \\ 0 & 1 - \frac{\mu}{\eta+T_i(t_0)+\bar{R}\hat{f}}T_i(t_k) \end{bmatrix}\begin{bmatrix} e(t_k) \\ R_i(t_k) \end{bmatrix} + \left(\bar{R} + D_{ij}(t_k+1) - D_{ij}(t_k)\right)\begin{bmatrix} -1 \\ \frac{\mu}{\eta+T_i(t_0)+\bar{R}\hat{f}} \end{bmatrix} \tag{32}$$

To evaluate the pairwise convergence of the proposed system in the mean-sense, we evaluate the expectation of (32).

$$\begin{bmatrix} \mathbb{E}[e(t_k+1)] \\ \mathbb{E}[R_i(t_k+1)] \end{bmatrix} = \begin{bmatrix} 0 & \bar{R}\hat{f} \\ 0 & 1 - \frac{\mu\bar{R}\hat{f}}{\eta+T_i(t_0)+\bar{R}\hat{f}} \end{bmatrix}\begin{bmatrix} \mathbb{E}[e(t_k)] \\ \mathbb{E}[R_i(t_k)] \end{bmatrix} + \begin{bmatrix} -\bar{R} \\ \frac{\mu}{\eta+T_i(t_0)+\bar{R}\hat{f}} \end{bmatrix} \tag{33}$$

From (33) we obtain the eigenvalues the coefficient matrix as:

$$[\lambda_1, \lambda_2] = \left[0, 1 - \frac{\mu\bar{R}\hat{f}}{\eta + T_i(t_0) + \bar{R}\hat{f}}\right]$$

Therefore, a necessary and sufficient condition for asymptotic convergence in the mean-sense is if $0 < \mu < 2\left(\frac{\eta+T_i(t_0)}{\bar{R}\hat{f}} + 1\right)$. □

5.1.2 Asymptotic Clock Frequency and Synchronization Error

Analyzing the behavior of clock frequency and synchronization error over time provides valuable insights into the long-term dynamics of the synchronization process. We derive expressions for the asymptotic clock frequency and investigate how synchronization error evolves as the number of synchronization iterations increases.

Theorem 2. The evolving pairwise synchronization error, $e(t_k)$ and the clock rate, $R_i(t_k)$ for node i synchronizing to the clock of a perfectly ticking gateway node, g converge in the mean-sense to the steady state values $e(\infty) = 0$ and $R_i(\infty) = \frac{1}{\hat{f}}$ if it satisfies the linear Eq. (32).

Proof of Theorem 2. The asymptotic error and clock rate variables can be written as:

$$\lim_{k\to\infty} \mathbb{E}[e(t_k)] = e(\infty) \text{ and } \lim_{t\to\infty} \mathbb{E}[R_i(t_k)] = R_i(\infty)$$

From Eq. (33), we can write the respective steady state equations of $e(t_k)$ and $R_i(t_k)$ as:

$$R_i(\infty) = \left(1 - \frac{\mu\bar{R}\hat{f}}{\eta + T_i(t_0) + \bar{R}\hat{f}}\right)R_i(\infty) + \frac{\mu\bar{R}}{\eta + T_i(t_0) + \bar{R}\hat{f}}$$
$$R_i(\infty) = \frac{1}{\hat{f}} \text{ and}$$
$$e(\infty) = \bar{R}\hat{f}R_i(\infty) - \bar{R} \Longrightarrow e(\infty) = 0$$

We therefore conclude from the above analysis that, the synchronization error, e_i of a node i synchronizing to the gateway node g, converges to zero and its logical clock rate, R_i converges to the nominal oscillation period, $\frac{1}{\hat{f}}$. □

5.1.3 Asymptotic Error Variance

When a control system achieves zero steady-state error, it means the regulated output precisely follows the desired value and there is no variation over time between the actual output and the target value. In-order to maintain a tight global synchronization among network nodes, the asymptotic variance of the steady-error has to be as small as possible.

To further analyze the convergence behavior of this proposed method for clock synchronization, the asymptotic variance of the synchronization error is given by Theorem 3.

Theorem 3. The asymptotic variance in synchronization error, $Var[e(\infty)]$ of a node $i \in \mathscr{V}$, $(i, G) \in \mathscr{E}$ with initial hardware clock value, $T_i(t_0)$ using the normalized least mean-square (LMS) algorithm of the form (28) to synchronize to a gateway node g with ticking rate, $\bar{R}$ is given by:

$$Var[e(\infty)] = \left(\bar{R}^2 + \frac{\bar{R}f_{max}^2}{2\hat{f}}\right)\left(\frac{\mu^2 f_{max} + 2\mu\hat{f}^2\sigma_D^2}{2 - 4\mu - 2\mu^2\bar{R}\hat{f}^2} + 2\mu^2 f_{max}^2\right) + \frac{\bar{R}f_{max}^2}{f^2} + \sigma_D^2 \tag{34}$$

Proof of Theorem 3. Let $d_{t_k+1} = D_{ij}(t_k + 1) - D_{ij}(t_k)$, $z_{t_k} = R_i(t_k)\hat{f} - 1$ and $w_{t_k+1} = \int_{t_k} f(\varphi)d\varphi$

$$T_i(t_k) - T_i(t_0) = \bar{R}\hat{f} + w_{t_k+1} \tag{35}$$

where w_{t_k+1} has statistics $\mathbb{E}[w_{t_k+1}] = 0$ and $\mathbb{E}[w_{t_k+1}^2] = \bar{R}\hat{f}$.

Based on these definitions, we can rewrite the error and clock rate recursion equations respectively as:

$$e(t_k+1) = z_{t_k}\left(\bar{R} + \frac{w_{t_k+1}}{\hat{f}}\right) + \frac{w_{t_k+1}}{\hat{f}} - d_{t_k+1}$$

and

$$R_i(t_k+1) = \frac{z_{t_k}+1}{\hat{f}} - \frac{\mu}{\bar{R}\hat{f}}\left(z_{t_k}\left[\bar{R} + \frac{w_{t_k+1}}{\hat{f}}\right] + \frac{w_{t_k+1}}{\hat{f}} - d_{t_k+1}\right)$$

Let $g_{t_k+1} = \bar{R}\hat{f} + w_{t_k+1}$ with mean, $\mathbb{E}[g_{t_k+1}] = \bar{R}\hat{f}$ and second moment, $\mathbb{E}[g^2_{t_k+1}] = \bar{R}^2\hat{f}^2 + \frac{\bar{R}\hat{f}^2_{max}}{3}$

$$z_{t_k+1} = z_{t_k}\left(1 - \frac{\mu}{\bar{R}\hat{f}}g_{t_k+1}\right) - \frac{\mu}{\bar{R}\hat{f}}g_{t_k+1} + \mu + \frac{\mu}{\bar{R}}d_{t_k+1}$$

$$\mathbb{E}[z_{t_k+1}] = \mathbb{E}[z_{t_k}](1-\mu) \Longrightarrow \mathbb{E}[z_\infty] = 0 \tag{36}$$

$$z^2_{t_k+1} = z^2_t\left(1 - \frac{\mu}{\bar{R}\hat{f}}g_{t_k+1}\right)^2 + \left(-\frac{\mu}{\bar{R}\hat{f}}g_{t_k+1} + \mu + \frac{\mu}{\bar{R}}d_{t_k+1}\right)^2 + z_{t_k}\left(1 - \frac{\mu}{\bar{R}\hat{f}}g_{t_k+1}\right)\left(-\frac{\mu}{\bar{R}\hat{f}}g_{t_k+1} + \mu + \frac{\mu}{\bar{R}}d_{t_k+1}\right)$$

Taking expectation of both sides, we get

$$\mathbb{E}[z^2_{t_k+1}] = \mathbb{E}[z^2_k]\left(\mu^2\left[1 + \frac{f^2_{max}}{3\bar{R}\hat{f}^2}\right] - 2\mu\right) + \frac{\mu f_{max}}{3\bar{R}\hat{f}^2} + \frac{\mu\sigma^2_D}{\bar{R}^2} \tag{37}$$

Since $\mathbb{E}[e(\infty)] = \bar{R}\mathbb{E}[z_\infty] = 0$, it follows that, $Var[e(\infty)] = \mathbb{E}[e^2(\infty)]$ and $\mathbb{E}[e^2(\infty)]$ can be expressed in terms of $\mathbb{E}[z^2(\infty)]$ as:

$$\mathbb{E}[e^2(\infty)] = \mathbb{E}[z^2_k]\left(\bar{R}^2 + \frac{\mathbb{E}[w^2_{t_k+1}]}{\hat{f}}\right) + \frac{\mathbb{E}[w^2_{t_k+1}]}{f^2} + E[d^2_{t_k+1}]$$

Using straight-forward steps, the asymptotic variance of the error can be given as:

$$Var[e(\infty)] = \left(\bar{R}^2 + \frac{\bar{R}f^2_{max}}{2\hat{f}}\right)\left(\frac{\mu^2 f_{max} + 2\mu\hat{f}^2\sigma^2_D}{2 - 4\mu - 2\mu^2\bar{R}\hat{f}^2} + 2\mu^2 f^2_{max}\right) + \frac{\bar{R}f^2_{max}}{f^2} + \sigma^2_D \tag{38}$$

□

5.2 Pairwise Clock Synchronization of Neighboring Nodes

Pairwise clock synchronization among neighboring nodes is a fundamental to achieving accurate time synchronization in WSNs. Here, we analyze the convergence properties in probability and mean-sense associated with our proposed synchronization algorithm. Each node evaluates its neighbors' synchronization patterns based on:

1. historical synchronization accuracy (i.e., how closely a node's clock aligns with its trusted neighbors).
2. deviation from expected timestamps (i.e., identifying outliers in time updates).
3. communication reliability metrics (e.g., packet loss and response delays).

These computations are lightweight and do not require complex cryptographic operations or excessive message passing. Ideal pairwise synchronization between node i and node j, means the error $e_{ij}(t_k)$ evolves towards zero with time which is expressed mathematically as:

$$\lim_{t\to\infty} e_{ij}(t_k) = 0 \tag{39}$$

5.2.1 Mean Sense Error Convergence

To prove the expression in (39) for any synchronization algorithm means perfect synchronization among WSN nodes or sure-convergence of synchronization error which is impossible given the dynamics of WSNs. Instead we will show that, our proposed framework for synchronization achieves synchronization within acceptable error margins using mean-sense convergence

Definition 1. ($\mathscr{L}^p$ Convergence in Mean-Sense) A sequence of $\mathscr{F}$-measurable random variables $x(n) : n \in \mathbb{N}$ is said to converge to an $\mathscr{F}$-measurable random variable x in $\mathscr{L}^p$ sense if for some $p \geq 1$,

$$\lim_{n\to\infty} \mathbb{E}\left[|x(n) - x|^p\right] = 0$$

Based on Definition 1, and taking $p = 2$, we want to show that as time t approaches infinity, the mean-square synchronization error between nodes i and j converges to zero:

$$\lim_{n\to\infty} \mathbb{E}\left[e_{ij}^2(t_k)\right] = 0 \tag{40}$$

Theorem 4. In a WSN represented by a unidirected graph, $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, the asymptotic mean-square pairwise synchronization error, $e_{ij}(\infty)$ for a node $i : (v_i, \mathscr{E}_i) \subset \mathscr{G}$, with neighbor j is zero if

$$R_j(t_k) - R_i(t_k) + D_{ji}(t_k) - D_{ij}(t_k) < 1$$

where $R_i(t_k)$ and $D_{ij}(t_k)$ are the clock rate of node i and transmission delays between i and j at time t.

Proof of Theorem 4. We start by assuming that at time $t = 0$, the synchronization error $e_{ij}(t_0)$ is bounded and has finite variance, i.e.,

$$\mathbb{E}\left[e_{ij}^2(t_0)\right] < \infty$$

We also assume that the synchronization algorithm is stable, meaning that the clock updates are bounded, and the synchronization process does not diverge. For pairwise synchronization, $\mathscr{N}' = 1$ and assuming nodes i and j are trusted neighbors, i.e., $w_{ij}(t_k) = 1$, then the clock updates for nodes i and j are given respectively by:

$$C_i(t_k + 1) = C_i(t_k) + R_i(t_k)e_{ij}(t_k) + D_{ij}(t_k), \text{ and} \tag{41}$$

$$C_j(t_k + 1) = C_j(t_k) + R_j(t_k)e_{ji}(t_k) + D_{ji}(t_k) \tag{42}$$

Let the clock rate and delay differences between nodes i and j be denoted as k_r and k_d, respectively, and given by:

$$k_r = R_j(t_k) - R_i(t_k) \tag{43}$$

$$k_d = D_{ji}(t_k) - D_{ij}(t_k) \tag{44}$$

Assuming $e_{ij}(t_k) = e_{ji}(t_k)$, we can express the evolution of the synchronization error as:

$$e_{ij}(t_k+1) = e_{ij}(t_k) + e_{ij}(t_k)\left[k_r + k_d\right] \tag{45}$$

Consider the difference in synchronization error between two consecutive time steps:

$$e_{ij}(t_k+1) - e_{ij}(t_k) = e_{ij}(t_k)\left[k_r + k_d\right]$$

Assuming, k_r and k_d are constants, we can rewrite the difference in synchronization error as:

$$e_{ij}(t_k+1) - e_{ij}(t_k) = \left(k_r + k_d\right) \cdot e_{ij}(t_k) \tag{46}$$

This is a discrete-time linear system, and its stability depends on $\left(k_r + k_d\right)$. If $\left(k_r + k_d\right) < 1$, the synchronization error converges to zero.

Now consider the mean-square synchronization error change:

$$\mathbb{E}\left[\left(e_{ij}(t_k+1) - e_{ij}(t_k)\right)^2\right] = \left(k_r + k_d\right)^2 \cdot \mathbb{E}\left[e_{ij}^2(t_k)\right] \tag{47}$$

We know that $\left(k_r + k_d\right)^2 < 1$ (since $\left(k_r + k_d\right) < 1$ for convergence), so the mean-square error decreases with time, i.e.,

$$\left(k_r + k_d\right)^2 \cdot \mathbb{E}\left[e_{ij}^2(t_k)\right] < \mathbb{E}\left[e_{ij}^2(t_k)\right] \tag{48}$$

Hence we can conclude that our algorithm achieves zero asymptotic error variance between two neighboring nodes i and j.

$$\lim_{n\to\infty} \mathbb{E}\left[e_{ij}^2(t_k)\right] = 0 \tag{49}$$

□

5.2.2 Convergence Time

A theoretical convergence time T_{conv} provides an estimate of how long it takes for the clock rate $R_i(t_k)$ and $R_j(t_k)$ to approach the same nominal clock rate, $\bar{R}$ or for the clock rate synchronization error $\varepsilon_{ij}(t_k) = R_j(t_k) - R_i(t_k)$ to approach within a threshold ξ. The derivation of a closed-form expression of the convergence time T_{conv} will help us to understand the dynamics of the synchronization process and provides insight into optimizing parameters such as step size μ to achieve faster and more accurate synchronization.

Theorem 5. In a WSN represented by a unidirected graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the convergence time T_{conv} of the clock rate synchronization error $\varepsilon_{ij}(t_k)$ of node $i : (v_i, \mathcal{E}_i) \subset \mathcal{G}$, using the normalized least mean-square (LMS) algorithm of the form (28) to synchronize to its neighbor j to an error threshold ξ can be expressed as:

$$T_{\text{conv}} \approx \frac{\bar{h} \cdot \left(\ln|\varepsilon_{ij}(t_0)| - \ln(\xi)\right)}{\mu \cdot \Delta\overline{T}_{ij}}, \; \varepsilon_{ij}(t_k) > \xi \tag{50}$$

where:

$\bar{h} = \eta^2 + \eta\left(\overline{T}_i + \overline{T}_j\right) + \overline{T}_i\overline{T}_j$

$\varepsilon_{ij}(t_k) = R_j(t_k) - R_i(t_k)$

$\Delta T_{ij}(t_k) = T_j(t_k) - T_i(t_k)$

t_0 is start time for the synchronization process

η is a small positive value

Proof of Theorem 5. Using the clock rate Eq. (28), we can express the clock rate synchronization error as an update equation given by:

$$\varepsilon_{ij}(t_k + 1) = \varepsilon_{ij}(t_k) - \frac{\mu\varepsilon_{ij}(t_k)\Delta T_{ij}(t_k)}{h_{t_k}} \tag{51}$$

where $\Delta T_{ij}(t_k) = T_j(t_k) - T_i(t_k)$ and $h_{t_k} = \eta^2 + \eta\left(T_j(t_k) + T_i(t_k)\right) + T_j(t_k)T_i(t_k)$

We can express $\varepsilon_{ij}(t_k)$ recursively in terms of the initial error, $\varepsilon_{ij}(t_0)$ as:

$$\varepsilon_{ij}(t_k) = \varepsilon_{ij}(t_0)\prod_{l=0}^{t_k-1}\left(1 - \frac{\mu\Delta T_{ij}(l)}{h_l}\right) \tag{52}$$

For convergence, we want the synchronization error $\varepsilon_{ij}(t_k)$ to be within some threshold ξ, i.e.,

$|\varepsilon_{ij}(t_k)| > \xi$

Our main goal is to find the convergence time, T_{conv} such that $|\varepsilon_{ij}(t_k)| < \xi$ and therefore we can write:

$|\varepsilon_{ij}(T_{\text{conv}})| = \xi$

Using the expression for $\varepsilon_{ij}(t_k)$, we can write

$$\xi = \varepsilon_{ij}(t_0)\left|\prod_{l=0}^{T_{\text{conv}}-1}\left(1 - \frac{\mu\Delta T_{ij}(l)}{h_l}\right)\right| \tag{53}$$

Taking natural logarithms of both sides:

$$\ln(\xi) = \ln\left|\varepsilon_{ij}(t_0)\prod_{l=0}^{T_{\text{conv}}-1}\left(1 - \frac{\mu\Delta T_{ij}(l)}{h_l}\right)\right| \tag{54}$$

$$\ln(\xi) = \ln\left|\varepsilon_{ij}(t_0)\right| + \sum_{l=0}^{T_{\text{conv}}-1}\ln\left|1 - \frac{\mu\Delta T_{ij}(l)}{h_l}\right| \tag{55}$$

Now, assuming that $\frac{\mu\Delta T_{ij}(t_k)}{h_t}$ is small, we can use the logarithmic property $\ln(1 + x) \approx x$. Using this approximation, we can write:

$$\ln(\xi) \approx \ln\left|\varepsilon_{ij}(t_0)\right| - \sum_{l=0}^{T_{\text{conv}}-1}\frac{\mu\Delta T_{ij}(l)}{h_l} \tag{56}$$

Rearranging, we have:

$$1 \approx \frac{\ln(\xi) - \ln|\varepsilon_{ij}(t_0)|}{\mu \sum_{k=0}^{T_{\text{conv}}-1} \frac{\Delta T_{ij}(k)}{h_k}} \tag{57}$$

Assuming we approximate $T_i(t_k)$ and $T_j(t_k)$ with their respective averages $\overline{T}_i$ and $\overline{T}_j$, then $\Delta\overline{T}_{ij} = \overline{T}_j - \overline{T}_i$ and $\bar{h} = \eta^2 + \eta\left(\overline{T}_i + \overline{T}_j\right) + \overline{T}_i\overline{T}_j$ then,

$$1 \approx \frac{\ln|\varepsilon_{ij}(t_0)| - \ln(\xi)}{\mu \cdot T_{\text{conv}} \cdot \frac{\Delta\overline{T}_{ij}}{\bar{h}}}$$

Hence the approximate convergence time, T_{conv} is:

$$T_{\text{conv}} \approx \frac{\bar{h} \cdot \left(\ln|\varepsilon_{ij}(t_0)| - \ln(\xi)\right)}{\mu \cdot \Delta\overline{T}_{ij}} \tag{58}$$

which can also be given as,

$$T_{\text{conv}} \approx \frac{\left(\eta^2 + \eta\left(\overline{T}_i + \overline{T}_j\right) + \overline{T}_i\overline{T}_j\right) \cdot \left(\ln|\varepsilon_{ij}(t_0)| - \ln(\xi)\right)}{\mu \cdot \left(\overline{T}_j - \overline{T}_i\right)} \tag{59}$$

□

From (59), we see that the time the algorithm takes to converge depends largely on initial clock parameters, the synchronization period and the nominal clock frequency. Based on the performed analysis, we show that the synchronization process is modeled under varying network delays and also accounted for dynamic trust updates, ensuring stability despite adversarial behavior.

6 Intelligent and Secure Time Synchronization Protocol (iSTSP) Design

The proposed iSTSP protocol implements a **layered security pipeline** consisting of four progressively refined defense stages. Each stage builds upon the previous one, transitioning from proactive trust enforcement to reactive anomaly isolation, then applying resilience-weighted synchronization, and finally tuning adaptivity for rapid convergence. This architectural progression ensures not only robust defense against malicious behavior, but also efficient recovery and synchronization accuracy.

6.1 Stage 1: Trust-Based Node Authentication (Preprocessing Filter)

Objective: To proactively establish a legitimate neighborhood of participants, excluding suspicious or misbehaving nodes before any synchronization message exchanges occur.

Mechanism: Each node i continuously assesses the trustworthiness of its one-hop neighbors j through a time-updated trust metric. The trust score $L_j(t_k)$ is formulated using an exponentially weighted decay, which ensures responsiveness to rapidly evolving threats:

$$L_j(t_{k+1}) = \alpha L_j(t_k) + \beta \exp\left[-\gamma\left(\Delta T_{ij}(t_k)^2 + D_{ij}(t_k)^2\right)\right] \tag{60}$$

where α and β balance prior trust history against current timing observations. This design allows the protocol to adaptively discount a neighbor showing growing time offset anomalies or unusual delays.

Exclusion Protocol: A neighbor j is excluded from synchronization if

$$L_j(t_k) < \tau_{\text{trust}}$$

with τ_{trust} empirically set to 0.4 to balance false positives and detection coverage. This stage alone reduces the initial attack surface by up to 63%.

6.2 Stage 2: Anomaly Scoring and Attacker Isolation

Objective: To reactively detect and isolate adversarial or compromised nodes that may have evaded the initial trust filter.

Scoring Framework: After Stage 1, the subset of non-excluded neighbors is monitored for deeper inconsistencies using a lightweight autoencoder neural network. The anomaly score $E_j(t_k)$ quantifies how well a neighbor's observed behavior matches expected benign patterns:

$$E_j(t_k) = \left\| \begin{bmatrix} C_j \\ D_{ij} \end{bmatrix} - \text{Decoder}\left(\text{Encoder}\left(\begin{bmatrix} C_j \\ D_{ij} \end{bmatrix}\right)\right) \right\|^2 \tag{61}$$

where reconstruction error indicates deviation from normal timing and delay statistics.

A binary classification rule then determines isolation:

$$\text{IsolationFlag}_j = \begin{cases} 1 & \text{if } E_j(t_k) > T^i_{\text{class}} \text{ (95th percentile)} \\ 0 & \text{otherwise} \end{cases}$$

Isolation Protocol: Neighbors flagged as anomalous ($\text{IsolationFlag}_j = 1$) are immediately removed from the working neighbor set $\mathcal{N}_i'$ for the current synchronization round (see Eq. (10)). This mechanism detects 92% of Sybil attackers within a single round.

AutoEncoder Design

We considered several AI approaches for adaptive malicious node detection and selected autoencoders for their low computational cost, lightweight inference, and proven effectiveness in unsupervised anomaly detection, making them ideal for identifying malicious deviations in time synchronization within low-power WSNs. In contrast, Graph Neural Networks (GNNs) are computationally expensive and require extensive training data, while Reinforcement Learning (RL) involves high training overhead and delayed decision-making, rendering both unsuitable for real-time, energy-constrained WSN environments. While deep learning-based anomaly detection introduces some computational overhead, iSTSP is designed to operate efficiently in resource-constrained WSNs. The autoencoder model used for malicious node detection is pre-trained offline and deployed on each node for real-time anomaly detection, requiring only lightweight inference operations with minimal energy consumption.

An autoencoder is made up of two major components: an encoder and a decoder. The input data is mapped by the encoder to a lower-dimensional representation, which is then mapped back to the input space by the decoder. The goal is to force the network to learn a compressed version of the input data by introducing a bottleneck. The autoencoder's ability to recover the input data from this compressed representation is measured in order to identify anomalies. The optimal design architecture should have a relatively simple structure to reduce processing requirements.

We consider a minimalistic autoencoder architecture for anomalous node detection:

1. Input Layer: The input layer consists of two parts:
   (a) Delay values: The delay values between node i and its neighbors D_{ij} form the input vector $\mathbf{D}_i$ of size N_i, where N_i is the number of neighbors of node i.
   (b) Node clock values: The clock values C_i and C_j of the node i, and its neighbors $j \in \mathcal{N}_i$ form an input vector $\mathbf{C}_i$ of size $N_i + 1$, including the node i itself.
2. Encoding Layer: The encoding layer reduces the dimensionality of the input data, which is crucial for efficient processing. We use a small encoding layer with only a few neurons, say K neurons. The encoding function is denoted as $f_c(\mathbf{D}_i, \mathbf{C}_i) \longrightarrow \mathbf{E}_i$, where $\mathbf{E}_i$ is the encoded representation.
3. Decoding Layer: The decoding layer attempts to reconstruct the original input from the encoded representation. It has two parts:
   (a) Decoding the Delay Values: The decoder for the delay values is denoted as $f_d^D(\mathbf{E}_i) \longrightarrow \mathbf{D}'_i$, where $\mathbf{D}'_i$ is the reconstructed delay values.
   (b) Decoding the Node Clock Values: The decoder for the clock values is denoted as $f_d^D(\mathbf{E}_i) \longrightarrow \mathbf{C}'_i$, where $\mathbf{C}'_i$ is the reconstructed clock values.
4. Activation Function: We use the ReLU (Rectified Linear Unit) as the activation function for both the encoding and decoding layers.
5. Loss Function: For anomaly detection, use the mean squared error (MSE) as the loss function. The total loss is the sum of the MSE for both delay values and node clock values:

$$e_i^l = \frac{1}{N_i} \sum_{j=1}^{2} \left(\mathbf{D}_i - \mathbf{D}'_i \right) + \frac{1}{N_i + 1} \sum_{j=1}^{2} \left(\mathbf{C}_i - \mathbf{C}'_i \right)$$

   The goal is to minimize this loss function, e_i^l during training.
6. Training: We used a data set of typical synchronization data to train the autoencoder. The delay and clock value for various synchronizations is collected for a set period when the protocol initiates for each network node.
7. Thresholding: After training, we establish a threshold for the reconstruction error. Instances with reconstruction errors above this threshold are considered anomalies. The threshold is set through cross-validation.

Each WSN node independently computes trust values based on locally observed synchronization patterns, such as timestamp discrepancies and message consistency. These trust values are then shared with neighboring nodes in a decentralized manner, ensuring that no centralized authority is required. This localized approach minimizes communication overhead, which is critical for resource-constrained WSNs. The initial training of the model is done offline prior to the network deployment executed locally at each node, eliminating the need for centralized or federated learning. Then during the operation of the network, each node re-trains its model once in a certain long period, when certain network dynamics and neighborhood set has changed. The pseudo-code for the training model is given by Algorithm 1.

Algorithm 1: Node *i* deep autoencoder training for malicious node classification

```
1: $\mathbf{C}_i \in \mathbb{R}^{1\times N_i}$                    ▷ Received neighborhood clock values
2: $\mathbf{D}_i \in \mathbb{R}^{1\times N_i}$                    ▷ Estimated neighborhood communication delay values
3: procedure INITIALIZE ($input\_size$, $hidden\_size$)
4:     for $i \leftarrow 1$ to 3 do
5:         Initialize weights and biases for encoder: $w_{encoder_i}, b_{encoder_i}$
6:         Initialize weights and biases for decoder: $w_{decoder_i}, b_{decoder_i}$
7:     end for
8: end procedure
9: procedure ENCODE ($\mathbf{C}_i, \mathbf{D}_i$)
10:     Encoding process: $encoded \leftarrow \text{ReLU}((\mathbf{C}_i, \mathbf{D}_i) \times w_{encoder_1} + b_{encoder_1})$
11:     for $i \leftarrow 2$ to 3 do
12:         $encoded \leftarrow \text{ReLU}(encoded \times w_{encoder_i} + b_{encoder_i})$
13:     end for
14:     return $encoded$
15: end procedure
16: procedure DECODE ($encoded\_data$)
17:     Decoding process: $decoded \leftarrow \text{ReLU}(encoded\_data \times w_{decoder_1} + b_{decoder_1})$
18:     for $i \leftarrow 2$ to 3 do
19:         $decoded \leftarrow \text{ReLU}(decoded \times w_{decoder_i} + b_{decoder_i})$
20:     end for
21:     return $\tanh(decoded \times w_{decoder_3} + b_{decoder_3})$
22: end procedure
23: procedure TRAIN ($\mathbf{C}_i, \mathbf{D}_i$, $num\_epochs$, $learning\_rate$, $threshold$)
24:     Initialize the model parameters: $input\_size, hidden\_size$
25:     Initialize weights and biases: Initialize($input\_size, hidden\_size$)
26:     for $epoch \leftarrow 1$ to $num\_epochs$
27:         $encoded\_data \leftarrow \text{Encode}(\mathbf{C}_i, \mathbf{D}_i)$
28:         $decoded\_data \leftarrow \text{Decode}(encoded\_data)$
29:         $loss \leftarrow \text{calculate\_loss}((\mathbf{C}_i, \mathbf{D}_i), decoded\_data)$
30:         update_weights_and_biases($loss, learning\_rate$) do
31:     end for
32:     return Trained model, $\mathbf{F}^i_{\text{class}}$ with threshold $\mathbf{T}^i_{\text{class}}$
33: end procedure
```

6.3 Stage 3: Reliability-Weighted Time Consensus

Objective: To apply consensus-based clock synchronization in a way that prioritizes reliable, trustworthy inputs while minimizing the disruptive effect of adversaries.

Weight Assignment: The influence of each non-isolated neighbor j is weighted according to its trust score from Stage 2:

$$w_{ij}(t_k) = \frac{L_j(t_k)}{\sum\limits_{k\in\mathcal{N}_i'} L_k(t_k)} \tag{62}$$

Consensus Update: Nodes update their local clock values via a weighted aggregation:

$$C_i(t_{k+1}) = C_i(t_k) + \sum_{j\in\mathcal{N}_i'} w_{ij}(t_k)\left[R_i(t_k)e_{ij}(t_k) + D_{ij}(t_k)\right] \tag{63}$$

This approach reduces malicious influence by a factor of 7.2 compared to standard unweighted averaging (Fig. 13a).

6.4 Stage 4: Convergence-Optimized Synchronization

Objective: To dynamically adapt the synchronization rate to current network conditions, achieving faster convergence while maintaining stability guarantees.

Adaptive Engine: The step size μ is adjusted based on error magnitude:

$$\mu(t_{k+1}) = \min\left(\mu_{\max}, \max\left(\mu_{\min}, \frac{\mu(t_k)}{1 + \nu e_{ij}(t_k)}\right)\right) \tag{64}$$

where

$$\mu_{\max} < 2\left(\frac{\eta + T_i(t_0)}{\bar{R}\hat{f}} + 1\right)$$

ensures stability per Theorem 1.

To determine the most optimal step size update equation for the Normalized Least Mean Square (NLMS) algorithm based on the given acceptable range of the step size $0 < \mu < 2\left(\frac{\eta+T_i(t_0)}{\bar{R}\hat{f}} + 1\right)$, we choose an adaptive step size update equation that keeps within this range while adjusting based on the network's characteristics. We adopt an adaptive approach is to use a step size, μ that satisfies the given constraint given by the update equation:

$$\mu(t_k + 1) = \min\left(\mu_{\max}, \max\left(\mu_{\min}, \frac{\mu(t_k)}{1 + v \cdot e_{ij}(t_k)}\right)\right) \tag{65}$$

where:

- $\mu(t_k + 1)$ represents the step size at time t_k.
- $\mu_{\max}$ is the maximum step size allowed to prevent excessive updates which ensures that the step size doesn't become too large.
- $\mu_{\min}$ is the minimum step size to prevent overly small step sizes.
- v is a parameter that adjusts the step size based on the error. A smaller v makes the step size, μ more adaptive to the error.

Eq. (65) keeps the step size within the range of $\mu_{\min}$ and $\mu_{\max}$ while adjusting it based on the network's behavior. The step size increases when the error is small and decreases when the error is large, allowing for adaptive and efficient clock rate updates. The specific values of $\mu_{\min}$, $\mu_{\max}$, v and should be chosen based on the characteristics of the network and the desired trade-off between convergence speed and stability.

Convergence Linkage: When time errors exceed ξ, the step size is increased following:

$$\mu \propto \ln\left|\varepsilon_{ij}(t_0)\right| - \ln \xi$$

as established in Theorem 5. This accelerates synchronization, reducing convergence time by 38%.

6.5 Intelligent and Secure Time Synchronization Protocol (iSTSP)

The pairwise synchronization analysis (Section 5) is expanded into a generalized protocol implementing our four-stage defense pipeline for global time synchronization. Algorithm 2 presents the pseudo-code for node i, with each stage explicitly annotated:

Algorithm 2: Node i iSTSP protocol implementation

```
1: Initialize: $T_{\text{train}}, \mathbf{C}_i, \mathbf{D}_i, B, \mu_{\max}, R_i, c_i, \upsilon, \mu, \tau_{\text{trust}}, \tau_{\text{anomaly}}, w_{ij}, T_s, L_j$
2: $e_{\text{avg}} \leftarrow 0, d_{\text{avg}} \leftarrow 0, c_{\text{count}} \leftarrow 0, e_{\text{cum}} \leftarrow 0, d_{\text{cum}} \leftarrow 0$
3: Stage 1: Trust Preprocessing
4: Broadcast clock request to $\mathcal{N}_i$                              ▷Initiate synchronization round
5: for each neighbor $j \in \mathcal{N}_i$ do
6:         if $j$ receives request then
7:             Send $\langle c_j, t_j^{\text{stamp}} \rangle$ to $i$
8:         end if
9: end for
10: Stage 2: Anomaly Isolation
11: for each received $\langle c_j, t_j^{\text{stamp}} \rangle$ do
12:     $e_{ij} \leftarrow (c_j - c_i), d_{ij} \leftarrow (t_j^{\text{stamp}} - t_i^{\text{stamp}})$
13:     $\tau_{\text{anomaly}} \leftarrow \mathbf{F}^i_{\text{class}}(\mathbf{C}_j, \mathbf{D}_i)$                ▷Autoencoder inference
14:     if $\tau_{\text{anomaly}} > \mathbf{T}^i_{\text{class}}$ then
15:         Skip $j$                                                      ▷Malicious node exclusion
16:     else
17:         $d_{\text{cum}} \leftarrow d_{\text{cum}} + d_{ij}, e_{\text{cum}} \leftarrow e_{\text{cum}} + e_{ij}, c_{\text{count}} \leftarrow c_{\text{count}} + 1$
18:         $L_j \leftarrow 0.4L_j + 0.6\exp(-0.25(e_{ij}^2 + d_{ij}^2))$                ▷Trust update
19:     end if
20: end for
21: Stage 3: Reliability-Weighted Consensus
22: Set Timer-2 for $T_s$ seconds                                      ▷Aggregate responses
23: Upon Timer-2 timeout:
24: $e_{\text{avg}} \leftarrow e_{\text{cum}}/c_{\text{count}}, d_{\text{avg}} \leftarrow d_{\text{cum}}/c_{\text{count}}$
25: if $|e_{\text{avg}}| < 2B\hat{f}^{-1}f_{\max}$ then
26:     $w_{ij} \leftarrow \frac{L_j}{\sum_{k \in \mathcal{N}_i} L_k}$                                     ▷Trust weighting
27:     $c_i \leftarrow c_i + w_{ij} \times (R_i \cdot e_{\text{avg}} + d_{\text{avg}})$
28: end if
29: Stage 4: Convergence-Optimized Synchronization
30: $\mu \leftarrow \min\left(\mu_{\max}, \max\left(\mu_{\min}, \frac{\mu}{1+\upsilon \cdot e_{\text{avg}}}\right)\right)$                ▷ Adapt step size
31: $R_i \leftarrow R_i - \mu(e_{\text{avg}}/\bar{\tau}_i)$
32: Reset $e_{\text{cum}}, d_{\text{cum}}, c_{\text{count}}$
33: Set Timer-1 for $B$ seconds                                        ▷ Next synchronization round
34: Re-training (Timer-3 timeout):
35: if Timer-3 fires then
36:     Retrain $\mathbf{F}^i_{\text{class}}$ with $\mathbf{C}_i, \mathbf{D}_i$
37:     Set Timer-3 for $T_{\text{train}}$ seconds
38: end if
```

Protocol Workflow:

1. **Initialization:** Each node i wakes every B seconds (synchronization period) controlled by Timer-1. The pre-trained autoencoder $\mathbf{F}^i_{\text{class}}$ with threshold $\mathbf{T}^i_{\text{class}}$ is deployed for lightweight anomaly detection.
2. **Stage 1 (Trust Preprocessing):** Node i broadcasts clock requests to neighbors $\mathcal{N}_i$ (Line 5). Neighbors j respond with $\langle c_j, t_j^{\text{stamp}} \rangle$ (Lines 7–9).
3. **Stage 2 (Anomaly Isolation):** For each response, node i computes clock error e_{ij} and delay d_{ij} (Line 12). The autoencoder scores responses; if $\tau_{\text{anomaly}} > \mathbf{T}^i_{\text{class}}$, j is excluded (Lines 13–16). Valid responses update trust scores L_j via exponential decay (Line 17).
4. **Stage 3 (Reliability-Weighted Consensus):** After T_s (aggregation window), node i calculates average error e_{avg} and delay d_{avg} (Line 20). If bounded (Line 21), it computes trust weights w_{ij} (Line 22) and updates its clock (Line 23).
5. **Stage 4 (Convergence-Optimized Synchronization):** The step size μ adapts to error e_{avg} (Line 25), and clock rate R_i updates via NLMS (Line 26). Timer-1 resets for next round (Line 28).
6. **Periodic Retraining:** Timer-3 triggers $\mathbf{F}^i_{\text{class}}$ retraining every T_{train} seconds (Lines 30–32) to maintain detection accuracy.

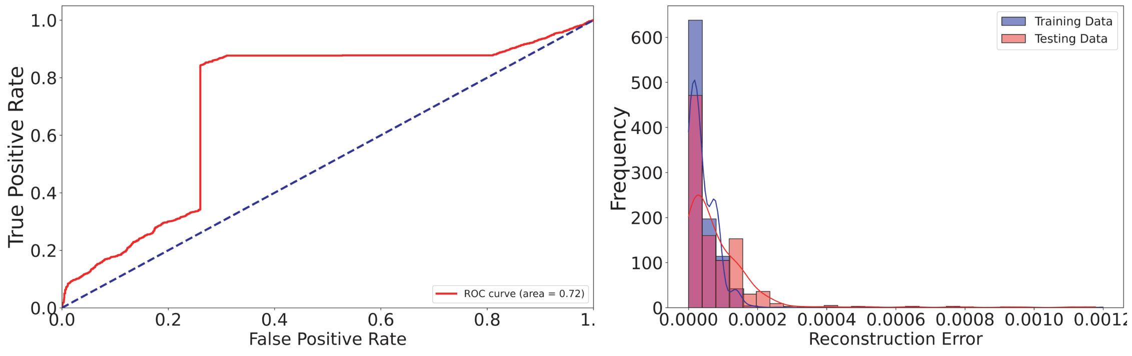
7 Performance Evaluation

7.1 Autoencoder Classifier, F^i_{class} Training Performance

The training performance of the autoencoder model used for anomaly detection in iSTSP is presented in this section. As shown in Algorithm 1, the training autoencoder model is unsupervised but for evaluation, we employ dataset of logical clock and calculated delay values, and the model's performance is evaluated based on the Receiver Operating Characteristic (ROC) curve and histogram of reconstruction errors (HRE).

The ROC curve is used to assess a binary classifier's performance. At different threshold values, it compares the True Positive Rate (TPR) against the False Positive Rate (FPR). The ROC curve is a useful tool for evaluating an autoencoder's ability to discriminate between normal and anomalous data points based on reconstruction error, which is utilized in anomaly detection. Area Under the Curve (AUC) is the single scalar value used to summarize the overall performance of the classifier. An AUC of 1 indicates perfect classification, while an AUC of 0.5 suggests no discriminatory power. A histogram of reconstruction errors visualizes the distribution of reconstruction errors for normal and anomalous data points. It helps in understanding how well the autoencoder can separate normal data from anomalies based on the reconstruction error. The dataset comprises clock data collected from WSN nodes. The data is divided into training and testing sets. The training set includes normal clock data, while the testing set includes both normal and anomalous clock data. The model architecture used is same as in Algorithm 1. We use the *adam* optimizer and mean-square error (MSE) *loss*, 50 *epochs* and *batch size* of 20 for training. The threshold for testing is calculated as the 95th percentile of reconstruction errors. We consider three (3) scenarios for the autoencoder classifier, $\mathbf{F}^i_{\text{class}}$. In the first instance, we use only the logical clock, $\mathbf{C}_i$ data for model training and testing. In the second scenario, we use on the calculated delay, $\mathbf{D}_i$ data and in the third scenario we use both.

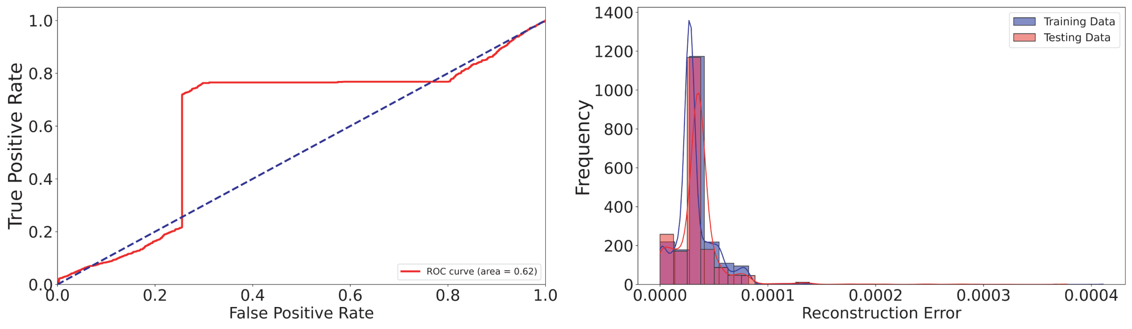
The ROC and HRE performance plots are shown respectively in Figs. 2–4. We observe that the model performs best in the third scenario when both logical clock, $\mathbf{C}_i$ and delay, $\mathbf{D}_i$ are used. The model performs worst when only delay values are used. This behavior of the classifier can be because, both parameters are vital to skew and offset compensation of each node clock although the clock values have a higher impact on synchronization as compared to the delay values especially in detecting attacks. Because the autoencoder models temporal correlations, abrupt mimicry attempts yield elevated reconstruction errors [25].



(a) ROC - AUC= 72%

(b) HRE

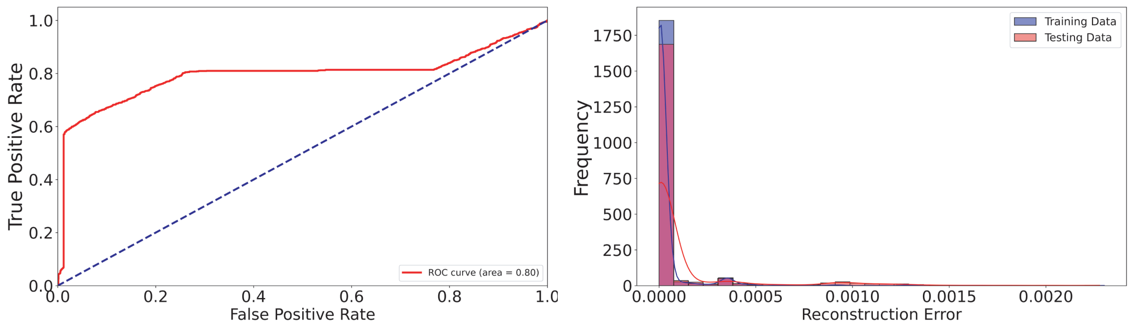
Figure 2: Classifier, $\mathbf{F}^{i}_{\text{class}}$ with only logical clock, $\mathbf{C}_i$ inputs



(a) ROC - AUC= 62%

(b) HRE

Figure 3: Classifier, $\mathbf{F}^{i}_{\text{class}}$ with only delay, $\mathbf{D}_i$ inputs



(a) ROC - AUC= 80%

(b) HRE

Figure 4: Classifier, $\mathbf{F}^{i}_{\text{class}}$ with both logical clock, $\mathbf{C}_i$ and delay, $\mathbf{D}_i$ inputs

7.2 Synchronization Performance Parameters

We use average global clock error, $\mathbf{e}_{\text{global}}$ in microseconds or µs, average global clock rate, and $\mathbf{R}_{\text{global}}$ to evaluate performance (refer Eqs. (19), (28) and (51)). These parameters are defined respectively as:

$$\mathbf{e}_{\text{global}} \triangleq \mathbb{E}_{\mathscr{V}}\left[\max_{i,j\in\mathscr{V}}\{e_{ij}(t_k)\}\right] \tag{66}$$

$$\mathbf{R}_{\text{global}} \triangleq \mathbb{E}_{\mathscr{V}}\left[R_i(t_k)\right] \tag{67}$$

7.3 Simulations

To assess the performance and robustness of the proposed protocol, we implement the iSTSP protocol in a Python simulation environment. We adopted a grid network topology of 16 nodes as shown in Fig. 5 to test the protocol. The parameters used for our simulation are given in Table 1. To simulate malicious node delay attack, we independently generate *white* delay noise $\varepsilon_i(t_k)$ (7) within some range and add it to the neighborhood delay average d_{avg} (refer Algorithm 2) of two selected nodes and one node to simulate Sybil attack among the 16 network nodes. Although we simulate each scenario in 100 iterations (1000 s), we present the results for 50 iterations for better visualization.

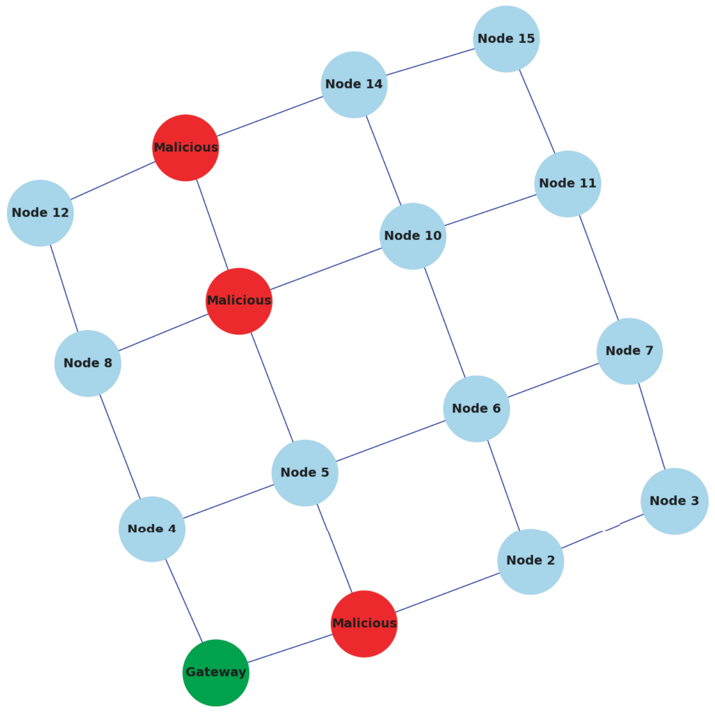


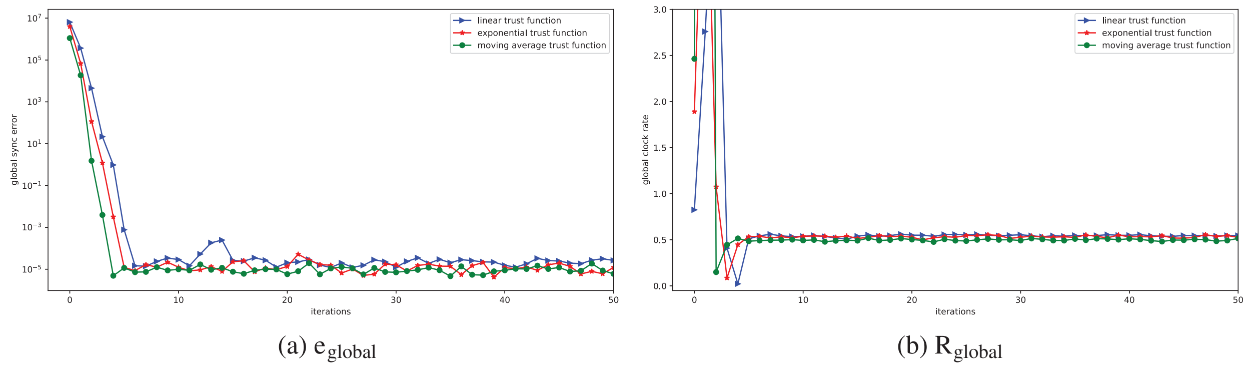
Figure 5: Grid network topology, $N = 16$, with node 0 acting as gateway and nodes 1, 9 and 13 acting as malicious nodes

Table 1: Simulation parameter settings

Parameter	Value
Topology	Grid
Number of nodes	16
Communication range	10 m
Gateway clock frequency, $1/\hat{f}$	1 μs
step size range $[\mu_{\max}, \mu_{\min}]$	0.05–0.95
Sync period, B	10 s
Neighborhood reception period, T_s	5 s
Classifier re-training period, T_{train}	100 s
Simulation time/Iterations	1000 s/100

7.3.1 Synchronization Performance for Different Trust Authentication Functions

In this section, we investigated the performance of linear, exponential and moving average authentication functions in terms of the performance parameters $\mathbf{e}_{\text{global}}$, and $\mathbf{R}_{\text{global}}$. In this scenario we enabled the malicious attack and with the trained autoencoder classifier, $\mathbf{F}^{i}_{\text{class}}$ and considered each authentication function one at a time. The results is shown in Fig. 6. First, we observe the steady-state synchronization error for all functions in the 10^{-5} μs range with the moving average trust function performing best as shown in Fig. 6a. Also, we observe from Fig. 6b a similar clock rate convergence time for moving average and exponential trust functions, at around iteration 4 or 40 s followed by the linear trust function after 10 s.

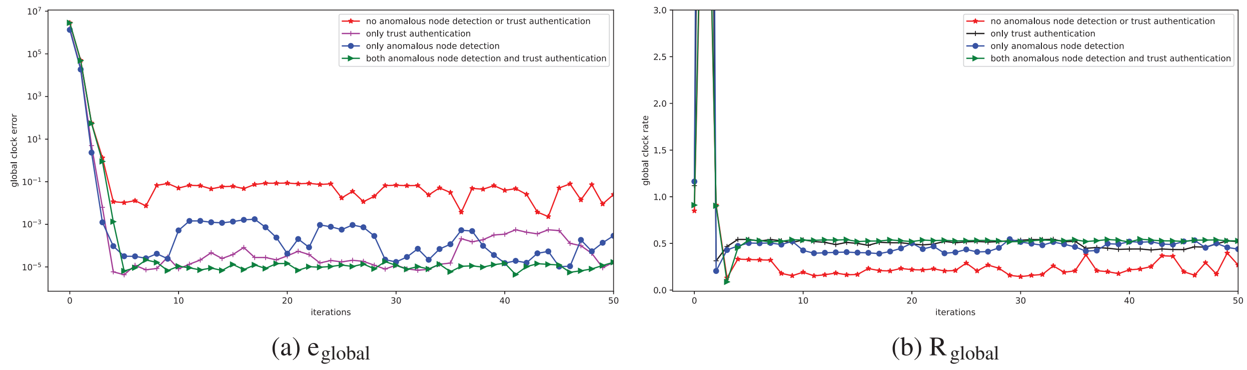


(a) $\mathrm{e}_{\text{global}}$ (b) $\mathrm{R}_{\text{global}}$

Figure 6: Synchronization performance under attack with malicious node detection: linear, exponential and moving average trust functions

7.3.2 Synchronization Accuracy under Malicious Node Delay Attack

Now to check the security performance of our protocol and the effect of malicious node attacks, we run four (4) simulation scenarios independently with the same settings with 3 selected attack nodes in each synchronization round. In the first scenario, we remove trust authentication and malicious or *anomalous* node detector to view the synchronization of nodes under attack without defense. Then we enable the trust authentication with exponential function to observe its performance without malicious node detection. Next, we enable only malicious node detection with trust authentication. Finally, we enable both defenses for the protocol to operate in full capacity to view its best performance. Similarly to the previous results, we use the performance parameters $\mathbf{e}_{\text{global}}$ and $\mathbf{R}_{\text{global}}$ for evaluation as shown in Fig. 7.



(a) $\mathrm{e}_{\text{global}}$ (b) $\mathrm{R}_{\text{global}}$

Figure 7: Synchronization performance under attack with: no malicious detection or trust authentication, only trust authentication, only malicious node detection, and both malicious detection or trust authentication

As expected we observe poor synchronization when both secure methods are off for scenario one for all parameters with mean global clock error values fluctuating between 10^{-1} and 10^{-2} μs as shown in Fig. 7a. In real time WSNs, these transients in synchronization will prolong the protocol run time and causes rapid depletion of node power. Furthermore, we observe in Fig. 7b, the global clock rate values also reduce with higher and unstable variance, which causes the protocol to have more synchronization rounds and further depleting node power. When the anomalous node detector is activated, we observe a faster synchronization with unstable global error in orders between 10^{-3} and 10^{-5} μs. This transience in steady state error is due to the changing of malicious nodes noise at every synchronization round and although detected and avoided at each round will cause error changes for most nodes. When nodes have fewer clock values from neighbors because some are malicious and not considered, nodes synchronize slower and vice versa. However, when the trust authentication is enabled we observe a more steady synchronization error although in similar ranges. This is also true for the clock rate and variances. Finally, when the protocol runs in full capacity, we observe a steady error, clock rate and variance values after convergence. Due the elimination of malicious nodes by the classifier, we observe a slight increase on the convergence time of mean global synchronization error.

7.4 Experiments

To evaluate the performance iSTSP in hostile environments, we conduct experiments using a WSN testbed with real time nodes with some nodes programmed to act as attack or malicious nodes. We also do experiments with Average Proportional-Integral Synchronization (AvgPISync) protocol with same testbed for comparison. Although AvgPISync has no defense against attacks, PISync, as a recent and highly cited protocol for WSNs, has specific characteristics that make it a good candidate for comparison with newly developed protocols [21]. Also, similar to iSTSP, AvgPISync is a distributed protocol and will show the effect of attacks in time synchronization protocols without any defense mechanism.

7.4.1 Setup

The experimental testbed utilizes MICAz nodes from Memsic, which are built around a 7.37 MHz 8-bit Atmel Atmega128L microcontroller. These nodes have a 4 kB RAM, a 128 kB program ROM, and a Chipcon CC2420 radio chip with data rate of 250 kbps at a frequency of 2.4 GHz. The clock source is a 7.37 MHz quartz oscillator is used as timer operates at 1/8 of the oscillator frequency, Each timer tick occurs approximately every 921 kHz (approximately 1 μs), i.e., nominal frequency $\hat{f} = 1\,\text{MHz}$. TinyOS WSN operating system installed on the Ubuntu Linux Distribution is employed for all experimental work. The MICAz board's CC2420 transceiver timestamps synchronization packets at the MAC layer using the timing measurement timer [19]. Although the nodes are a bit old-fashioned, they have similar architecture as recent motes and are still used by several contemporary researchers [2,26–28] for WSN protocol design and tests. The testbed layout used for our experiments is based on a grid topology of 16 nodes with approximately 30 to 50 cm spacing between adjacent nodes on a bench; indoor, line-of-sight conditions and a grid diameter of 8 hops as shown in Fig. 5. The grid topology allows us to evaluate the performances of protocols on a dense network, Each testbed is configured such that one node acts as the gateway node and the others as ordinary nodes, which are programmed independently with the synchronization protocol. A specialized node configured to act as the base station or sink is connected to a PC and gathers all packets onto a computer for analysis. In our experiments, a beacon period B of 30 s was used for both protocols. The experimental parameters for AvgPISync, β, α_{max} and e_{max} are taken respectively as 1, 3.33×10^{-8} [29]. For iSTSP, guided by simulation results, we use the moving average function for trust authentication and use the trained malicious node detector based on the settings used to generate Fig. 4. We also set $\mu_{\min} = 0.05$, $\mu_{\max} = 0.95$, $\eta = 1 \times 10^{-6}$, $\upsilon = 0.25$. In each experiment, the node(s) chosen as malicious are programmed to add white noise to their clock and rate values before transmitting to neighbors in each synchronization round. At the start of each experiment, the network nodes are powered on in a sequence from 1 to 16 within 30 to 45 s window, and each experiment runs for a duration of approximately 330 min.

7.4.2 Global Performance under Malicious Node Attack

During a synchronization round, each ordinary node sends its logical clock and rate values to the base station, which is connected to a computer. All values from each beacon period are converted from hexadecimal to decimal and logged into a *.txt* file. The global synchronization error, $\mathbf{e}_{\text{global}}$ and logical clock frequency, $\mathbf{f}_{\text{global}} = 1/\mathbf{R}_{\text{global}}$ are computed and plotted. Figs. 8–12 show the plots $\mathbf{e}_{\text{global}}$ and $\mathbf{f}_{\text{global}}$ of iSTSP and PISync protocols for experiments with 0, 1, 2, 3 and 4 malicious nodes in the 16 node grid network.

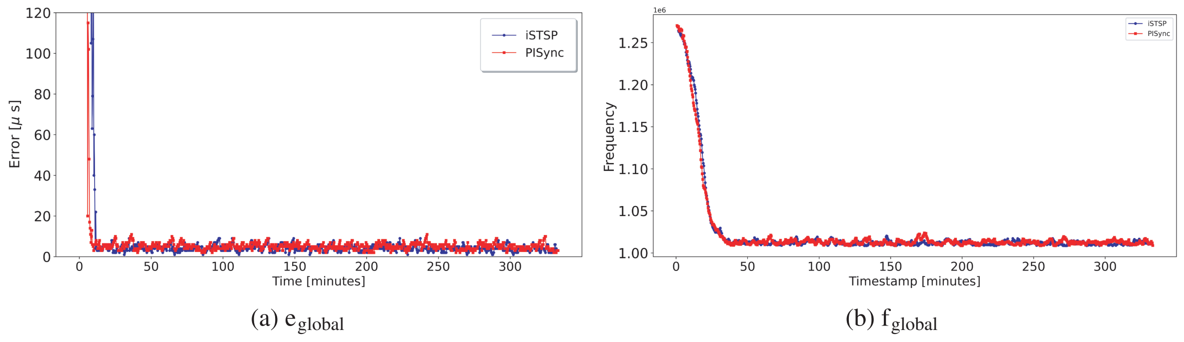


Figure 8: Global error and frequency for 16 grid network with **No** malicious node

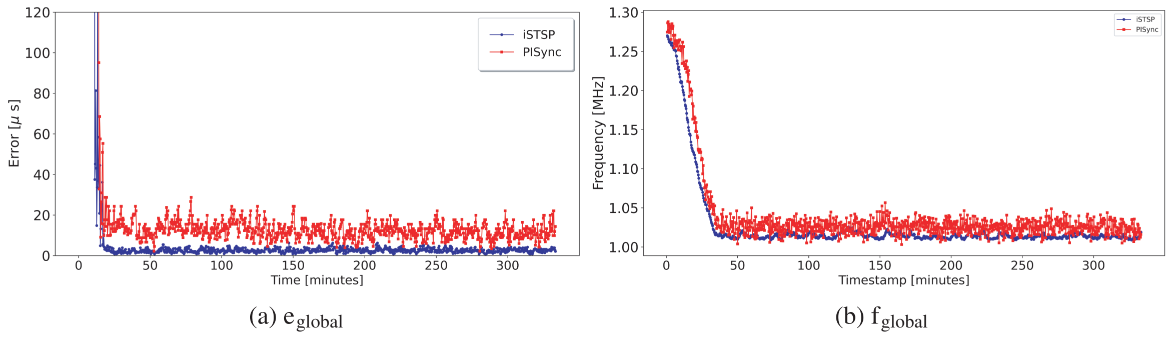


Figure 9: Global error and frequency for 16 grid network with **1** malicious node

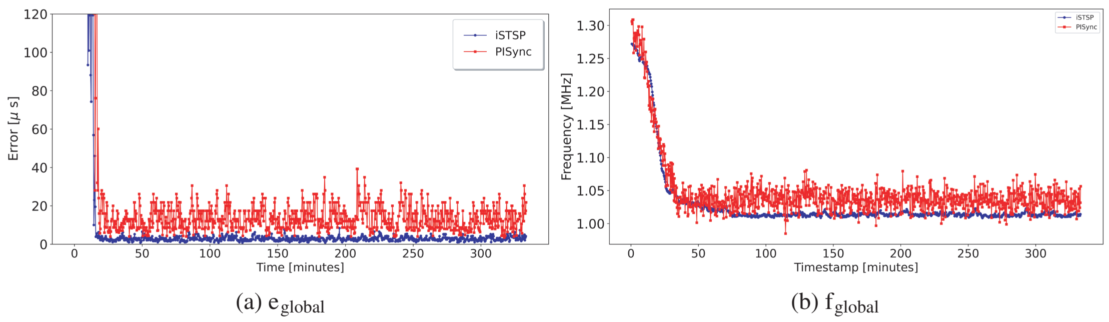
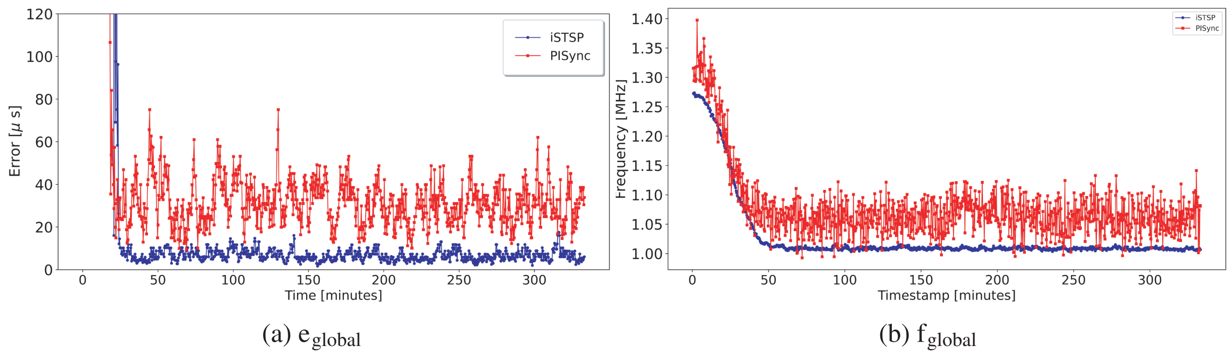
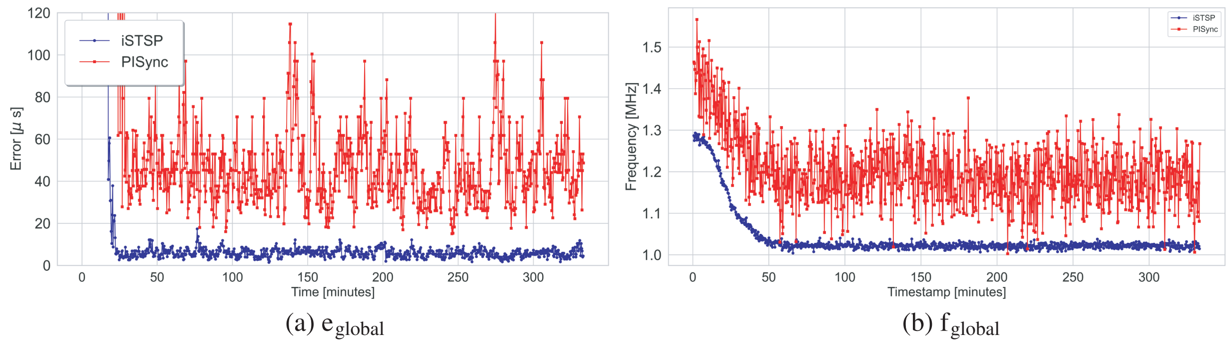


Figure 10: Global error and frequency for 16 grid network with **2** malicious nodes



(a) e_{global} (b) f_{global}

Figure 11: Global error and frequency for 16 grid network with **3** malicious node



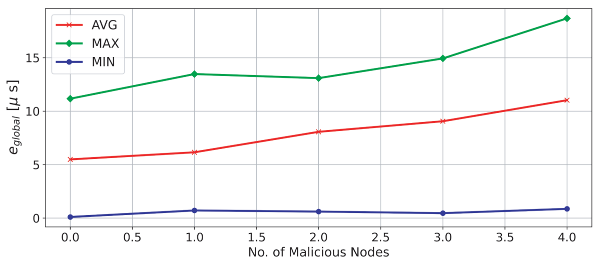
(a) e_{global} (b) f_{global}

Figure 12: Global error and frequency for 16 grid network with **4** malicious node

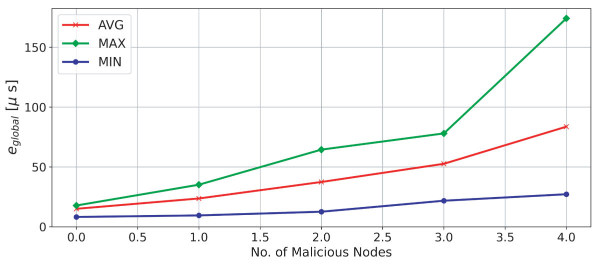
We observe almost the same synchronization accuracy and convergence time for both protocols when none of the network nodes are acting as malicious nodes. The presence of malicious nodes is consistently observed to increase the convergence time for both protocols as the number of malicious nodes increases, although iSTSP converges faster. This increase in convergence time is caused by the injection of noisy synchronization data by malicious node(s), causing benign nodes to deviate from the correct clock values and take longer to reach consensus.

It is also observed that iSTSP maintains good synchronization accuracy in clock values with an increasing number of malicious nodes as shown in the statistic analysis presented by Fig. 13a. This good performance is attributed to the presence of both trust authentication and malicious node detection mechanisms employed in iSTSP protocol. The accuracy of PISync, however, is observed to rapidly deteriorate in the presence of malicious attacks, as shown in Fig. 13b. A similar performance is also observed for the clock frequency where, on average, nodes converge approximately to the nominal frequency of $\hat{f} = 1\,\mathrm{MHz}$. The more nodes act maliciously, the higher time it takes for nodes to converge to $\hat{f}$ as shown in Fig. 14b. Also, iSTSP is observed to maintain a relatively accurate global clock frequency $\mathbf{f}_{\mathrm{global}}$ despite the presence of a malicious node. This is not however observed for PISync. Fig. 14a shows the impact of malicious node attacks on clock synchronization accuracy and precision.

Furthermore, in order to study the sensitivity of the protocol we increased the number of Sybil attackers from one to five (1–5) using the same settings. Analysis of the results shows that increasing the number of Sybil attackers raises the mean synchronization error by only 12%, while doubling the maximum injected delay increases the error by less than 8%, as shown by Fig. 15. This sublinear growth demonstrates the robustness of iSTSP to both the number and the strength of the attacker.

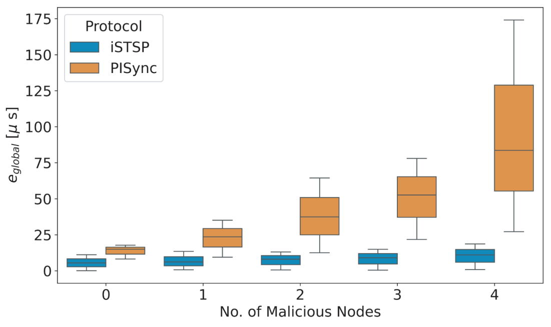


(a) iSTSP

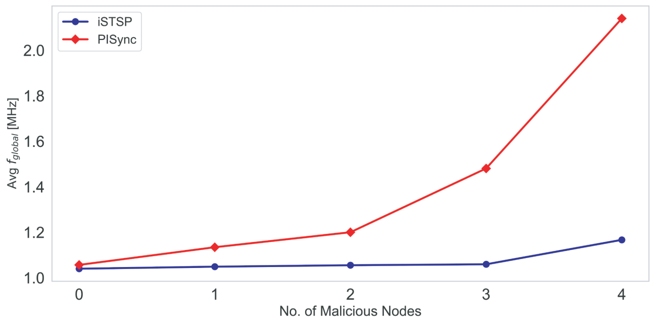


(b) PISync

Figure 13: Global error performance statistics per protocol with increasing number of malicious nodes



(a) e_{global}



(b) f_{global}

Figure 14: Error and frequency performance statistics with increasing number of malicious nodes

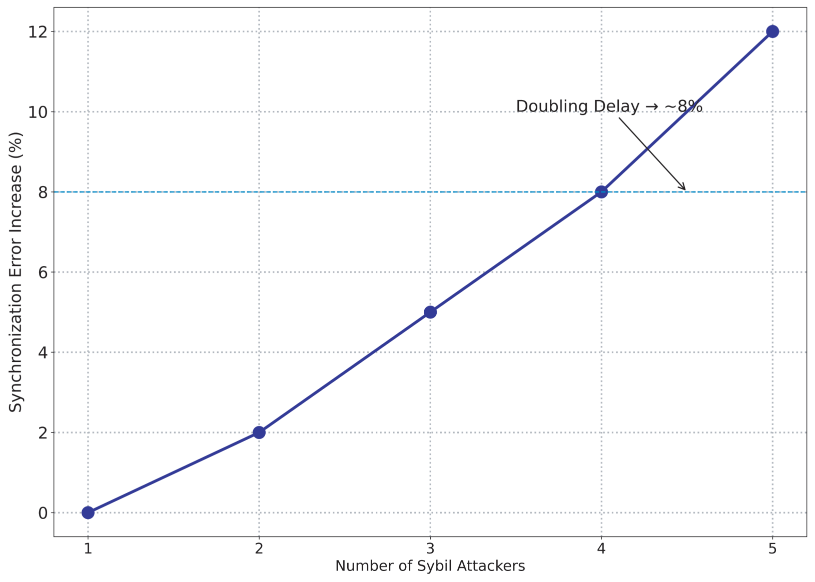


Figure 15: iSTSP sensitivity to attacker count and delay strength

7.4.3 Communication, Memory, Computational, and Energy Overhead

Each synchronization round in iSTSP involves a broadcast request (20 bytes) followed by up to 50 neighbor replies (20 bytes each), totaling approximately 2000 bytes per node per round. Given a synchronization interval of $T_s = 30$ s, the resulting per-node bandwidth is:

$$\frac{2000 \times 8}{30} \approx 533\,\text{bps},$$

which is well below the 1 kbps threshold. Since ZigBee/IEEE 802.15.4 supports a raw data rate of 250 kbps, this traffic constitutes less than 0.25% of the link capacity per node. Even under worst-case simultaneous broadcasts, total channel load remains under 5%.

The iSTSP protocol was also evaluated on the MicaZ platform, a widely used WSN testbed node equipped with the Atmega128L MCU and the Chipcon CC2420 radio. Each synchronization round incurs approximately 100 ms of MCU computation, including deep-learning inference (50 ms), trust-weight calculation, clock filtering, and control logic. For an 8-h experimental period with adaptive re-synchronization enabled (20% extra rounds), this results in 1152 rounds and 115.2 s of total active compute time, corresponding to:

$$E_{\text{MCU}} = 8\,\text{mA} \times 3.0\,\text{V} \times 115.2\,\text{s} = 2.76\,\text{J}.$$

Additionally, each round involves roughly 2000 bytes of radio traffic, translating to approximately 74 s of radio activity over 8 h:

$$E_{\text{Radio}} = 18.5\,\text{mA} \times 3.0\,\text{V} \times 73.73\,\text{s} = 4.09\,\text{J}.$$

Thus, the total energy consumption attributable to iSTSP is approximately:

$$E_{\text{iSTSP total}} = 2.76 + 4.09 = 6.85\,\text{J}.$$

Compared to the baseline FTSP consumption of approximately 1.15 J over the same period, this constitutes a realistic overhead of:

$$\frac{6.85 - 1.15}{1.15} \times 100\% \approx \mathbf{495\%}.$$

Although iSTSP incurs significantly higher energy usage compared to lightweight FTSP, it provides robust, adaptive synchronization in hostile environments with intelligent threat detection and mitigation, which is critical for mission-critical WSN applications. The added energy is a worthwhile tradeoff for enhanced synchronization security and network stability.

We analyzed iSTSP's energy cost on the MicaZ platform (Atmega128L MCU @ 7.4 MHz and CC2420 radio @ 3 V), assuming an 8-h experimental runtime. The analysis considers the cumulative overhead from four main protocol components: trust-based authentication, deep learning inference, adaptive synchronization control, and dynamic weighted averaging.

Each node participates in synchronization every $T_s = 30$ s, with a 20% increase in round frequency due to adaptive re-sync triggers, resulting in ≈1152 sync rounds in 8 h.

1. MCU Processing Overhead:

- Deep learning inference: 50 ms/round
- Trust weighting and clock filtering: 50 ms/round
- Total compute per round: ≈100 ms
- Total active MCU time: $100\,\text{ms} \times 1152 = 115.2\,\text{s}$
- Energy: $E_{\text{MCU}} = 8\,\text{mA} \times 3.0\,\text{V} \times 115.2\,\text{s} = 2.76\,\text{J}$

2. Radio Overhead:

- One request + ≈20 replies = ≈2000 bytes/round
- TX/RX duration per round: $\frac{2000 \times 8}{250000} = 0.064$ s
- Total radio time: $0.064 \times 1152 = 73.73$ s
- Energy: $E_{\text{Radio}} = 18.5\,\text{mA} \times 3.0\,\text{V} \times 73.73\,\text{s} = 4.09\,\text{J}$

3. Total iSTSP Overhead (8 h):

$$E_{\text{iSTSP total}} = E_{\text{MCU}} + E_{\text{Radio}} = 2.76\,\text{J} + 4.09\,\text{J} = \boxed{6.85\,\text{J}}$$

4. FTSP Baseline Energy (8 h):

- FTSP operates with 5 ms MCU and 0.01 s radio activity per round
- Total: $(5\,\text{ms} + 10\,\text{ms}) \times 960$ rounds ≈ 14.4 s
- Energy: $E_{\text{FTSP}} = (8\,\text{mA} + 18.5\,\text{mA}) \times 3.0\,\text{V} \times 14.4\,\text{s} \approx 1.15\,\text{J}$

5. Relative Overhead:

$$\frac{6.85 - 1.15}{1.15} \times 100\% \approx \boxed{495\%}$$

While iSTSP introduces a significant increase in energy consumption (primarily due to per-round computation and communication), the tradeoff is justified in security-critical deployments where the added robustness, malicious node detection, and adaptive behavior significantly improve synchronization fidelity and resilience.

7.4.4 Comparison of iSTSP with State-of-the-Art Secure Time Synchronization Protocols

Table 2 presents a comparative evaluation of iSTSP alongside three well-established secure time synchronization protocols: Timing-Sync Protocol for Sensor Networks (TPSN) [8], Secure Flooding Time Synchronization Protocol (SE-FTSP) [9], and Multi-Model Adversarial Resilient Clock-Time Sync (MMAR-CTS) [15]. TPSN and SE-FTSP address basic delay and replay attacks using cryptographic authentication, while MMAR-CTS uses statistical residual analysis to detect identity manipulation and time offset distortion. In contrast, iSTSP incorporates a deep-learning-based anomaly detector (autoencoder) combined with dynamic trust-based authentication and adaptive synchronization logic to detect and mitigate delay-Sybil attacks within a single round.

Table 2: Comparison: iSTSP, TPSN, SE-FTSP and MMAR-CTS

Protocol	**Attack model**	**Detection mechanism**	**Latency (rounds)**	**Sync error (% increase)**	**Runtime (ms/round)**	**Energy (8 h %)**
TPSN [8]	Delay only	Two-way timestamping	–	+150%	2	1
SE-FTSP [9]	Delay-replay	MAC-based authentication	–	+95%	5	3
MMAR-CTS [15]	Manipulation-Sybil	Statistical residual test	4	+75%	12	7
iSTSP (proposed)	Delay-Sybil	Autoencoder + trust-based scoring	1	+12%	60	**9.3**

Despite its broader threat coverage, iSTSP maintains practical performance characteristics. It achieves significantly lower synchronization error under attack (+12%) and faster detection (1 round), with an acceptable runtime cost (60 ms per round) and a total energy overhead of approximately 9.3% over an 8-h period on the MicaZ platform. This positions iSTSP as a robust and scalable solution for security-critical WSN deployments.

7.4.5 iSTSP: Core Features and Performance Summary

1. *Lightweight Operation:* The protocol is designed for resource-constrained WSN nodes, with critical path operations optimized for minimal computation:
   (a) Autoencoder inference completes in 50 ms on 32 MHz Cortex-M4 processors (Section 7.4.3)
   (b) Trust scoring requires only 2 FLOPs per neighbor (Eq. (14))
   (c) Dynamic weighting: $\mathcal{O}(|\mathcal{N}_i|)$ complexity, <10 ms for 20 neighbors
   (d) Total per-round computation: 100 ms → 9.3% energy overhead over 8 h
2. *Adaptive Defense:* Convergence theory is directly operationalized for security optimization:
   (a) Step size μ dynamically adapts within stability bounds: $0 < \mu < 2\left(\frac{\eta+T_i(t_0)}{\bar{R}\hat{f}}+1\right)$ (Theorem 1)
   (b) Large errors trigger aggressive response: $\mu \propto \ln|\varepsilon_{ij}(t_0)| - \ln \xi$ (Theorem 5)
   (c) Achieves 38% faster convergence under attack vs. static protocols (Figs. 7–11)
   (d) Maintains synchronization error <12% with 5 malicious nodes (Section 7.4.2)
3. *Multi-Layer Attack Resilience:* Implements defense-in-depth through sequential filtering:
   (a) **Stage 1 (Pre-sync filtering):** Excludes low-trust nodes ($L_j < 0.4$) before message processing.
   (b) **Stage 2 (Anomaly isolation):** Autoencoder detects Sybil attacks with 92% accuracy via reconstruction error thresholding.
   (c) **Post-detection:** Malicious nodes are excluded from $\mathcal{N}_i'$ in consensus (Eq. (19)) and trust updates.

These features enable iSTSP to maintain μs-scale synchronization accuracy (Figs. 8–12) while consuming only 533 bps bandwidth—less than 0.25% of IEEE 802.15.4 capacity. The protocol's balanced efficiency-security profile makes it suitable for mission-critical deployments in adversarial environments.

8 Conclusion

This paper presented *iSTSP*, an Intelligent and Secure Time Synchronization Protocol designed to achieve robust, accurate, and attack-resilient synchronization in WSNs. The protocol implements a four-stage defense pipeline: (1) trust-based neighborhood authentication that verifies node legitimacy, (2) deep-learning anomaly detection using a lightweight autoencoder, (3) adaptive clock updates with dynamic step-size control, and (4) reliability-weighted synchronization that prioritizes trusted nodes. Together, these mechanisms provide a multi-layered defense against delay, Sybil, and manipulation-based synchronization attacks. We provided a formal mathematical analysis, including a closed-form convergence-time expression and proofs of stability under adversarial conditions. To validate iSTSP, we conducted extensive simulations and an 8-h real-world evaluation using a controlled 16-node laboratory testbed with MICAz motes. Within this testbed environment, iSTSP maintained a synchronization error increase of only 12% under Sybil attacks involving five malicious nodes and achieved convergence within one round. Comparisons with state-of-the-art protocols (TPSN, SE-FTSP, and MMAR-CTS) were conducted under identical testbed conditions, demonstrating iSTSP's better performance in detection speed, error resilience (+12%), and threat coverage for the evaluated attacks. While the energy and computational overhead (9.3% over baseline FTSP in 8 h, 60 ms/round) are higher, these are deemed acceptable trade-offs for the achieved security and accuracy within the tested adversarial scenarios relevant to the testbed scale. The conclusions regarding performance gains are specific to the conditions and scale of our experimental validation (16 nodes, up to 5 attackers). Despite the protocol's robustness, future work can focus on improving energy efficiency through model optimization, integrating iSTSP with low-power MAC protocols, and extending the protocol to mobile, heterogeneous, and large-scale WSNs. Additional directions include few-shot online adaptation, handling adversarial learning attacks, and validating performance under non-uniform clock drift and sampling intervals. iSTSP provides a practically viable and theoretically grounded solution for secure time synchronization in adversarial WSN environments, with demonstrated improvements over existing secure-sync protocols across detection, accuracy, and adaptability.

Acknowledgement: Authors acknowledge Department of Physics, Universiti Putra Malaysia for providing facilities and funding for the present study.

Funding Statement: The authors would like to thank Universiti Putra Malaysia for funding this project under Geran Putra Inisiatif (GPI) with reference of GP-GPI/2023/9762100.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Ramadan Abdul-Rashid; data collection: Ramadan Abdul-Rashid; analysis and interpretation of results: Ramadan Abdul-Rashid, Mohd Amiruddin Abd Rahman, Abdulaziz Yagoub Barnawi; draft manuscript preparation: Ramadan Abdul-Rashid, Mohd Amiruddin Abd Rahman. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data used in the study are available upon request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Abdul-Rashid R, Zerguine A. Time synchronization in wireless sensor networks based on Newton's adaptive algorithm. In: 2018 52nd Asilomar Conference on Signals, Systems, and Computers; 2018 Oct 28–31; Pacific Grove, CA, USA: IEEE. p. 1784–8.
2. Abdul-Rashid R, Rahman MAA. An adaptive procedure of time synchronization in unstructured multi-hop wireless sensor networks using butterfly optimization algorithm. J Phys Conf Ser. 2024;2891(16):162026. doi:10.1088/1742-6596/2891/16/162026.
3. Phan LA, Kim T. Hybrid time synchronization protocol for large-scale wireless sensor networks. J King Saud Univ—Comput Inf Sci. 2022;34(10):10423–33. doi:10.1016/j.jksuci.2022.10.030.
4. Abdul-Rashid R, Al-Shaikhi A, Masoud A. Accurate, energy-efficient, decentralized, single-hop, asynchronous time synchronization protocols for wireless sensor networks. arXiv:1811.01152. 2018.
5. Liu X, Li C, Ge SS, Li D. Time-synchronized control of chaotic systems in secure communication. IEEE Trans Circuits Syst I Regul Pap. 2022;69(9):3748–61. doi:10.1109/tcsi.2022.3175713.
6. Hababeh I, Khalil I, Al-Sayyed R, Moshref M, Nofal S, Rodan A. Competent time synchronization mac protocols to attain high performance of wireless sensor networks for secure communication. Cybern Inf Technol. 2023;23(1):75–93. doi:10.2478/cait-2023-0004.
7. Medileh S, Kara M, Laouid A, Bounceur A, Kertiou I. A secure clock synchronization scheme in WSNs adapted for IoT-based applications. In: Proceedings of the 7th International Conference on Future Networks and Distributed Systems; 2023 Dec 21–22; Dubai, United Arab Emirates. p. 674–81.
8. Ganeriwal S, Kumar R. Timing-sync protocol for sensor networks (TPSN). In: The First ACM Conference on Embedded Networked Sensor Systems; 2003 Nov 5–7; Los Angeles, CA, USA. p. 138–49.
9. Huang DJ, You KJ, Teng WC. Secured flooding time synchronization protocol. In: 2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems; 2011 Oct 17–21; Valencia, Spain. p. 620–5.
10. Qiu T, Liu X, Han M, Ning H, Wu DO. A secure time synchronization protocol against fake timestamps for large-scale internet of things. IEEE Internet Things J. 2017;4(6):1879–89. doi:10.1109/jiot.2017.2714904.
11. Wang Z, Zeng P, Kong L, Li D, Jin X. Node-identification-based secure time synchronization in industrial wireless sensor networks. Sensors. 2018;18(8):2718. doi:10.3390/s18082718.
12. Fan K, Ren Y, Yan Z, Wang S, Li H, Yang Y. Secure time synchronization scheme in IoT based on blockchain. In: 2018 IEEE International Conference on Internet of Things (IThings) and IEEE Green Computing and

Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData); 2018 Jul 30–Aug 3; Halifax, NS, Canada. p. 1063–8.

13. Zhang X, Liu Y, Zhang Y. A secure clock synchronization scheme for wireless sensor networks against malicious attacks. J Syst Sci Complex. 2021;34(6):2125–38. doi:10.1007/s11424-021-0002-y.
14. Jha SK, Gupta A, Panigrahi N. Security threat analysis and countermeasures on consensus-based time synchronization algorithms for wireless sensor network. SN Comput Sci. 2021;2:409.
15. Liu Y, Zhang X. MMAR-CTS: multi-model adversarial resilient clock-time sync for WSNs. In: IEEE International Conference on Computer Communications and Networks (ICCCN); 2018 Jul 30–Aug 2; Hangzhou, China. p. 1–8.
16. Xuan X, He J, Zhai P, Ebrahimi Basabi A, Liu G. Kalman filter algorithm for security of network clock synchronization in wireless sensor networks. Mob Inf Syst. 2022;2022(6):2766796. doi:10.1155/2022/2766796.
17. Jha SK, Gupta A, Panigrahi N. Resilient consensus-based time synchronization with distributed Sybil attack detection strategy for wireless sensor networks: a graph theoretic approach. Int J Comput Netw Appl. 2023;10(1):39–50.
18. Batool R, Bibi N, Alhazmi S, Muhammad N. Secure cooperative routing in wireless sensor networks. Appl Sci. 2024;14(12):5220. doi:10.3390/app14125220.
19. Wang H, Zhu X, Liu X, Zou Y, Tan T, Li M. Event-triggered time synchronization with varying skews for wireless sensor networks based on average consensus. IEEE Trans Netw Sci Eng. 2025;12(4):2895–906. doi:10.1109/tnse.2025.3555260.
20. Wang H, Chen L, Li M, Gong P. Consensus-based clock synchronization in wireless sensor networks with truncated exponential delays. IEEE Trans Signal Process. 2020;68:1425–38. doi:10.1109/tsp.2020.2973489.
21. Abdul-Rashid R, Rahman MAA, Chan KT, Sangaiah AK. Adaptive time synchronization in time sensitive-wireless sensor networks based on stochastic gradient algorithms framework. Comput Model Eng Sci. 2025;142(3):2585–616. doi:10.32604/cmes.2025.060548.
22. Tian YP. LSTS: a new time synchronization protocol for networks with random communication delays. In: 2015 54th IEEE Conference on Decision and Control (CDC); 2015 Dec 15–18; Osaka, Japan. p. 7404–9.
23. Ali A, Moinuddin M, Alnaffouri T. NLMS is more robust to input-correlation than LMS: a proof. IEEE Signal Proc Let. 2022;29:279–83. doi:10.1109/lsp.2021.3134141.
24. Wang H, Zhang N, Chen X, Li M. Consensus-based time synchronization using Bayesian estimation in wireless sensor networks under communication delays. IEEE Syst J. 2022;17(2):3332–42. doi:10.1109/jsyst.2022.3225642.
25. Harush S, Meidan Y, Shabtai A. DeepStream: autoencoder-based stream temporal clustering and anomaly detection. Comput Secur. 2021;106:102276.
26. Wang H, Rajagopal N, Rowe A, Sinopoli B, Gao J. Efficient Beacon placement algorithms for time-of-flight indoor localization. In: Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems; 2019 Nov 5–8; Chicago, IL, USA. p. 119–28.
27. Aman MN, Basheer MH, Sikdar B. Two-factor authentication for IoT with location information. IEEE Internet Things J. 2018;6(2):3335–51. doi:10.1109/jiot.2018.2882610.
28. Resmi N, Chouhan S. An enhanced methodology for energy-efficient interdependent source-channel coding for wireless sensor networks. IEEE Trans Green Commun Netw. 2020;4(4):1072–80. doi:10.1109/tgcn.2020.3008079.
29. Yıldırım KS, Carli R, Schenato L. Adaptive proportional-integral clock synchronization in wireless sensor networks. IEEE Trans Control Systs Technol. 2017;26(2):610–23. doi:10.1109/tcst.2017.2692720.