electronics

MDPI

Article

An Adaptive Trust-Based Approach to the Protection of Software-Defined Wireless Sensor Networks Against Multiple Types of DoS Attacks

Lijuan Wang, Krassie Petrova * and Mee Loong Yang

School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland 1010, New Zealand; lijuan.wang@autuni.ac.nz (L.W.); bobby.yang@aut.ac.nz (M.L.Y.)
* Correspondence: krassie.petrova@aut.ac.nz

Abstract

Software-defined wireless sensor networks (SDWSNs) are deployed in mission-critical applications such as environmental monitoring, smart cities, and healthcare. However, existing protocol architectures such as SDN-WISE lack built-in mechanisms for the detection or mitigation of Denial-of-Service (DoS). Existing trust-based security solutions are rarely fully integrated into the protocol stack; nearly all of them rely on manually tuned attack-specific detection thresholds and metric weights, which severely undermine their effectiveness. This study integrates the adaptive trust model ATAW-TM into the SDN-WISE protocol stack; the ATAW-TM model was adjusted to the resource-constrained sensor node environment by embedding three lightweight adaptations that void the invocation of the exponential and logarithmic software library functions: an inverse-quadratic kernel of Gaussian-like cooperation probability, a quadratic form of second-order Rényi (replacing Shannon entropy) for entropy weight, and a softsign for the aging factor. To evaluate the integrated model, we developed a comprehensive evaluation framework encompassing 14 distinct DoS attack types and the data tampering attack against SDWSNs. The integrated model was evaluated through extensive Cooja-based simulations. The experimental results indicate that the proposed security solution effectively addresses the majority of the DoS attack range and the data-tampering attack through attack detection, abnormality identification, and mitigation. However, for grey hole and sinkhole attacks, the attacker's trust value stabilizes at a level lower than that of legitimate nodes but above the detection threshold; second, the detection precision of flooding-type attacks is relatively low and may result in victim nodes being wrongly classified as malicious. These security enhancements are achieved with only negligible performance overhead, introducing less than 5.3% additional hop delay and 8.1% additional forwarding delay compared with the standard SDN-WISE. Furthermore, simulation results show that the proposed method can adapt to variations in network traffic conditions without requiring manual threshold tuning. Overall, the proposed lightweight trust-based attack recognition model offers a practically achievable security solution for resource-constrained SDWSNs, without sacrificing protocol efficiency.

Keywords: denial-of-service (DoS) attacks; trust model; software-defined wireless sensor networks (SDWSNs); Contiki; Cooja

Check for updates

Academic Editor: Stefano Scanzio

Received: 14 August 2026
Revised: 24 October 2026
Accepted: 26 October 2026
Published: 2 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

1.1. Software-Defined Wireless Sensor Networks

Software-defined wireless sensor networks (SDWSNs) decouple the control plane from the data plane of the network, enabling centralized network management, dynamic reconfiguration, and improved resource utilization [1]. By integrating the flexibility of software-defined networking (SDN) into the resource-constrained nature of wireless sensor networks (WSNs), SDWSNs achieve efficient routing, adaptive flow management, and simplified network programming [2]. As WSNs are increasingly deployed in mission-critical applications such as environmental monitoring [3], smart cities, and healthcare [4,5], and given that SDWSNs represent the future trend in WSN deployment [6], ensuring the security of SDWSNs has become a priority.

1.2. SDN-WISE and Its Vulnerability

SDN-WISE is a stateful SDN protocol architecture for wireless sensor networks that stands out as a lightweight and energy-efficient solution specifically designed for constrained IoT environments [7]. SDN-WISE extends the SDN paradigm to WSNs by introducing a finite-state-machine-based forwarding mechanism, where sensor nodes maintain flow tables and process packets according to rules installed by the controller. Subsequently, they extended the SDN-WISE framework by introducing network function virtualization and developing approaches for enforcing regulatory compliance on sensor nodes [8]. While SDN-WISE provides significant advantages in terms of enhanced programmability and network visibility, its current specification lacks built-in mechanisms for detecting and mitigating DoS attacks. Malicious nodes can exploit this limitation by injecting forged packets, replaying legitimate flows, or launching flooding attacks that deplete node energy and bandwidth. Consequently, an unmodified SDN-WISE deployment remains vulnerable to a number of DoS attack vectors, including blackhole, grey hole, flooding, and selective forwarding attacks [9].

1.3. Existing Defense Approaches and Their Limitations

Existing approaches to DoS attack mitigation in WSNs can be broadly categorized into cryptographic [10], anomaly-based [11], and trust-based approaches. Cryptographic solutions (e.g., digital signatures, message authentication codes) provide strong authentication but incur significant computation and communication overhead and cannot deal with internal attacks triggered by node capture [12]. Anomaly-based detection systems that use machine learning to identify deviations from normal behavior typically require large training datasets and centralized processing; this may introduce latency and impede scalability [13]. In trust-based models, nodes evaluate the trustworthiness of their peers based on observed behaviors such as packet sending rate, number of packets received, and residual energy rate [14]. Notable trust models for WSNs include a secure trust-aware RPL routing protocol for the Internet of Things (SecTrust-RPL) [15], Realistic Weight-Based Reliable Trust Model for Large Scale WSNs (RTM) [16], Securing Software-Defined WSNs Communication via Trust Management (TSW) [17], and an Energy-efficient Trust Management and Routing Mechanism for SDWSNs (ETMRM) [18]. However, several limitations still persist: (i) reliance on manually tuned thresholds that are sensitive to network dynamics [17]; (ii) lack of adaptivity to changing attack patterns [18]; and (iii) only a few trust-based models have been implemented in Contiki or other operating systems [15]. More importantly, only one of the existing trust models (ETMRM) has been specifically tailored to the SDN-WISE protocol architecture. This leaves a critical research gap in protocol-level DoS attack recognition in SDWSNs.

https://doi.org/10.3390/electronics15215084

1.4. Research Approach

To address the limitations of the current approaches to defending WSNs against DoS attacks and, in particular, to equip SDN-WISE (developed by the University of Catania, Catania, Italy) with a practical, lightweight DoS attack recognition capability, this paper integrates our previously developed ATAW-TM [19] into the SDN-WISE protocol stack. ATAW-TM is a trust model specifically designed for SDWSNs that eliminates reliance on manually tuned attack-specific detection thresholds and metric weights. This adaptive model for attack detection does not rely on manually tuning traffic-rate-specific thresholds for different operating conditions. Unlike many existing trust models that require predefined packet-rate thresholds to distinguish between normal and malicious behavior [17,18], ATAW-TM evaluates node behavior relative to the behavioral distribution of neighboring nodes observed by the same evaluator. As a result, the model can operate under different normal traffic rates without requiring parameter reconfiguration.

For deployment on resource-constrained sensor nodes, we adopted three lightweight standard functions: an inverse-quadratic kernel for cooperation probability estimation, a quadratic form of second-order Rényi entropy instead of Shannon entropy for entropy weight calculation, and a scaled and shifted softsign function for modeling temporal trust decay through an aging factor mechanism. The adaptations reduce the computational overhead of ATAW-TM to fit within the memory and processing constraints of typical WSN motes (e.g., TI CC2538, Atmel AVR) without compromising detection effectiveness.

We conduct an extensive performance evaluation of the enhanced SDN-WISE prototype using Cooja-based simulations. Our evaluation encompasses 14 distinct DoS attack scenarios, including blackhole, grey hole, flooding, selective forwarding, and their variants, as well as the data tampering attack; the scenarios used in the evaluation were selected with the purpose of providing comprehensive coverage of adversarial behaviors targeting SDWSNs. Experimental results demonstrate that our integrated solution successfully detected most evaluated attacks, identified control-plane blackhole attacks through abnormal trust-update patterns, and mitigated bad-mouthing attacks and good-mouthing attacks; compared to the standard SDN-WISE, the overhead was negligible. However, while the attacker's trust value in the grey hole and sinkhole attack scenarios stabilizes below that of legitimate nodes, it is still above the detection threshold. Flooding-type attacks can cause legitimate victim nodes to be misidentified as malicious, with precision as low as 0.57.

Furthermore, we assess the adaptability of our proposed solution under various network traffic rates to show its robustness across different operational conditions. Overall, the results confirmed that the proposed lightweight trust-based attack recognition model was feasible without sacrificing protocol efficiency and could be used in real-life implementations of resource-constrained SDWSNs.

1.5. Contributions

The main contributions of this paper are as follows:

(i) Trust-Integrated SDN-WISE protocol stack: To the best of our knowledge, this is the first study to integrate ATAW-TM (an adaptive trust model that operates without attack-specific threshold tuning and applies an automatic weight-assigning approach) into the SDN-WISE protocol stack. This integration enables protocol-native recognition of multiple DoS attack types and data tampering attacks.

(ii) Lightweight Embedded Adaptation of ATAW-TM: We integrate three lightweight standard functions into ATAW-TM to adapt the model to the resource-constrained sensor nodes.

(iii) Comprehensive evaluation framework: We developed a comprehensive evaluation framework comprising a significant range of attacks on SDWSNs (14 DoS attacks and

the data tampering attack) on an SDWSN that can be used to test the performance of new security-enhancing integrations.

The remainder of this paper is structured as follows. Section 2 describes the design of the integrated system, the experimental configuration, the simulation parameters, the attack models, and the evaluation metrics. Sections 3 and 4 present and discuss the experimental results. Section 5 concludes the paper and suggests directions for future research.

2. Materials and Methods

This section presents the materials and methods employed in this study, including experimental hardware and software platforms and a detailed description of the design of the integrated trust-aware SDN-WISE system. Subsequently, we present the attack models implemented at the evaluation stage, the simulation parameters, and the performance evaluation metrics used.

2.1. Threat Model

2.1.1. Trust Assumptions

The SDN controller is assumed to be physically protected and trusted. Sensor nodes are assumed to be resource-constrained devices that can be compromised by an adversary. All nodes except the compromised ones are assumed to be honest.

2.1.2. Adversary Capabilities

A compromised node can: (i) drop, modify, replay, or fabricate data and control packets, including BEACON, DATA, REPORT, and flow-request packets; selectively misbehave toward specific sources or during specific time windows to evade detection; (ii) overhear transmissions within its radio range due to the promiscuous mode of IEEE 802.15.4 radios; (iii) inject forged identities or manipulate routing metrics; and (iv) generate excessive traffic to exhaust node energy and bandwidth.

2.1.3. In-Band Trust Reports

Trust reports are transmitted in-band as REPORT packets through the same multi-hop data plane used for normal traffic. Consequently, an on-path adversary can drop, delay, or alter trust reports before they reach the controller. This creates two residual risks:

1. Report dropping: an on-path attacker can prevent trust reports from downstream nodes from reaching the controller. This is illustrated by the case of the control-plane blackhole attack (Section 3.2.1), where the controller identifies the anomaly through the prolonged absence of trust-value updates rather than through a decline in trust values.
2. Report alteration: an on-path attacker can modify trust reports to inflate its own trust value or defame other nodes. The controller-side weighted voting mechanism mitigates the risk by assigning lower weight to reports that deviate from the global consensus. The effectiveness of this mitigation approach is demonstrated in the bad-mouthing and good-mouthing attack scenarios (Sections 3.2.7 and 3.2.8, respectively).

2.1.4. Attack and Defense Scope

We do not consider adversaries that can compromise the controller, jam the wireless channel at the physical layer, or break the underlying cryptographic primitives. Second, protecting trust reports with message authentication codes or digital signatures is not considered in this study, partly because it would introduce an additional computation and communication overhead on resource-constrained nodes.

2.2. Experimental Platform

2.2.1. Hardware Platform

All experiments were conducted using the Cooja (Contiki version 3.0) network simulator (open-source, Contiki community), which is part of the Contiki version 3.0 operating system. The simulated sensor nodes were configured as Sky motes (Crossbow Technology, San Jose, CA, USA), emulating the hardware characteristics of the Texas Instruments MSP430F1611 microcontroller (Texas Instruments, Dallas, TX, USA) and the CC2420 IEEE 802.15.4 radio transceiver (Texas Instruments, Dallas, TX, USA). According to the platform configuration file (platform-conf.h) from the Contiki source tree, the following hardware parameters were used:

- MCU: MSP430 microcontroller (Texas Instruments, Dallas, TX, USA) operating at 3.9 MHz.
- Memory: 10 kB RAM and 48 kB ROM (consistent with MSP430F1611 specifications).
- Radio: IEEE 802.15.4-compliant transceiver (Texas Instruments, Dallas, TX, USA) operating at 2.4 GHz.
- System clock resolution: 512 Hz.

These specifications are representative of typical resource-constrained wireless sensor nodes, making them suitable for evaluating the lightweight nature of the proposed trust integration. We modified the system clock resolution from 128 Hz to 512 Hz because, in our experiment, the node cannot hear neighbors' sending or receiving packets immediately under a 128 Hz system clock resolution. A delay in hearing will negatively affect the accuracy of the final calculated trust value.

2.2.2. Software Platform

The base protocol architecture for our experiments was SDN-WISE, implemented within the Contiki environment. We used the open-source SDN-WISE Contiki implementation [20], which includes the Contiki-based sensor node platform. The original SDN-WISE implementation was extended by embedding the ATAW-TM trust model into the sensor node, while trust aggregation capability was also incorporated into the controller side. The controller was implemented as a Java-based application running on the host machine, based on the SDN-WISE Java controller framework [21], and communicated with sensor nodes via serial links emulated by Cooja. The code and the data used in the experiment are available as Supplementary Materials.

2.3. System Design: Embedding the ATAW-TM Model into the SDN-WISE Protocol Stack

This subsection describes the design of our integrated system, which embeds the ATAW-TM trust model directly into the SDN-WISE protocol stack. We implemented three primary components: a trust evidence collection mechanism that continuously observes both direct and overheard packet transmissions, a node-level trust computation engine, and a controller-level trust aggregation mechanism.

2.3.1. Overview of the Integrated Architecture

Figure 1 illustrates the implementation of ATAW-TM within the SDN-WISE architecture. ATAW-TM is integrated into the existing SDN-WISE sensor-node and SDN-controller components. The unshaded components represent the original components of SDN-WISE, whereas the shaded components indicate the ATAW-TM components implemented and integrated into the SDN-WISE architecture in this work. Solid arrows indicate that the lower layer provides services to the layer above; for example, the Physical Layer serves the IEEE 802.15.4 Radio MAC Layer. Dashed arrows indicate that the Sensor Node sends messages to

the Controller to report trust values. For clarity, only the SDN-WISE components relevant to the integration of the trust model are shown in the figure. A more detailed architecture of SDN-WISE can be found in [7,8]. In this trust-enhanced SDN-WISE system, each sensor node runs a local trust computation engine that continuously evaluates the trustworthiness of its neighbors based on observed behaviors. The computed trust values are stored in an extended neighbor table. Periodically, nodes report local trust values to the SDN controller, which performs global trust aggregation.

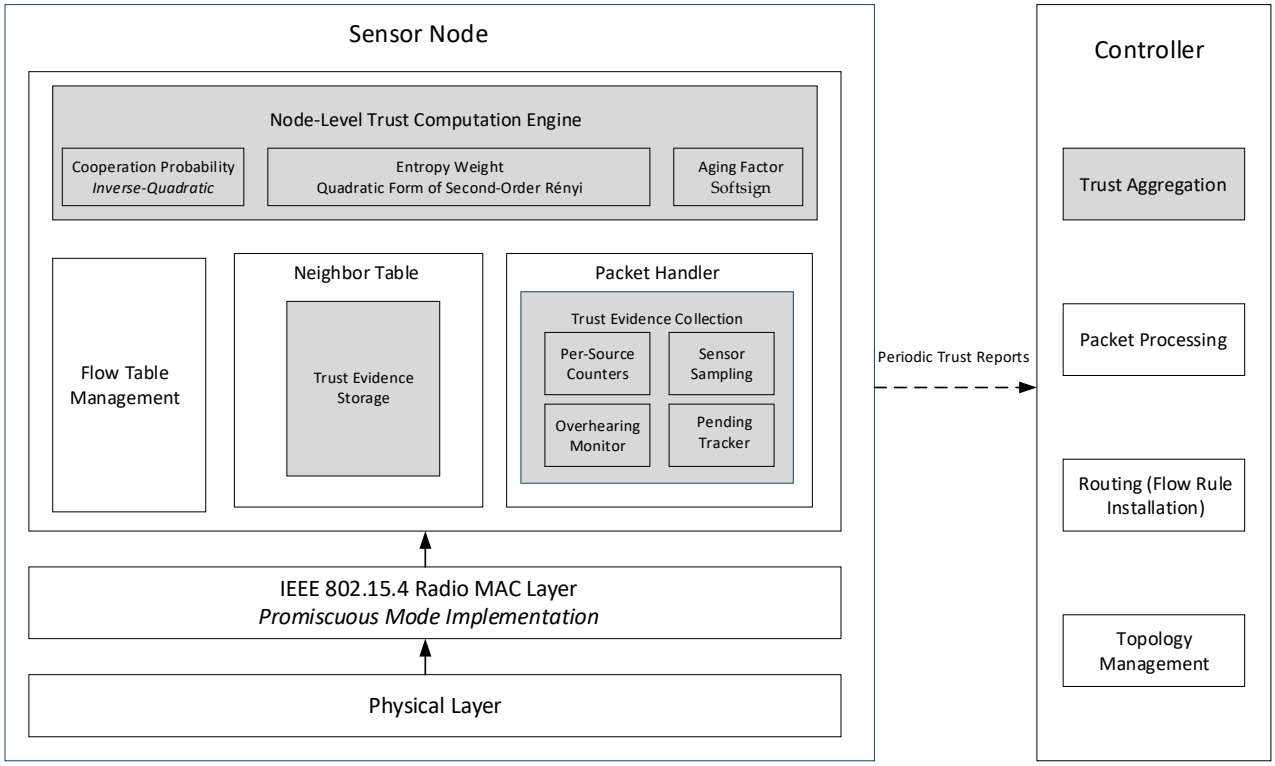


Figure 1. The implementation of ATAW-TM within the SDN-WISE architecture.

ATAW-TM: Metrics and Collational Characteristics

A passive monitoring framework that leverages the promiscuous mode capability of the IEEE 802.15.4 radio forms the foundation of the integration architecture. In the Cooja simulation environment, the underlying radio model inherently allows all nodes within the transmission range to receive the transmitted frames. To enable overhearing, we modified the protocol stack to bypass the destination address filtering. Consequently, any frame successfully received during the ContikiMAC wake-up windows is passed to the upper layers for trust evidence extraction. This software-level modification leverages the existing duty-cycling mechanism: nodes process overheard packets only when their radios are already active for normal reception, thereby avoiding the need for continuous listening.

For every observed packet, the node extracts behavioral evidence for eight trust metrics, including the data packet sending rate (DSR), control packet sending rate (CSR), data packet receiving rate (DRR), control packet receiving rate (CRR), data packet forwarding rate (DFR), control packet forwarding rate (CFR), energy consumption rate (ECR), and data accuracy (DA), as defined in [19]. A pending forwarding tracker maintains temporary entries for packets expected to be forwarded by neighboring nodes, enabling accurate verification of forwarding behavior with configurable timeouts. This eavesdropping-based approach significantly increases the sample size for trust evaluation without generating additional control traffic.

Each node runs the ATAW-TM trust computation engine, which continuously evaluates the trustworthiness of its neighbors. The three lightweight functions used in this work (i.e., an inverse-quadratic kernel, a quadratic form of second-order Rényi entropy, and a softsign function) are known mathematical functions. In this research, we integrated the three functions into the SDN-WISE protocol stack on MSP430-class nodes. The specific advantage of these functions in the embedded context is that they avoid calls to exp() and log(): executing such calls would require software floating-point library support on the MSP430-class nodes, since they lack a hardware floating-point unit. The computed per-neighbor trust values are stored in an extended neighbor table.

Periodically, nodes report locally computed trust scores to the SDN controller. The controller aggregates the reports received from multiple nodes using a weighted voting mechanism, where reports from nodes with higher historical accuracy receive greater weight. The aggregated global trust values enable the controller to identify persistently malicious nodes.

2.3.2. Trust Evidence Collection

To support the ATAW-TM trust model with reliable behavioral observations, each node continuously monitors both direct and overheard packet transmissions.

Eavesdropping-Based Monitoring

Unlike conventional trust schemes that rely solely on firsthand observations, each node exploits the promiscuous mode capability of the IEEE 802.15.4 radio to monitor all packets transmitted within its radio range, regardless of their intended next hop. The eavesdropping mode provides two key advantages: (i) it increases the sample size for trust evaluation without additional control overhead, and (ii) it enables the detection of misbehaviors that would otherwise remain hidden, such as selective dropping or grey-hole attacks.

Behavioral Metrics Collected

For each monitored packet, the node collects evidence data for the metrics shown in Table 1:

Table 1. Evidence collected for each metric.

Metric	Evidence Collected From
DFR/CFR	Matching pending entries with observed forwards
DSR/DRR/CSR/CRR	Sequence number tracking
ECR	BEACON packet payload
DA	DATA packet payload

Pending Forwarding Tracker

A pending forwarding table maintained at node level is used to evaluate accurately whether a node actually forwards packets it is expected to forward. When a node overhears a packet destined to a neighbor (i.e., the node itself is not the next hop), it records a pending entry.

The pending tracker operates as follows:

1. Pending creation: When a node overhears a packet where the next hop (nxh) is not the final destination, it creates a pending forwarding entry for the expected forwarder.
2. Success verification: When a subsequent transmission from the expected forwarder is observed (either directly or through eavesdropping), the pending forwarding entry is 'matched'. A successful match occurs when all the following conditions hold:

(i) Source and destination address match.
(ii) Sequence number matches.
(iii) Packet type matches.
(iv) TTL (Time-to-Live) has decreased (indicating the packet was processed and forwarded).

3. Timeout mechanism: Each pending entry has a watchdog timeout of 30 × CLOCK_SECOND (30 s). If a pending forwarding entry is not matched within this window, the pending entry is automatically cleared, preventing memory leaks.

Per-Sender Trust Statistics

To support per-source trust differentiation, the neighbor table maintains separate counters for each data source. As shown in the code below, the design enables the trust model to distinguish between node-level trust (an indicator of overall reliability) and per-flow trust (behavior toward specific sources). This is critical for detecting selective forwarding attacks where a node drops packets from certain sources only.

```
for(int j = 0; j < TM_MAX_SENDER; j++)
{
if(index_for_forward_record[j] == packet->header.src.u8[1])
{
fwd->data_forwarded_success[j]++;
break;
}
}
```

Sequence Number-Based Packet Counting

To accurately count packet transmissions, the integrated system implements sequence number tracking. In this approach, the modulo 256 arithmetic correctly accounts for sequence number overflow (wrap-around). Out-of-order packets are handled by stipulating that they are not counted as new transmissions. This allows the monitoring node to tolerate missed packets caused by temporary link congestion while preventing duplicated or previously observed packets from being counted as new transmissions.

The collected statistics on each neighbor's packet transmissions, packet receptions, and packet forwarding activities are stored in the corresponding counters of the neighbor table. These counters are subsequently used for trust evaluation.

Data Value Sampling for Anomaly Detection

The monitoring function extracts sensor readings (e.g., temperature) from DATA packets and stores them in a fixed-size sliding window of recent observations. The sliding window preserves the most recent sensor readings for each neighbor and provides the input for subsequent trust evaluation. Significant deviations from the recent readings may indicate a data tampering attack or node malfunctions.

Energy and Link Quality Monitoring

Two pieces of information carried by BEACON packets are used for trust evaluation. The energy level (energy_level) reports the node's battery level, which should decrease gradually. Sudden increases or inconsistencies may indicate false reporting. The LQI (lqi) is aggregated from received BEACON packets to assess link stability. Poor LQI in most neighbors may indicate channel conditions rather than malicious behavior.

This trust evidence collection framework provides the raw behavioral data required by the ATAW-TM trust computation engine described in Section 2.3.3, integrated within the resource-constrained SDN-WISE deployments.

2.3.3. Node-Level Trust Computation Engine

The node-level trust computation engine implements the core ATAW-TM logic, continuously evaluating the trustworthiness of each neighbor based on multiple behavioral metrics. To accommodate the severe resource constraints of sensor motes, the engine employs the three lightweight standard functions described below. The complete trust computation is triggered periodically by the calculate_trust_values() function, which performs the following steps.

LQI-Based Trust Evaluation Gate

Before performing any trust computation, the engine evaluates the overall link quality using an LQI consistency check function. This function calculates the mean LQI for each neighbor from received beacons. If the majority of neighbors (more than 60%) exhibit LQI above 219 (this value is a reference to [22]), the evaluation proceeds; otherwise, it is suspended to avoid attributing channel-related losses to malicious behavior.

Cooperation Probability Calculation Using Inverse-Quadratic Kernel

The cooperation probability for each metric is computed using an inverse-quadratic kernel of the form $1/(1 + z^2)$, a standard radial basis function [23].

As shown in Figure 2, the inverse-quadratic kernel preserves the characteristic shape of a Gaussian distribution while requiring only multiplication and division operations. The condition std_dev $\leq$ 0.30f $\times$ mean serves as a consistency gate. When the standard deviation is less than or equal to 30% of the mean, the dataset exhibits low dispersion, indicating that the node's behavior is stable and predictable. Under such consistent conditions, the cooperation probability is assigned a value of 1.0 (full cooperation).

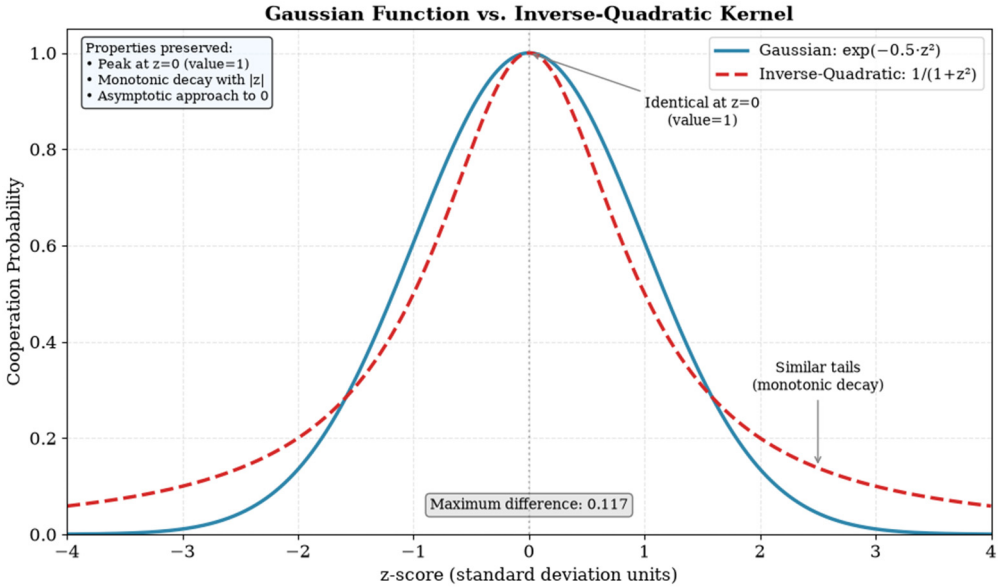


Figure 2. Comparison of the Gaussian cooperation probability function and inverse-quadratic kernel cooperation probability function.

This design choice has two key benefits. It reduces unnecessary computation when data is already trustworthy and prevents the inverse-quadratic kernel from generating false

low probabilities due to numerical instability when std_dev approaches zero. The threshold of 0.30 was empirically determined to balance sensitivity to genuine behavioral changes while tolerating minor natural variations in network conditions.

In the initial phase of network deployment, sensor nodes start up sequentially rather than simultaneously. Consequently, a node that starts early will overhear a large number of packets from its already-active neighbors, while it may observe very few packets from neighbors that start later. This asymmetric observation window can lead to false trust reports and significantly slow down trust convergence: early starting nodes may be incorrectly perceived as more trustworthy simply due to the higher packet counts, while late-starting, legitimate nodes may be unfairly penalized for having fewer observations.

To address this cold-start problem, we introduced a debouncing mechanism in the cooperation probability calculation, as follows: If the total number of data packets or control packets transmitted by a node within a single evaluation period (30 s) is 10 or less, the cooperation probability is assigned a value of 1 (indicating full cooperation). Otherwise, when sufficient samples (>10 packets) are available, the cooperation probability is calculated according to our standard algorithm. This threshold-based assignment ensures that nodes with insufficient observation samples are not unfairly penalized during the network bootstrap phase.

Entropy Weight Calculation Using Quadratic Form of Second-Order Rényi Entropy

We employ a quadratic entropy measure, the Gini-Simpson index $1 - \sum_i p_i^2$, for entropy weight calculation [24]. This quantity is a monotonic transform of the second-order Rényi entropy, $-\log\left(\sum_i p_i^2\right)$, and preserves its ordering while avoiding the logarithmic operation [25]. Rényi entropy-based methods have been successfully applied in sensor networks for uncertainty measurement [26]. By using the quadratic form $1 - \sum_i p_i^2$ instead of the logarithmic form of second-order Rényi entropy $\left(-\log\left(\sum_i p_i^2\right)\right)$, the implementation eliminates expensive logarithmic operations, making it suitable for resource-constrained sensor nodes.

Figure 3 illustrates the entropy weight comparison between Shannon entropy and the quadratic Rényi entropy across four synthetic metrics with distinct distribution characteristics. Each metric consists of four values, representing the trust scores of that metric across four neighboring nodes. The upper panels show the probability distributions of each metric: Metric A is highly skewed ($[0.04, 0.15, 0.30, 0.96]$), Metric B is moderately skewed ($[0.30, 0.60, 0.70, 0.97]$), Metric C is slightly skewed ($[0.60, 0.75, 0.80, 0.95]$), and Metric D is nearly uniform ($[0.94, 0.95, 0.95, 0.96]$). The lower panel presents the entropy weights (λ) computed by both methods. A larger spread among the trust scores of a metric, as seen in Metric A, indicates that this metric is actively affected by attacks and thus possesses stronger discriminative power, warranting a higher weight. In contrast, when trust scores are nearly identical across neighbors, as in Metric D, the metric remains stable and contributes little to distinguishing malicious nodes, thereby receiving a near-zero weight.

Aging Factor with Softsign Function

The engine employs a linear transformation of the softsign function, $f(z) = \frac{1}{2} \cdot \frac{z}{1+|z|} + \frac{1}{2}$, which maps its range from $(-1, 1)$ to $(0, 1)$, to model temporal trust decay. The softsign function, $softsign(z) = \frac{z}{1+|z|}$, is a standard activation function in neural networks [27]. It avoids calls to $\exp()$ and preserves the monotonicity and S-shape of the standard logistic sigmoid. The aging factor determines how much historical trust influences the updated trust value.

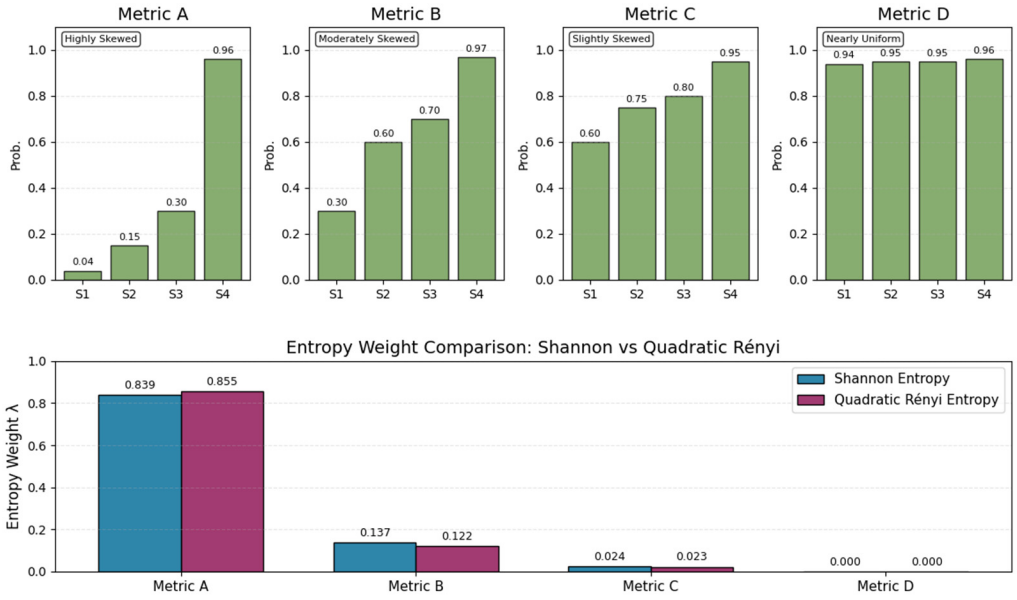


Figure 3. Normalized entropy comparison between Shannon entropy and quadratic Rényi entropy.

Figure 4 illustrates the aging factor. Figure 4a compares the standard logistic sigmoid with the scaled and shifted softsign function. The latter avoids exp() calls while preserving monotonicity, the S-shape, and the range (0, 1). Figure 4b is a plot of the aging factor α as a function of trust change ΔTrust for different steepness parameters k. The steepness parameter $k = 10$ and the midpoint $x_0 = 0$ were chosen for the aging factor calculation. This configuration produces a sharp transition around $\Delta\text{Trust} = 0$, meaning that any negative change in trust value (indicating potential malicious behavior) will result in an aging factor $\alpha < 0.5$. As a result, a larger weight is assigned to the current trust value that is lower than the historical trust value, causing rapid trust decay. Conversely, positive trust changes yield $\alpha > 0.5$, giving more weight to the historical trust value (which is lower than the current trust value) and enabling slower but steady trust recovery. This asymmetric behavior is critical for the timely detection of on–off attacks, where adversaries periodically alternate between cooperative and malicious behavior. The value $k = 10$ provides sufficient sensitivity without causing instability, while $x_0 = 0$ creates a natural threshold at zero trust change.

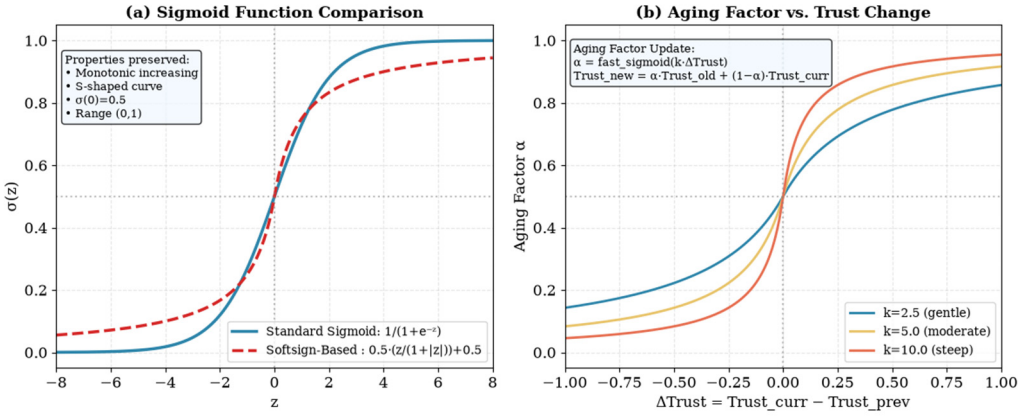


Figure 4. Softsign-based aging factor calculation.

Counter Reset and Monitoring Control

After each trust evaluation, the engine resets all counters to prepare for the next evaluation period. The monitoring pause flag determines whether the monitoring function

collects new evidence while computation is in progress. This is necessary to ensure a consistent snapshot of the evidence. Trust computation involves multiple steps, including cooperation probability calculation and entropy weight assignment. If the neighbor set or the collected packet counters were allowed to change during this process (e.g., evaluating cooperation probability over three neighbors but aggregating trust over five neighbours), the computation would become inconsistent.

2.3.4. Controller-Level Trust Aggregation

The SDN controller periodically collects trust reports from individual nodes to build a global view of the network trustworthiness. The controller aggregates reports from multiple sources using a weighted voting mechanism defined in ATAW-TM [19]. In this mechanism, reports from nodes with higher historical accuracy receive greater weight. The aggregated trust value of node j is computed as a weighted average of the trust values reported by the set of reporting nodes, where the weight of each reporting node is determined by its historical reliability as specified in [19].

2.3.5. Code Modifications

Code Modifications to the SDN-WISE Contiki Implementation

The integration of ATAW-TM required modifications to several source files in the sdn-wise-contiki repository [20], which implements SDN-WISE on the Contiki operating system, as summarized in Table 2.

Table 2. Code Modifications to the SDN-WISE Contiki Implementation.

File	Modification
contikimac.c	Implemented promiscuous mode in the MAC layer.
unicast.c	Implemented promiscuous mode in the Rime layer.
sky\contiki-conf.h	Modified channel check rate according to the modified clock resolution.
sky\platform-conf.h	Modified clock resolution.

Code Modifications in the SDN-WISE-Contiki Repository (Node-Side)

The integration of ATAW-TM required modifications to the source files in the sdn-wise-contiki repository, as summarized in Table 3.

Table 3. Code Modifications in the sdn-wise-contiki Repository.

File	Modification
Makefile	Included the trust module in the compilation process.
neighbor-table.c	Added the local trust value to the report payload.
neighbor-table.h	Added trust-related fields to the neighbor table structure.
node-conf.h	Added a data sequence field to the node configuration structure.
packet-buffer.h	Added an LQI field to the packet information structure.
packet-creator.c	Implemented energy consumption computation; modified the packet creation functions to include the data sequence field in data packets and the calculated remaining energy in beacon and report packets.

Table 3. *Cont.*

File	Modification
packet-handler.c	Implemented packet monitoring during packet handling.
sdn-wise.c	Added a periodic timer to invoke trust computation every 30 s.
trust-metrics.c	New source file implementing trust computation.
trust-metrics.h	Header file declaring the trust computation interfaces.

Code Modifications in the SDN-WISE-Java Repository (Controller-Side)

The integration of ATAW-TM required modifications to the source files in the sdn-wise-java repository [21], as summarized in Table 4:

Table 4. Code Modifications in the sdn-wise-java Repository.

File	Modification
ReportPacket.java	Modified the parsing process of report packet for the local trust value field.
TrustAggregation.java	New source file for trust aggregation.
NetworkGraph.java	Added updating trust aggregation with the newly received local report in the updateMap function process and printing the trust value matrix and aggregated trust values.

2.4. Attack Models

To simulate internal threat scenarios where compromised sensor nodes launch DoS attacks from within the network, we implemented 14 DoS attack models and the data tampering attack by modifying the source code of the sdn-wise-contiki repository (Table 5). The attacks were executed at designated nodes and time instants according to the strategies outlined in Section 2.5.3. The resulting attack models were executed and evaluated in the Cooja simulator.

Table 5. Implementation of Attack Models in the sdn-wise-contiki Repository.

Attack Type	Description	Implementation Method
Blackhole (data plane or control plane)	The malicious node drops all DATA packets or control packets (except BEACON packets) without forwarding them.	Modified the “handle_packet()” function to discard all received DATA packets or control packets while allowing BEACON packets to be processed.
Grey Hole	The malicious node selectively or periodically drops packets.	Modified the “handle_packet()” function to periodically discard all received packets using the trust computation timer.
Selective Forwarding	The malicious node selectively drops packets coming from specific node.	Modified the “handle_packet()” function to discard non-beacon packets whose source address matches a predefined node address.
Flooding	The malicious node injects a high volume of dummy packets.	Increased the packet transmission frequency by reducing the data sending interval from 5 s to 0.1 s.
LDoS	The malicious node injects a small volume of dummy packets.	Increased the packet transmission frequency by reducing the data sending interval from 5 s to 1 s to emulate low-rate flooding.

Table 5. *Cont.*

Attack Type	Description	Implementation Method
Sinkhole	The malicious node advertises false routing metrics to attract traffic.	Modified the "create_beacon()" function to advertise an unrealistically high routing metric.
Bad-Mouthing	The malicious node falsely reports neighboring nodes as malicious.	Modified report packets so that neighboring nodes were assigned a local trust value of 0.
Good-Mouthing	The malicious node falsely reports malicious nodes as trustworthy.	Modified report packets so that malicious nodes were assigned a local trust value of 100.
Hybrid	The malicious node simultaneously launches multiple attacks.	Combined the implementations of the LDoS and Selective Forwarding attacks.
Sybil	A malicious node assumes multiple forged identities to mislead the controller and neighboring nodes.	Modified the periodic timer task so that the attacker alternates between its legitimate address and a forged, non-existent address every 3 s. When operating under the forged identity, the attacker neither transmits sensing data nor forwards packets.
Vampire	The malicious node manipulates network routing protocols to direct packets along longer or circular paths.	Modify the flow table, changing the next-hop address to its previous-hop address, so that the data plane routing forms a loop.
DDoS	Multiple malicious nodes simultaneously launch a DoS attack.	Select multiple nodes to launch a DoS attack.
On–Off	The malicious node alternates between malicious and normal behavior.	Select a node to carry out an LDoS attack for 3 min, then stop the attack.
New-Flow	Malicious nodes repeatedly send control-plane flow REQUEST packets.	Triggered additional flow request packets by repeatedly invoking the "create_and_send_request()" function.
Data Tampering	The malicious node modifies DATA packet payload contents.	Modified the "create_data()" function to alter the payload of DATA packets before transmission.

2.5. Simulation Setup

2.5.1. Network Configuration and Traffic Model

The simulation was conducted in a square sensor field of 200 m × 200 m, where 31 sensor nodes were randomly and uniformly deployed. All nodes were assumed to be static after deployment. Each node was equipped with an omnidirectional antenna; the transmission (RX) range was set to 50 m. The interference (INT) range was configured as 100 m.

From the GraphStream1.3 window on the controller side, the network topology can be seen as shown in Figure 5. The label below each node indicates its address, which consists of three parts: the network ID, a fixed 0, and the node ID. For example, 1.0.12 is the address of Node 12, belonging to network address 1. In our topology, all nodes belong to the same network address 1. The arrows on the edges indicate the direction of the links.

Following common practice in WSN performance evaluations [28,29], we adopted a constant bit rate (CBR) approach to network traffic modelling in all subsequent experiments. While this approach does not capture real-world traffic modes such as burstiness [30], it

provides a controlled baseline for evaluating the detection performance of the proposed trust mechanism. The following three types of periodic messages were generated in the network:

(i) BEACON packets are broadcast every 3 s to maintain neighbor discovery and local topology awareness.
(ii) DATA packets are generated every 5 s to represent routine sensing and reporting traffic.
(iii) REPORT packets, carrying monitoring or trust-related information, are transmitted every 7 s.

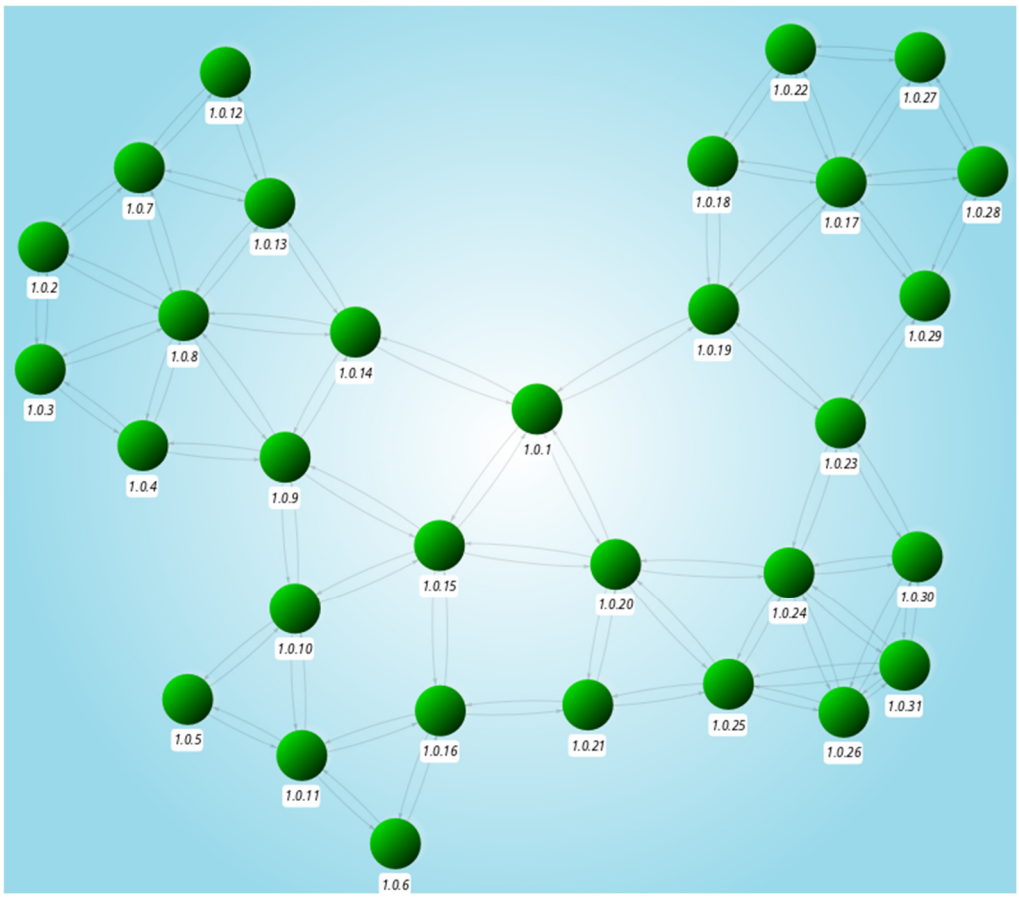


Figure 5. Network topology from the GraphStream window.

The intervals of beacon, data, and report messages were intentionally configured as non-harmonic values to prevent long-term synchronization effects and periodic packet collisions. This practice aligns with the principle of distributing transmission timings of different packet types along the time axis to suppress collision-induced packet loss, as demonstrated in prior work on periodic traffic management in WSNs [31].

2.5.2. Two Aging Strategies Comparison

We evaluated two aging initialization strategies in a no-attack scenario. The first strategy applies the ATAW-TM aging algorithm from the very first trust evaluation, starting from a state of complete distrust (initial trust value = 0). Under this strategy, trust accumulates gradually through consecutive cooperative behaviors, requiring multiple positive interactions to reach a high confidence level.

The second strategy delays the application of the aging algorithm until the second trust evaluation. In this approach, the node's initial trust value is determined by the level of cooperation observed during the first interaction. Subsequent evaluations update the trust value based on observed cooperative behavior, applying the aging mechanism. This

leads to slow trust growth during cooperation while ensuring rapid trust degradation when malicious behavior is detected.

2.5.3. Attack Configuration

For each attack type, we selected nodes strategically, depending on the attack's nature [9]: either to maximize the attack's impact or to make detection particularly challenging. The specific attack strategies and the rationale behind it are described below.

(i) High-Impact Edge-Based Attacks (Flooding, LDoS, New-Flow, DDoS): We chose the edge nodes that are farthest from the sink. This maximizes the attack's impact, as such nodes influence every node on their multi-hop paths to the sink.
(ii) Stealthy Selective Forwarding Attack: We selected nodes located on multi-hop paths and configured them to drop packets only from specific sources, not directly connected to the attacker. This makes the selective forwarding attack significantly more difficult to detect, as the attacker continues to forward packets normally for most sources, exhibiting malicious behavior only toward the targeted victims.
(iii) Sink-Proximate Attacks (Blackhole, Grey Hole, Sinkhole, Vampire): We selected the node closest to the sink. This choice affects all traffic that passes through paths involving that node, maximizing the attack's disruptive potential.
(iv) Reputation-Based Attacks (Bad-Mouthing, Good-Mouthing): For the Bad-Mouthing attacks, we selected nodes with many neighbors to act as attackers. This allowed us to observe the attack's impact on a larger number of neighbors. For the Good-Mouthing attack, we selected nodes with few neighbors as attackers: as each neighbor's trust contribution will carry a greater weight, the false trust recommendations made by the attacking node will be more effective.
(v) Topology-Manipulation Attack (Sybil): A legitimate node located either near the sink or at an intermediate position along a multi-hop routing path was selected as the attacker. The attacker periodically alternated between its legitimate identity and a forged, non-existent identity. These locations were chosen because forged identities introduced near the sink or on critical forwarding paths can exert a greater influence on topology perception, routing decisions, and trust assessment at the controller.
(vi) Temporal Attack (On–Off): We implemented the on–off attack by periodically activating and deactivating an LDoS attack and positioned the attacker at an edge node.
(vii) Data Tampering Attack: Attacking node selection was randomized, as the attack's effectiveness does not depend on network position.

The attack start time was randomized within the first 0 to 200 s of simulation. Each experimental run lasted between 10 and 30 min, depending on the attack type and evaluation requirements.

2.5.4. Repeated Experiments and Evaluation Metrics

To reduce the impact of the topology-specific characteristics, attacker placement, and random initialization, each attack scenario was evaluated using two network topologies consisting of 31 and 51 sensor nodes deployed within a 200 m × 200 m area. For each attack type, malicious nodes were randomly selected from the candidate node set according to the node-selection strategy described in Section 2.5.3. To avoid bias caused by fixed attacker locations, a different attacker placement was selected for each experimental repetition. Three independent experimental runs using different random seeds were conducted for each topology, resulting in six runs per attack scenario.

The following metrics were used to evaluate the effectiveness of attack detection.

(i) The time-to-detect (TTD): Time interval between the start of an attack and the moment it is identified.
(ii) True Positive (TP): Number of malicious nodes correctly identified as malicious.
(iii) False Positive (FP): Number of legitimate nodes incorrectly identified as malicious.
(iv) False Negative (FN): Number of malicious nodes that were not detected.

Based on these values, precision (PR), recall (RC), and false negative rate (FNR) were calculated in Equations (1)–(3):

$$PR = \frac{TP}{TP + FP} \tag{1}$$

$$RC = \frac{TP}{TP + FN} \tag{2}$$

$$FNR = \frac{FN}{TP + FN} \tag{3}$$

PR measures the precision of attack detection, as only the malicious nodes correctly identified as malicious are the ones compromised. RC measures the attack detection capability level as the malicious nodes that remain undetected continue to be compromised. FNR shows the proportion of malicious nodes that remain undetected.

2.5.5. Adaptability Test

To evaluate the adaptability of the proposed trust model to varying network traffic rates, we selected the LDoS attack as the test case. This attack type is inherently tied to traffic rates, and most existing studies require threshold adjustments tailored to different rate settings for effective detection. Moreover, LDoS is particularly challenging to detect due to its low-rate nature, which allows it to evade conventional detection mechanisms.

We designed three sets of experimental scenarios to evaluate the model:

(i) Adaptability to normal traffic variations: The normal traffic rate is varied (one packet every 5, 10, 15, 20, and 25 s), while the attack node maintains a fixed sending rate of one packet per second. This tests whether the model requires manual threshold reconfiguration when background traffic changes.
(ii) Sensitivity to attack intensity: The normal traffic is fixed at one packet every 5 s, while the attacker's rate is varied to one packet every 2 s and one packet every 3 s. This evaluates the detection performance when the attack rate is closer to the normal background traffic.
(iii) High background traffic: Both the normal nodes and the attacker transmit at a relatively high rate (one packet every 2 s and one packet per second, respectively). This scenario tests the model's performance when malicious behavior is statistically similar to legitimate high-traffic conditions.

2.5.6. Debouncing Mechanism Evaluation

As the debouncing mechanism is specifically designed for extremely low-traffic conditions, we choose LDoS as the test case for this evaluation since this attack type is directly influenced by the underlying traffic rate and poses a relevant challenge in low-rate settings. We conducted two experiments:

(i) With debouncing enabled: The trust model applies the debouncing logic, treating low-traffic periods ($\leq$10 packets per evaluation period) as fully cooperative by default.
(ii) With debouncing disabled: The same experiments were repeated after removing the debouncing code, allowing us to observe how the trust model adapts to varying network rates without the debouncing mechanism.

By comparing the evolution of aggregated trust values in each scenario, we assessed the impact of the debouncing mechanism on trust stability during network bootstrap and

low-traffic periods, the ability of the trust model to distinguish correctly between legitimate low-rate behavior (including late-starting nodes) and actual malicious activity, and the extent to which the debouncing mechanism accelerated trust convergence in the presence of asymmetric node startup times.

2.5.7. Parameter Settings

Table 6 summarizes the key parameter settings used throughout all experiments. Unless otherwise stated, these settings were kept unchanged across all attack scenarios.

Table 6. Experimental Parameter Settings.

Parameter	Value
Simulation area	200 m × 200 m
Network topologies	31 nodes, 51 nodes
Radio range	50 m
Interference range	100 m
DATA interval	5 s
BEACON interval	3 s
REPORT interval	7 s
Trust evaluation interval	30 s
Detection threshold	50
LQI threshold	219
Consistency gate	30% of mean
Debouncing threshold	10 packets
Aging parameter k	10
Aging parameter x_0	0
Number of runs	6
Random seeds	Different for each run
Supply voltage	3.0 V
CPU active-mode current	1.8 mA
Low-power-mode current	54 µA
Radio transmit current	17.4 mA
Radio receiving current	18.8 mA

2.6. Data Collection and Analysis Methods

2.6.1. Data Collection Methods

To enable comprehensive post-experiment analysis, we instrumented both sensor nodes and the SDN controller to log all relevant runtime information.

The node-side and controller-side logging included the logging of all transmitted and received packets, their timestamps, and full packet contents. The flow tables of the data plane and the next-hop address of the control plane were also recorded in node-side logging. Additionally, the controller exports the global trust value matrix and aggregated trust values for each node at every trust evaluation interval. This enables tracking the evolution of trust states over time and correlating them with network events and attack activities. Log data were collected during each simulation run and stored as text files for subsequent offline analysis.

2.6.2. Attack Detection Analysis Methods

To evaluate attack detection performance, we focused on analyzing the evolution of the aggregated trust values over time. Using Python 3.13 with the Matplotlib 3.10.0 and Pandas 3.1.0 libraries, we generated step plots that visualize how the aggregated trust value of each node changes over the course of each experiment. In these plots, the key timestamps mark attack start and end times (when applicable) and the time at which the attack was

detected. In addition, we reconstructed attack propagation paths from the node-side logs using the flow tables (data plane) and the next-hop addresses (control plane) that were logged at each node.

For each attack scenario, detection performance was summarized using TTD, PR, RC, and FNR. TTD statistics were reported as mean ± standard deviation for each attack type. Since only six repetitions were available for each attack scenario, TTD confidence intervals were not reported individually for each attack type. The overall 95% confidence interval was calculated based on all successful detections across all scenarios.

2.7. Baseline Comparison Approach and Function Variants

To evaluate the performance of the proposed ATAW-TM-enhanced SDN-WISE, we compared it to the standard SDN-Wise model as a baseline model that does not include any trust mechanisms. Additionally, we compared ATAW-TM-enhanced SDN-WISE to the baseline model running with promiscuous mode enabled. This allowed us to observe the impact of the trust value collection on the expected computational overhead. All comparative experiments were conducted on a 31-node network. Additionally, to evaluate the impact of the lightweight functions, we compared the performance of the lightweight version of ATAW-TM to the performance of the unmodified functions' version. The metrics used in this evaluation are summarized in Table 7.

Table 7. Metrics for baseline comparisons.

Metric	Definition
Hop Delay	The time from packet transmission at the source to reception at the next hop.
Forwarding Delay	The processing time from packet reception to retransmission at intermediate nodes.
ROM Usage	Program memory occupied by the firmware image.
RAM Usage	Runtime memory occupied by static and global variables.
Remaining Energy Level	Residual node energy after 30 min of operation.
Trust Computation Time	Time required for a sensor node to complete one trust-evaluation cycle for all neighbors contained in its neighbor table.

We conducted statistical analyses of the performance metrics extracted from node-side logs as follows:

(i) Single-hop Delay Analysis: We parsed the transmission and reception timestamps from node logs to calculate single-hop delays for all successfully delivered packets. The following statistics were computed for each protocol: minimum, maximum, mean, and median single-hop delays, 95th and 99th percentile delays, and delay distribution across intervals: 0–10 ms, 10–20 ms, 20–50 ms, 50–100 ms, 100–200 ms, 200–500 ms, 500–1000 ms, 1000–5000 ms, and >5000 ms.

(ii) TTL-Based Hop Delay: To understand how delay accumulates as packets traverse multiple hops toward the sink, we analyzed average hop delays stratified by TTL value.

(iii) Forwarding Delay: Forwarding delay was calculated for intermediate nodes that forwarded packets (rather than originating or consuming them). This metric reflects the processing overhead introduced by each protocol, including trust computation and trust evidence collection in the ATAW-TM-enhanced SDN-WISE. Key statistics include

the minimum, maximum, mean, and median forwarding delays and the standard deviation and percentile distributions (50th, 75th, 90th, 95th, 99th).

(iv) ROM and RAM Usage: These were measured directly from the compiled firmware images compiled in debug mode using the MSP430 size analysis tool (msp430-size). ROM usage was calculated as the sum of the .text and .data sections, while RAM usage was calculated as the sum of the .data and .bss sections.

(v) Remaining Energy Level: Node energy consumption was estimated using Contiki's Energest framework [32]. CPU, low-power mode (LPM), transmission (TX), and reception (RX) times were collected from Energest counters. The remaining node energy was then calculated using the standard energy model [33]. Parameters for the energy model are provided in Table 6. The remaining battery capacity is normalized using the nominal battery capacity (2000 mAh) and mapped to an 8-bit energy level ranging from 0 to 255, where 255 indicates a fully charged battery, and 0 indicates an exhausted battery.

(vi) Trust Computation Time: The execution time of the trust computation function was measured by inserting timing checkpoints at its entry and exit points.

3. Results

Throughout this section, trust values are reported on a scale of 0 to 100. A node is considered malicious when its aggregated trust value falls below the detection threshold of 50. The threshold was experimentally determined, as most legitimate node trust values were consistently above 50 under normal network conditions, while malicious nodes dropped below 50 within a detectable time window. This is consistent with the trust model design where a value of 50 represents the neutral midpoint at which $\alpha = \beta$ (i.e., equalized cooperative and uncooperative observations.

3.1. Aging Strategies Comparison Results

As shown in Figures 6 and 7, the second strategy, which initiates aging from the second evaluation round onward, enables faster initial trust convergence for legitimate nodes. Consequently, this strategy was employed throughout all our experiments.

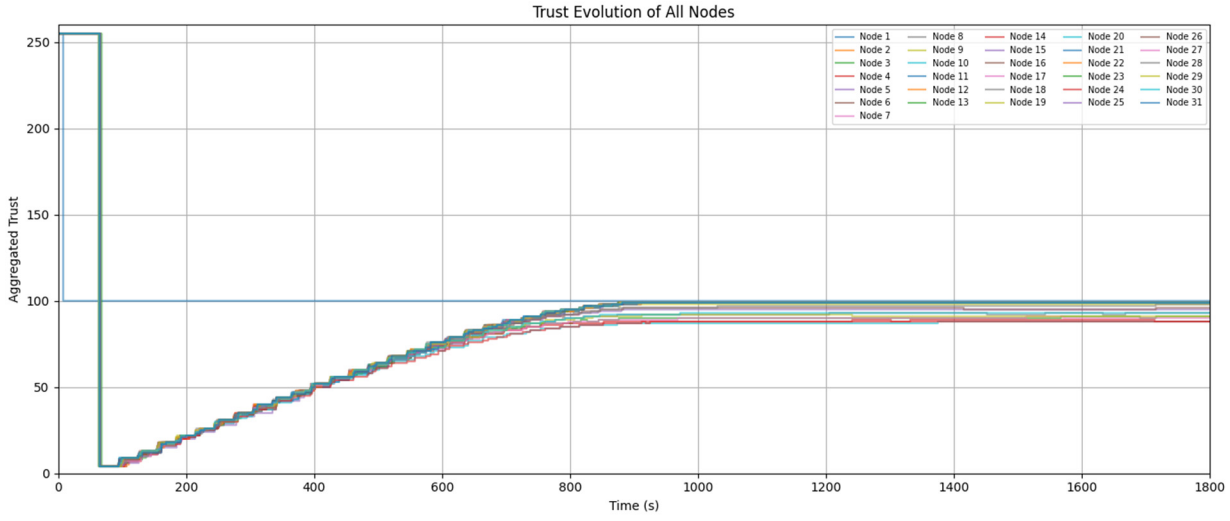


Figure 6. Evolution of aggregated trust values under aging strategy 1 (aging from first evaluation round).

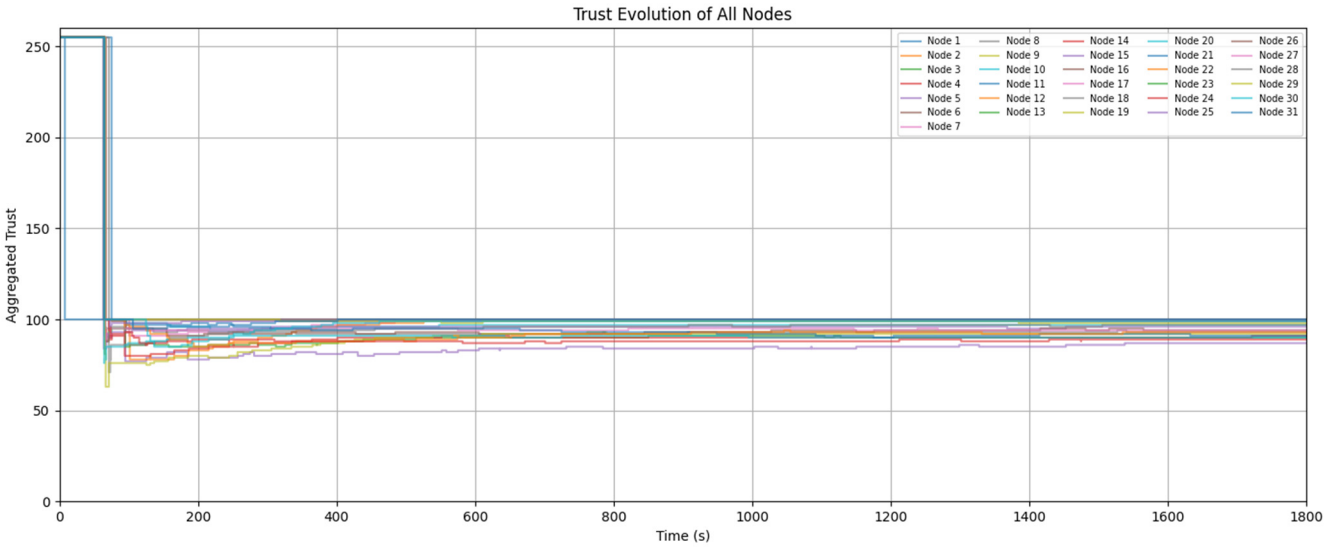


Figure 7. Evolution of aggregated trust values under aging strategy 2 (aging from second evaluation round).

3.2. Attack Detection Results

The subsections below present and interpret the experimental results illustrating node behaviors under the different attack scenarios. For each scenario, we analyze the evolution of trust values over time and reconstruct the attack path.

3.2.1. Blackhole Attack

Blackhole Attack Targeting the Control Plane

In this attack scenario, Node 14 performs a blackhole attack on the control plane, discarding all received control packets with the exception of BEACON packets. Figure 8 shows the attack path, in which Node 14 drops all control packets except BEACON packets sent from Nodes 2, 3, 4, 7, 8, 9, 12, and 13 (the red dashed arrow) but still sends its own control packets (the yellow solid arrow). In addition, Node 14 discards all control packets originating from the sink node, thereby blocking both itself and its neighboring nodes from receiving flow table updates from the controller. The flow tables of the data plane also show that there are no entries for Nodes 2, 3, 4, 7, 8, 9, 12, 13, and 14 (Figure 9), in which each row represents a flow table entry (data plane). For example, the first row indicates that the next hop from Node 27 to Node 1 (the sink node) is Node 28.

Figure A1 shows that the aggregated trust of some nodes remains at an invalid value (255); these nodes can be identified from the entries in Figure 10. In the trust value matrix of this figure, the row index e represents the evaluating node, while the column index t represents the evaluated node. The valid trust value range is from 0 to 100. The initial trust value is set to 255 as an invalid marker; this value is displayed as "." in the output. We can see that the trust values of Nodes 2, 3, 4, 7, 12, and 14 have always remained at an invalid value. Since Node 14 still sends data and control packets, including trust reports for Nodes 8, 9, and 13, the trust values of these nodes are updated.

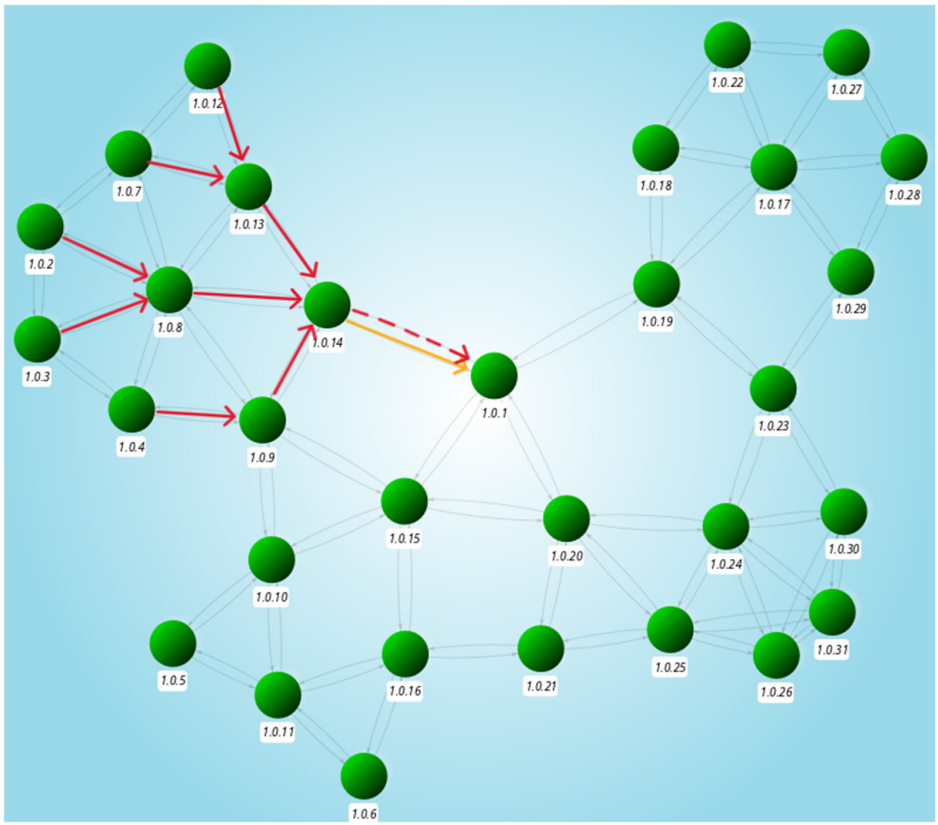


Figure 8. The attack path of a blackhole attack by Node 14 targeting the control plane.

```
10:24.557  ID:27  [FLT]: IF (P.2 = 1;){FORWARD_U  0 28;}[0 116]
10:24.564  ID:11  [FLT]: IF (P.2 = 1;){FORWARD_U  0 10;}[0 112]
10:24.679  ID:16  [FLT]: IF (P.2 = 1;){FORWARD_U  0 15;}[0 196]
10:24.691  ID:15  [FLT]: IF (P.2 = 1;){FORWARD_U  0 1;}[0 478]
10:24.747  ID:30  [FLT]: IF (P.2 = 1;){FORWARD_U  0 31;}[0 118]
10:24.815  ID:31  [FLT]: IF (P.2 = 1;){FORWARD_U  0 24;}[0 196]
10:24.844  ID:22  [FLT]: IF (P.2 = 1;){FORWARD_U  0 17;}[0 116]
10:24.945  ID:10  [FLT]: IF (P.2 = 1;){FORWARD_U  0 15;}[0 265]
10:25.073  ID:19  [FLT]: IF (P.2 = 1;){FORWARD_U  0 1;}[0 642]
10:25.141  ID:20  [FLT]: IF (P.2 = 1;){FORWARD_U  0 1;}[0 605]
10:25.201  ID:21  [FLT]: IF (P.2 = 1;){FORWARD_U  0 25;}[0 120]
10:25.247  ID:26  [FLT]: IF (P.2 = 1;){FORWARD_U  0 25;}[0 119]
10:25.563  ID:24  [FLT]: IF (P.2 = 1;){FORWARD_U  0 20;}[0 258]
10:25.630  ID:23  [FLT]: IF (P.2 = 1;){FORWARD_U  0 19;}[0 114]
10:25.668  ID:25  [FLT]: IF (P.2 = 1;){FORWARD_U  0 20;}[0 308]
10:26.184  ID:17  [FLT]: IF (P.2 = 1;){FORWARD_U  0 19;}[0 416]
10:26.501  ID:6   [FLT]: IF (P.2 = 1;){FORWARD_U  0 16;}[0 111]
10:27.027  ID:28  [FLT]: IF (P.2 = 1;){FORWARD_U  0 17;}[0 208]
10:27.330  ID:18  [FLT]: IF (P.2 = 1;){FORWARD_U  0 19;}[0 119]
10:27.416  ID:29  [FLT]: IF (P.2 = 1;){FORWARD_U  0 17;}[0 113]
10:27.428  ID:5   [FLT]: IF (P.2 = 1;){FORWARD_U  0 10;}[0 109]
```

Figure 9. The node-side output of data plane flow tables under a blackhole attack by Node 14 targeting the control plane.

```
==== Trust Matrix ====
  e\t   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
  1     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  2     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  3     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  4     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  5     .   .   .   .   .   .   .   .   . 83 100   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  6     .   .   .   .   .   .   .   .   .   . 100   .   .   .   .  87   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  7     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  8     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  9     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  10    .   .   .   . 100   .   .   . 100   . 100   .   .   . 95   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  11    .   .   .   . 100 100   .   .   . 95   .   .   .   .   . 87   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  12    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  13    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  14    .   .   .   .   .   .   . 100 100   .   .   . 100   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  15    .   .   .   .   .   .   .   . 100 100   .   .   .   .   . 100   .   .   .   .  93   .   .   .   .   .   .   .   .   .   .
  16    .   .   .   .   . 100   .   .   .   . 100   .   .   . 86   .   .   .   .   .   . 100   .   .   .   .   .   .   .   .   .
  17    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 100  92   .   . 100   .   .   .   . 100  94  99   .   .
  18    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .  97   . 93   .   . 100   .   .   .   .   .   .   .   .
  19    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 100  96   .   .   .   . 100   .   .   .   .   .   .   .
  20    .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 100   .   .   .   .   . 100   .   . 100  90   .   .   .   .   .
  21    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 100   .   .   . 95   .   .   .   . 92   .   .   .   .   .
  22    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 83  99   .   .   .   .   .   .   . 100   .   .   .   .
  23    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 77   .   .   .   . 100   .   .   .  99 100   .
  24    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 96   .   . 100   . 96 100   .   .   . 100  88
  25    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 98 100   .   . 95   . 100   .   .   .   . 100
  26    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 80  95   .   .   .   . 100  95
  27    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 86   .   .   .   . 100   .   .   .   .   . 90   .   .   .
  28    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 90   .   .   .   .   .   .   .   .   . 100   . 99   .   .
  29    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 87   .   .   .   .   . 100   .   .   .   . 100   .   .   .
  30    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 100  79   . 100   .   .   .   . 90
  31    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 55  99 100   .   .   . 100   .
==== Aggregated Trusts ====
    id   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
trust 100   .   .   . 100 100   . 100 100  93 100   . 100   .  94  93  90  98  87  95 100 100 100  85  94 100 100  95  99 100  93
============================
```

Figure 10. The controller-side output of the trust value matrix and aggregated trust values under a blackhole attack by Node 14 targeting the control plane.

Blackhole Attack Targeting the Data Plane

In this attack scenario, Node 14 mounts a blackhole attack on the data plane, discarding all received data packets. Figure 11 shows the attack path, in which Node 14 drops all data packets sent from Nodes 2, 3, 4, 7, 8, 9, 12, and 13, as indicated by the red dashed arrow. Figure A2 shows that the aggregated trust value of Node 14 drops to under 50 at 65.1 s.

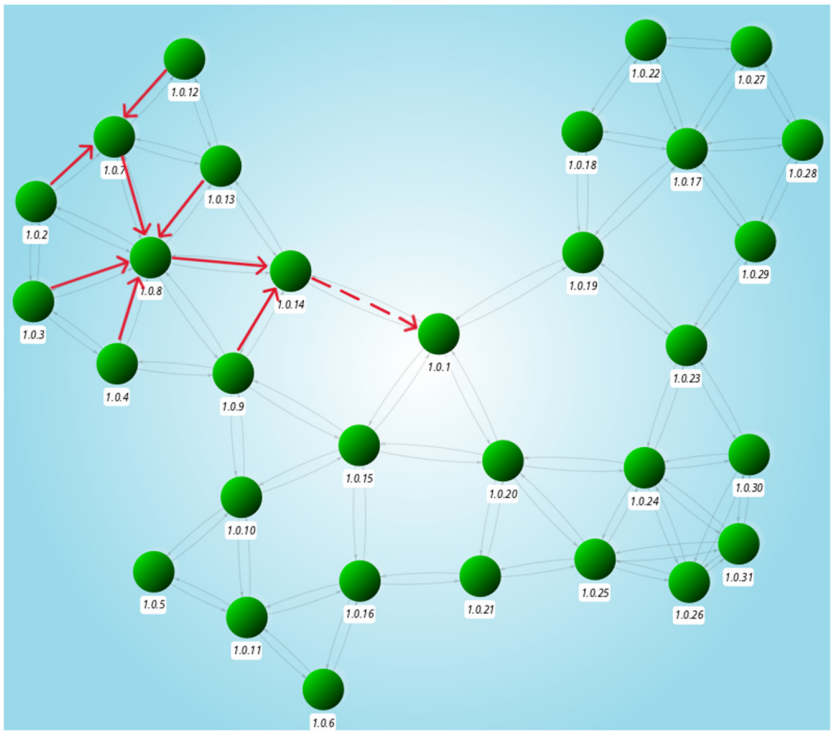


Figure 11. The attack path of a blackhole attack by Node 14 targeting the data plane.

3.2.2. Grey Hole Attack

In this attack scenario, Node 14 drops all received packets for 30 s, then resumes normal operation for 30 s, alternating throughout the simulation. Figure 12 shows the attack path, in which Node 14 drops all packets from Nodes 2, 3, 4, 7, 8, 9, 12, and 13 periodically. The attack paths targeting the control plane and the data plane are indicated by the red and yellow dashed arrows, respectively. Figure A3 shows that the aggregated trust value of Node 14 drops to under 50 at 64.6 s. When the trust value of node 14 stabilizes, it remains relatively low compared to that of legitimate nodes, and it no longer falls below the detection threshold.

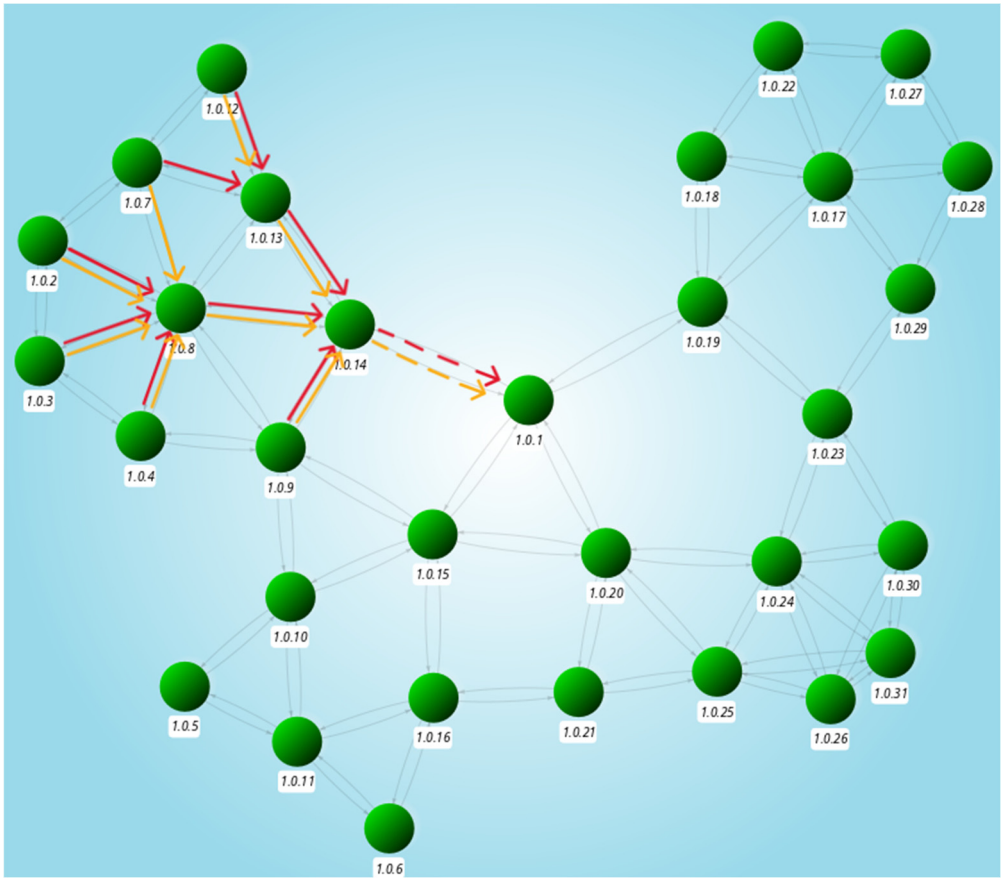


Figure 12. The attack path of a grey hole attack by Node 14.

3.2.3. Selective Forwarding Attack

In this attack scenario, Node 14 mounts a selective forwarding attack by dropping all packets coming from the specific Node 2. Figure 13 shows the attack path, in which Node 14 selectively drops only the packets sent by Node 2 as indicated by the red dashed arrow; it keeps forwarding packets sent by other nodes (see the yellow solid arrows). Figure A4 shows that the aggregated trust value of node 14 drops to under 50 at 335.5 s.

3.2.4. Flooding Attack

In this attack scenario, Node 12 initiates a flooding attack on the data plane by sending 10 DATA packets per second. Figure 14 shows the attack path, which is from Node 12 to Node 13 to Node 8 to Node 14, and finally to Node 1 (see the solid red arrows). Figure A5 shows that the aggregated trust value of Node 12 drops to under 50 at 78.0 s. However, the trust value of the victim nodes also declines; the trust value of victim Node 8 even drops to below 50.

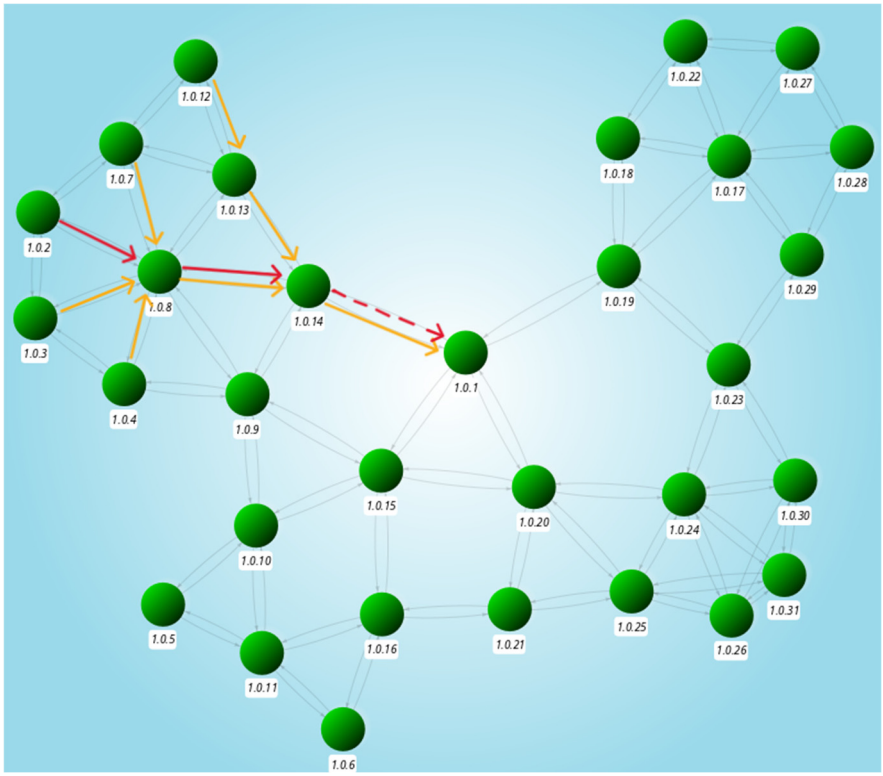


Figure 13. The attack path of a selective forwarding attack by Node 14.

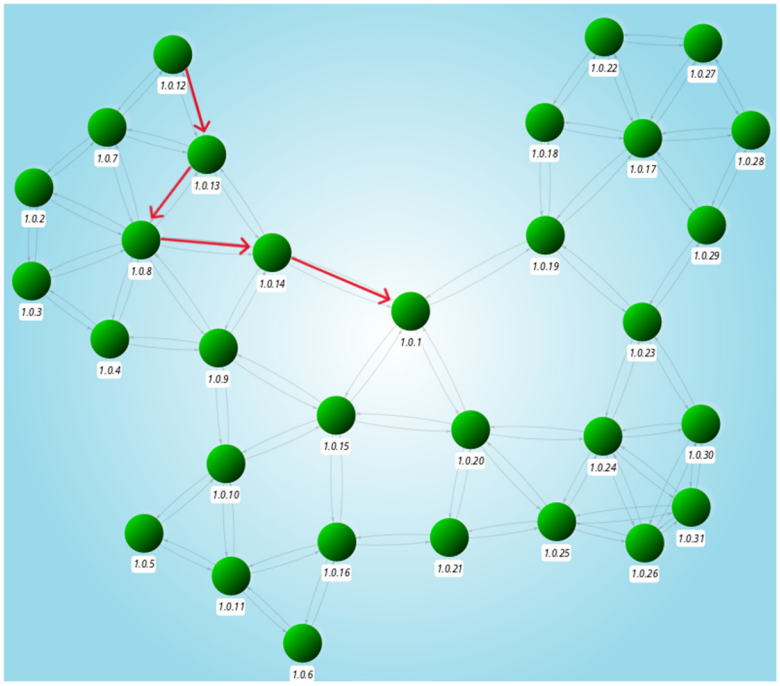


Figure 14. The attack path of a flooding attack by Node 12.

3.2.5. LDoS Attack

In this attack scenario, Node 12 mounts an LDoS attack on the data plane by sending one DATA packet per second. The attack path is the same as the flooding attack (Figure 14). Figure A6 shows that the aggregated trust value of Node 12 drops to under 50 at 73.5 s. The trust value of the victim nodes also declines; the trust value of victim Node 14 drops even further, to below 50.

3.2.6. Sinkhole Attack

In this attack scenario, Node 15 performs a sinkhole attack, advertising false routing metrics to attract traffic. Figure 15 shows the attack path, in which Node 15 attracts more than half of the control plane traffic to itself (see the red solid arrows). Figure A7 shows that the aggregated trust value of Node 15 drops to under 50 at 141.1 s. When the trust value of Node 15 stabilizes, it remains relatively low compared to that of legitimate nodes, but it may no longer be flagged as malicious.

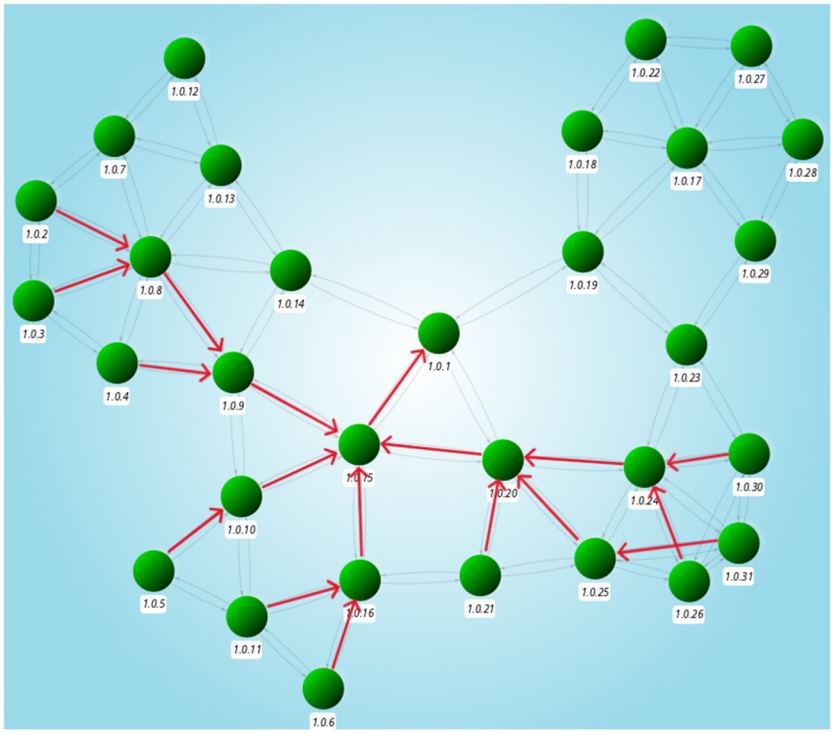


Figure 15. The attack path of a sinkhole attack by Node 15.

3.2.7. Bad-Mouthing Attack

In this attack scenario, Node 8 mounts a bad-mouthing attack, falsely reporting neighboring nodes as malicious. Figure 16 shows the controller-side output of the trust value matrix and aggregated trust values under this attack. While Node 8 reported a trust value of zero for each of its neighbors (red box), the trust values of these neighboring nodes (yellow boxes) did not decline appreciably in response. Figure A8 also shows that this attack has little impact on the trust values of the targeted nodes.

3.2.8. Good-Mouthing

In this attack scenario, Node 14 carries out a selective forwarding attack by dropping packets from Node 2, as shown in Figure 13. At the same time, the trust value of Node 14 as reported by Node 8 is 100, attempting to conceal the malicious behavior of Node 14. Figure A9 shows that the aggregated trust value of Node 14 drops to under 50 at 499.2 s. While this point is 163.7 s later than the detection time of 335.5 s for a pure selective forwarding attack (as described in Section 3.2.3), the attack is eventually detected.

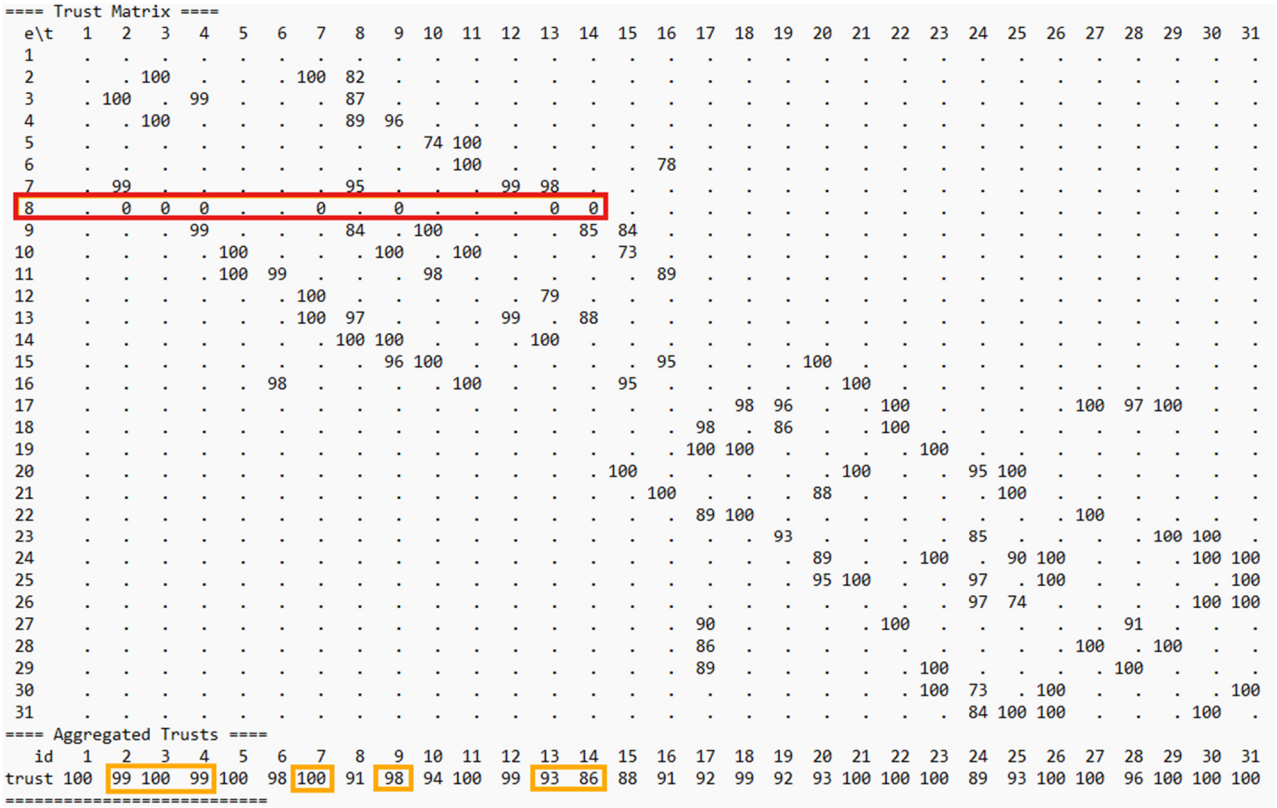


Figure 16. The controller-side output of the trust value matrix and aggregated trust values under a bad-mouthing attack by Node 8.

3.2.9. Hybrid Attack

In this attack scenario, Node 8 launches a hybrid attack by launching a selective forwarding attack and an LDoS attack simultaneously. The attack path is shown in Figure 17. Node 8 selectively drops packets sent by Node 2 (see the red dashed arrow) while still forwarding packets sent by other nodes (see the yellow solid arrows). Simultaneously, Node 8 launches an LDoS attack, sending one DATA packet per second (see the purple solid arrow). Figure A10 shows that the aggregated trust value of Node 8 drops to under 50 at 372.7 s. The trust value of the victim, Node 14 also drops below 50.

3.2.10. Sybil Attack

In this attack scenario, Node 8 performs a sybil attack by periodically alternating between its legitimate identity and a forged, non-existent identity every 3 s. While operating under the forged identity, the attacker neither generates sensing data nor forwards packets. Figure 18 shows the topology observed at the controller, where the forged identity appears as an additional network entity despite being associated with the same physical node (see the red circle). Figure A11 shows that the trust value associated with the forged identity rapidly declines because the identity does not exhibit normal sensing or forwarding behavior. Consequently, the forged identity is identified as malicious at 121.6 s.

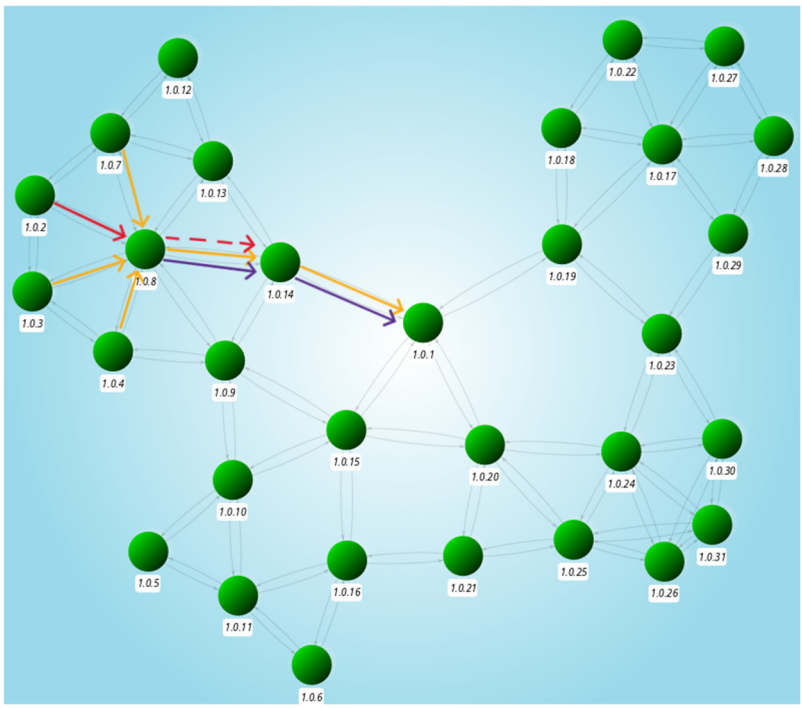


Figure 17. The attack path of a hybrid attack by Node 8.

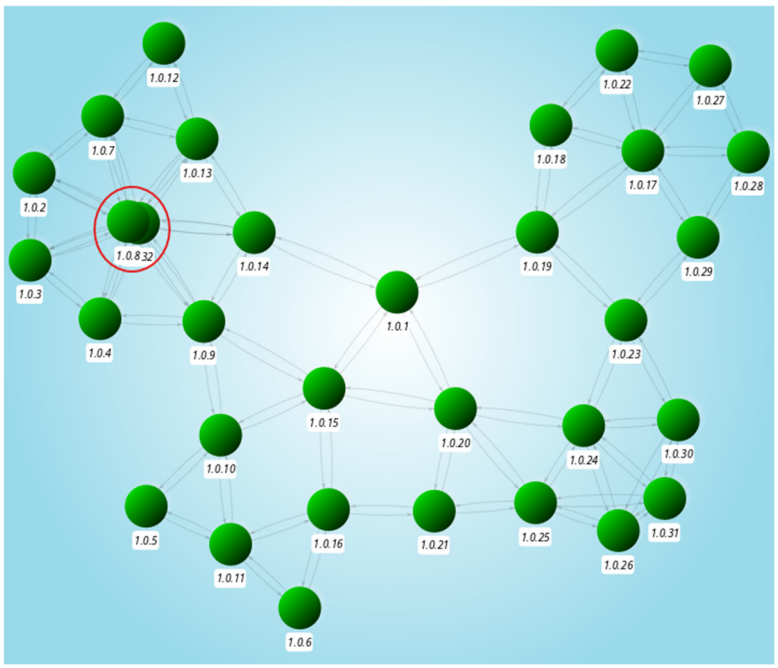


Figure 18. Controller-side topology view under a sybil attack showing the appearance of a forged identity (Node 32).

3.2.11. Vampire Attack

In this attack scenario, Node 19 launches a vampire attack, directing packets along a circular path. The attack path is shown in Figure 19. Node 19 forwards back to its previous hop, Node 17 (the red solid arrow), all transmitting packets that would otherwise go to the sink (the yellow dashed arrow). This creates a loop between Node 19 and Node 17. Figure A12 shows that the aggregated trust value of Node 19 drops to under 50 at 369.6 s. However, the trust value of the victim, Node 17 also drops below 50.

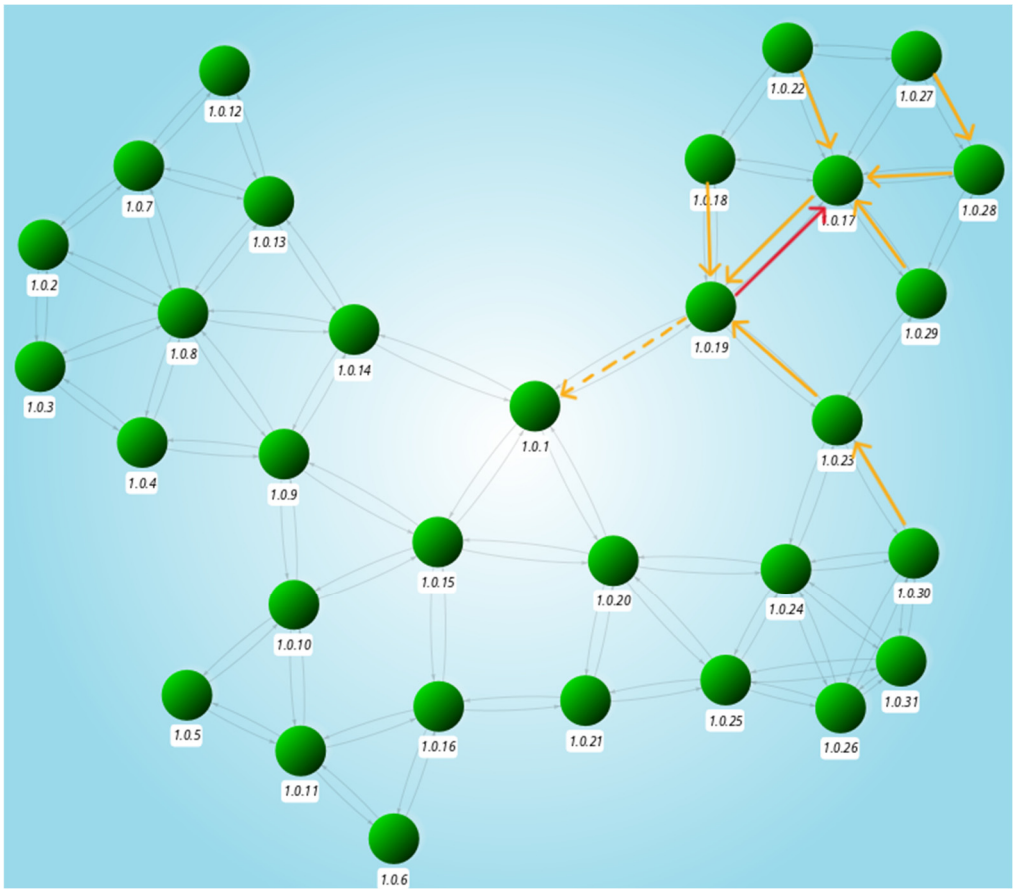


Figure 19. The attack path of a vampire attack by Node 19.

3.2.12. DDoS Attack

In this attack scenario, Nodes 12, 27, and 31 mount a coordinated DDoS attack by simultaneously launching LDoS attacks. Figure 20 shows the attack path from these attackers to the sink (see the red solid arrows). As shown in Figure A13, the aggregated trust values of Nodes 12, 27, and 31 drop below 50 at 117.7 s. The aggregated trust value of the victim nodes also declined, dropping below 50; these are identified in the controller-side trust value matrix shown in Figure 21. Red boxes show trust values of attacker, while yellow boxes show trust values of victim nodes.

3.2.13. On–Off Attack

Our grey hole attack test in Section 3.2.2 follows a similar on–off pattern, with the attacker alternating between 30 s of attack and 30 s of inactivity. We have verified the effectiveness of our approach against this type of attack. In this test, we focus on two key metrics: the time taken for the attacker's trust value to drop from its peak to its nadir during the attack and the time taken for it to recover from the nadir to the peak after the attack ceases. These metrics allow us to assess the cost of the attack in terms of trust degradation and recovery. In this attack scenario, Node 12 initiates an LDoS attack on the data plane by sending one DATA packet per second for 3 min and then stops the attack. Figure 22 shows the attack path, which is from Node 12 to Node 13 to Node 14, and finally to the sink (see the red solid arrows). Once the attack is stopped, the time it takes for the aggregated trust value of Node 12 to rise from the lowest to near the highest point is five times longer than the time it takes for the aggregated trust value to drop from the highest point during the attack (Figure A14).

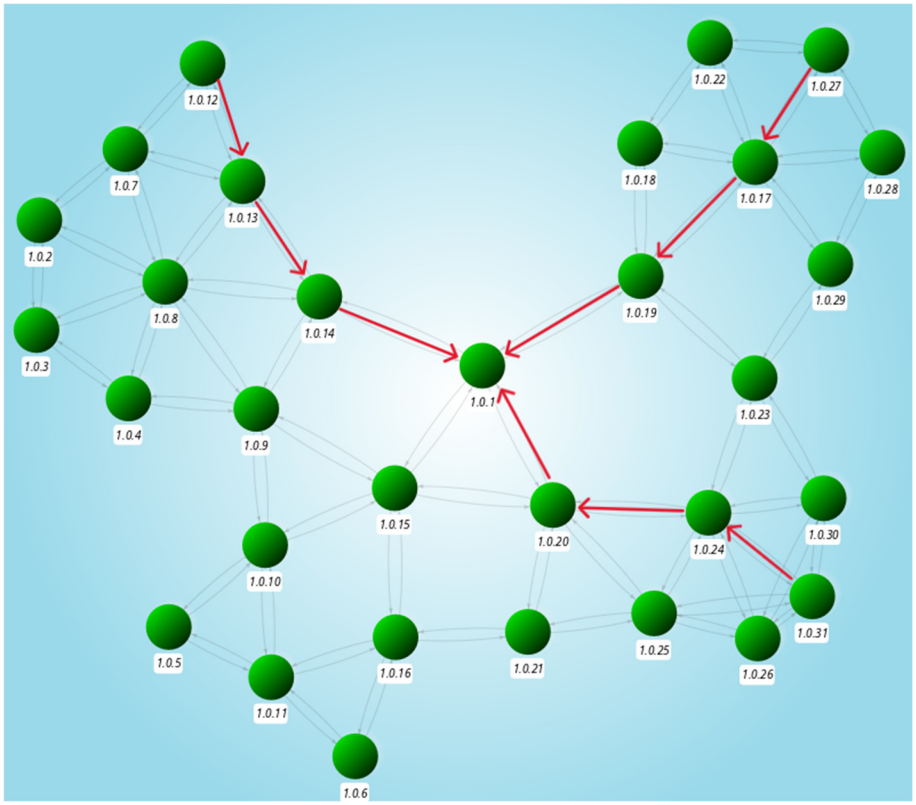


Figure 20. The attack path of a DDoS attack by Nodes 12, 27, and 31.

```
==== Trust Matrix ====
  e\t   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
  1     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  2     .   . 100   .   .   . 100  76   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  3     .  74   . 100   .   .   .  93   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  4     .   . 100   .   .   .   .  96  93   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  5     .   .   .   .   .   .   .   .   . 85 100   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  6     .   .   .   .   .   .   .   .   .   . 100   .   .   .   . 82   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  7     .  99   .   .   .   .   .  97   .   .   .  29  25   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  8     .  98  99  99   .   .  99   .  99   .   .   .  99  46   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  9     .   .   . 100   .   .   . 100   . 100   .   .   .  45  96   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
 10     .   .   .   . 100   .   .   . 100   . 100   .   .   .  87   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
 11     .   .   .   . 100 100   .   .   .  95   .   .   .   .   . 90   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
 12     .   .   .   .   .   . 100   .   .   .   .   . 21   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
 13     .   .   .   .   .   .   . 100  96   .   .   .  25   .  37   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
 14     .   .   .   .   .   .   .   . 100 100   .   .   . 100   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
 15     .   .   .   .   .   .   .   .   . 100 100   .   .   .   .   . 100   .   .   .   . 100   .   .   .   .   .   .   .   .   .
 16     .   .   .   .   . 100   .   .   .   .   . 100   .   .   .  93   .   .   .   .   .   . 100   .   .   .   .   .   .   .   .
 17     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 100  20   .   . 100   .   .   .   . 16 100 100   .   .
 18     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .  94   .  15   .   . 100   .   .   .   .   .   .   .   .
 19     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 100 100   .   .   .   . 100   .   .   .   .   .   .   .
 20     .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 100   .   .   .   .   . 100   .   . 100 100   .   .   .   .   .
 21     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 100   .   .   . 62   .   .   .   . 100   .   .   .   .   .
 22     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 15 100   .   .   .   .   .   .   .  33   .   .   .   .
 23     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 83   .   .   .   . 78   .   .   .   . 100  99   .
 24     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .  35   .   . 99   .  79  99   .   .   . 99  17
 25     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 31  99   .   . 10   . 99   .   .   .   . 20
 26     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 10  96   .   .   .   . 99  25
 27     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 15   .   .   .   . 100   .   .   .   .   . 100   .   .   .
 28     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 15   .   .   .   .   .   .   .   .  33   .  99   .   .
 29     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 77   .   .   .   .   . 100   .   .   .   . 100   .   .   .
 30     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 100  15   . 100   .   .   .   . 25
 31     .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   . 10  87  99   .   .   . 99   .
==== Aggregated Trusts ====
   id   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
trust 100  91 100 100 100 100 100  94  98  95 100  27  61  43  94  93  52 100  35  55 100 100 100  30  93  99  28 100 100  99  22
============================
```

Figure 21. The controller-side output of the trust value matrix and aggregated trust values under a DDoS attack by Node 12, 27, and 31.

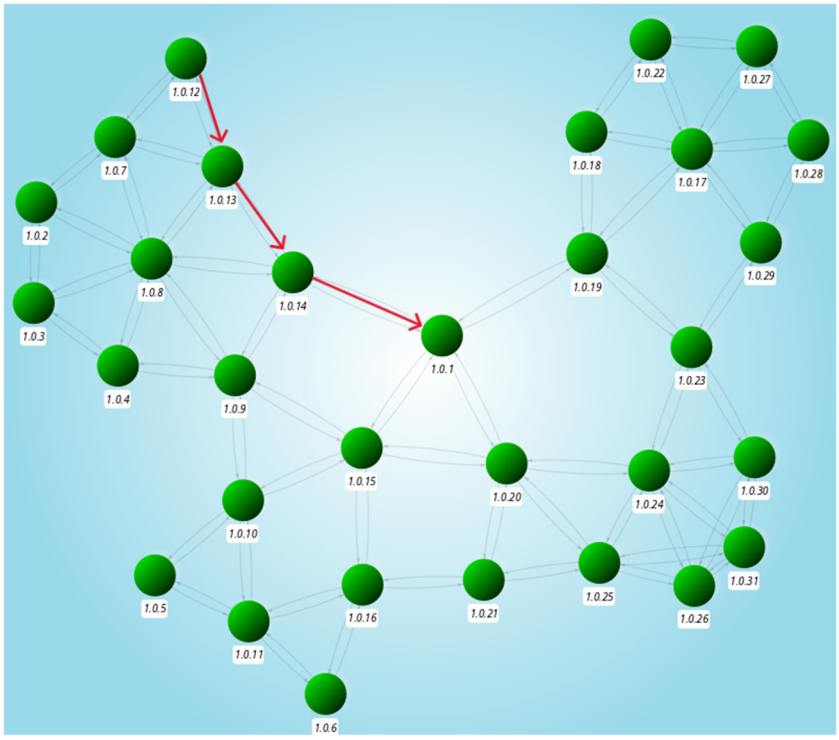


Figure 22. The attack path of an on–off attack by Node 12.

3.2.14. New-Flow Attack

In this attack scenario, Node 12 mounts a new-flow attack, sending two REQUEST packets every 3 s. The attack path is the same as the attack path of the on–off attack (Figure 22), which is from Node 12 to Node 13 to Node 14, and finally to the sink. Figure A15 shows that the aggregated trust value of Node 12 drops to under 50 at 76.1 s. The trust value of the victim nodes also declined, with the trust value of victim Node 14 even dropping to below 50.

3.2.15. Data Tampering Attack

In this attack scenario, Node 8 mounts a data tampering attack, altering the payload of DATA packets before transmission. As shown in Figure A16, the aggregated trust value of the attacker (Node 8) drops to under 50 at 281.4 s.

3.3. Statistical Summary of Detection Performance

To provide a statistically rigorous assessment of attack-detection performance, each attack scenario was evaluated in six independent runs using different random seeds, network topologies, and attacker locations. Table 8 summarizes the resulting detection statistics.

To ensure consistency in interpreting the results, we distinguish among three attack-handling outcomes. Detection refers to the explicit identification of an attack through the degradation of the malicious node's trust value below the predefined detection threshold. Identification of abnormality refers to situations in which abnormal behavior is inferred from persistent anomalies, such as the prolonged absence of trust-value updates. Mitigation refers to situations in which the attack is effectively neutralized by the trust-processing and trust-aggregation mechanisms, preventing it from achieving its intended objective.

Accordingly, the handling of control-plane blackhole attacks is classified as identification of abnormality. In this attack, the malicious node prevents trust reports from reaching the controller. Rather than a measurable decline in trust values, there is a prolonged absence of trust-value updates. Consequently, a specific TTD cannot be determined for this scenario.

Table 8. Statistical summary of attack detection, abnormality identification, and mitigation performance.

Attack Type		TTD Mean ± SD (s)	TP	FP	FN	PR	RC	FNR	Detection Type
Blackhole	control plane	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Identification of Abnormality
	data plane	96.5 ± 83.8	6	0	0	1	1	0	Detection
Grey Hole		32.6 ± 3.1	6	0	0	1	1	0	Detection
Selective Forwarding		198.3 ± 129.5	5	0	1	1	0.833	0.167	Detection
Flooding		57.4 ± 21.1	6	2	0	0.75	1	0	Detection
LDoS		46.1 ± 4.6	6	2	0	0.75	1	0	Detection
Sinkhole		100.4 ± 31.8	5	0	1	1	0.833	0.167	Detection
Bad-Mouthing		N/A	N/A	N/A	N/A	N/A	N/A	N/A	Mitigation
Good-Mouthing		272.7 ± 193.8	5	0	1	1	0.833	0.167	Mitigation
Hybrid		88.1 ± 91.3	6	3	0	0.667	1	0	Detection
Sybil		107.8 ± 13.3	5	0	1	1	0.833	0.167	Detection
Vampire		222.4 ± 113.3	4	3	2	0.571	0.667	0.333	Detection
DDoS		66.0 ± 15.0	18	5	0	0.783	1	0	Detection
On–Off		123.3 ± 50.7	6	0	0	1	1	0	Detection
New-Flow		55.3 ± 21.9	6	1	0	0.857	1	0	Detection
Data Tampering		226.3 ± 81.0	6	0	0	1	1	0	Detection

Furthermore, the handling of the bad-mouthing attack is classified as mitigation because the false trust reports are effectively neutralized by the trust-aggregation mechanism without the need for explicit attack detection. As a result, no TTD is reported for this attack. The handling of the good-mouthing attack is also classified as mitigation. Although the attacker attempts to conceal the malicious behavior of another compromised node by assigning it an artificially high trust value of 100, the trust-aggregation mechanism prevents the forged recommendation from significantly influencing the global trust assessment. Consequently, the promoted malicious node is still identified as malicious by the proposed trust mechanism. Therefore, the attack fails to achieve its objective and is considered mitigated rather than detected.

Across all successful detections ($N = 78$), the overall mean TTD was 115.8 s (95% CI: 92.7–138.9 s).

3.4. Adaptability Test Results

As shown in Table 9, we evaluated the detection performance under various normal and attack traffic rates. When the attacker maintains a constant rate of 1 packet/s, the TTD remains relatively stable across different normal traffic conditions (ranging from 32.8 s to 51.5 s as the normal rate decreases from 1/10 s to 1/25 s). This demonstrates that the model adapts to varying normal traffic rates without requiring manual threshold reconfiguration.

However, two boundary conditions warrant further discussion. First, when the attacker's rate is extremely low (1 packet per 3 s), the attack is not detected (TTD = ∞). This is because the attacker generates approximately 10 packets within the 30 s trust evaluation period. Under the debouncing mechanism (Section 2.3.3), nodes transmitting $\leq$10 packets in an evaluation window are not penalized. Consequently, the attacker's low-rate behavior is ignored, and the attacker node is not flagged as malicious.

Table 9. LDoS Attack Detection Performance under Different Normal Traffic Rates.

Normal DATA Rate	Attacker DATA Rate	TTD
1/5 s	1/3 s	∞
1/5 s	1/2 s	39.5 s
1/2 s	1/s	428.4 s
1/5 s	1/s	51.5 s
1/10 s	1/s	32.8 s
1/15 s	1/s	38.8 s
1/20 s	1/s	42.5 s
1/25 s	1/s	41.6 s

Second, when the normal traffic rate increases to 1/2 s (closely matching the attacker's 1/s rate), the TTD is significantly prolonged (428.4 s). This delay is caused by the physical limitation inherent to wireless environments. Due to interference and inherent packet loss on the wireless links, the passive monitoring counters are not 100% accurate. When the normal and malicious traffic rates are highly similar, the system requires a longer observation period (multiple evaluation cycles) to accumulate sufficient negative evidence to overcome statistical noise and distinguish the malicious node from legitimate ones.

3.5. Debouncing Mechanism Evaluation Results

Figure 23 shows that the debouncing mechanism effectively prevents false trust reports during network initialization and maintains trust value stability under legitimate low-traffic conditions. Figure 24 shows the attack (an LDoS attack) path of this experiment.

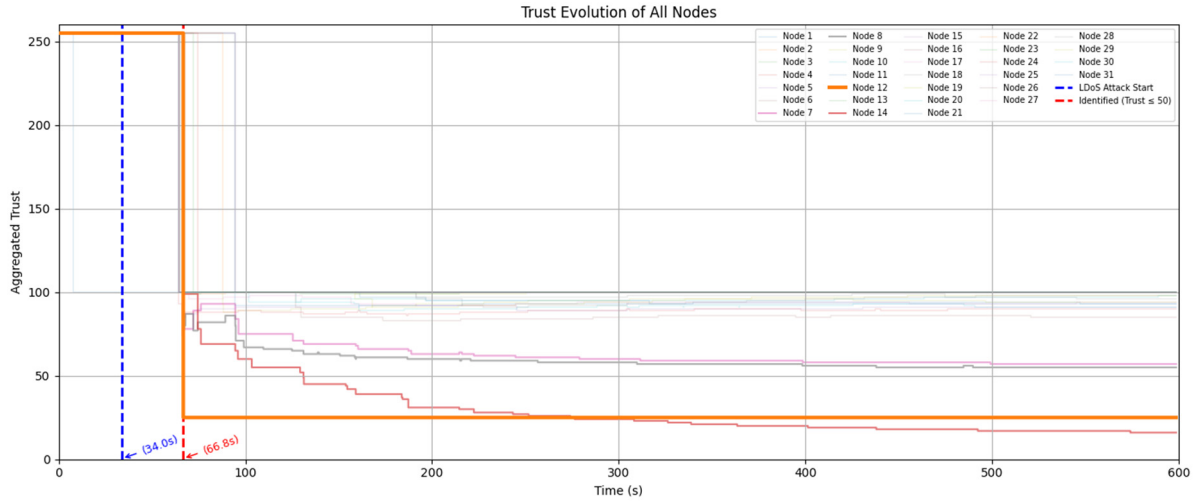


Figure 23. The evolution of the aggregated trust of all nodes with the debouncing mechanism under an LDoS attack by Node 12 in a traffic condition in which normal nodes send one DATA packet per 10 s, while the attacker sends one DATA packet per second.

In contrast, when debouncing is disabled, trust values exhibit significant fluctuation and slower convergence. This leads to legitimate nodes reaching a trust value above 50 relatively late, at the 400 s time point. Only then can attacks be truly identified, as shown in Figure 25. The attack path of this experiment is the same as in Figure 22. This demonstrates the necessity of our approach for reliable trust evaluation in practical deployments where nodes start asynchronously.

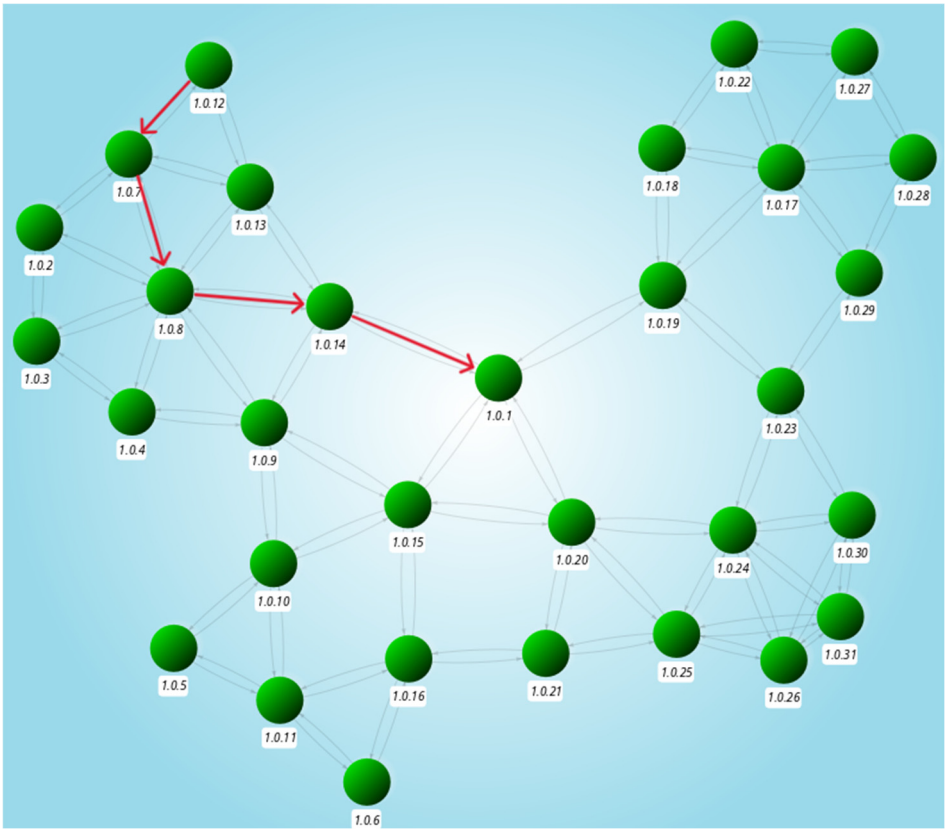


Figure 24. The attack path of an LDoS attack by Node 12 in a traffic condition in which normal nodes send one DATA packet/10 s, while the attacker sends one DATA packet per second.

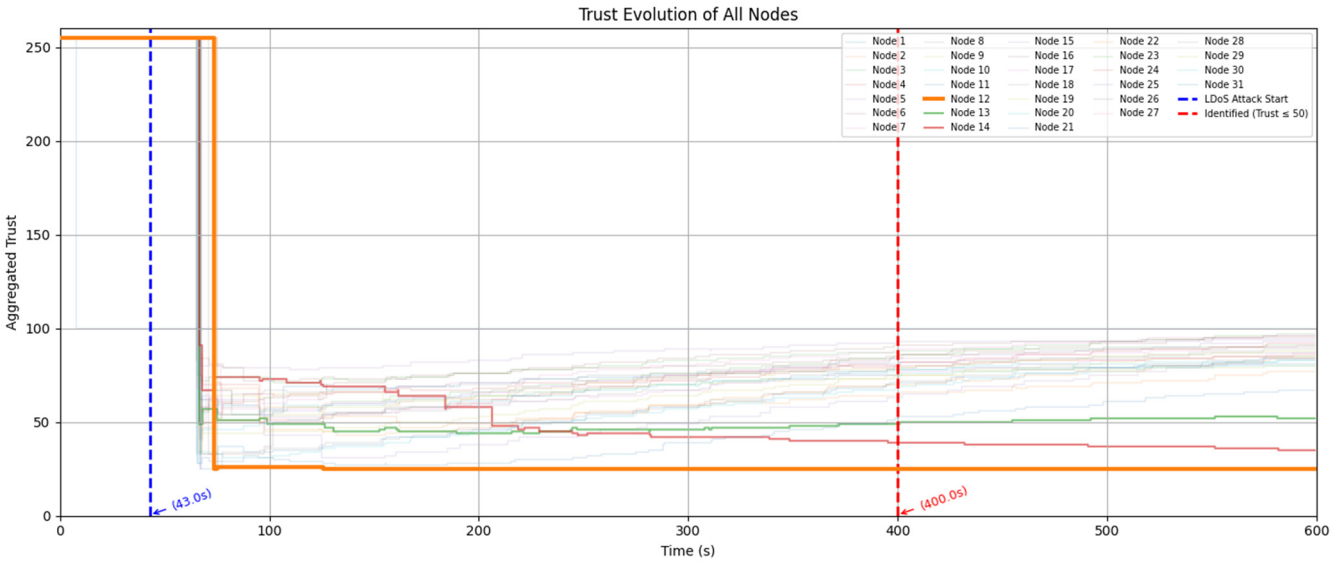


Figure 25. The evolution of the aggregated trust of all nodes without the debouncing mechanism under an LDoS attack by Node 12 in a traffic condition, where normal nodes send one DATA packet per 10 s, while the attacker sends one DATA packet per second.

3.6. Baseline Comparison and Function Variants' Results

3.6.1. Delay Comparison

As shown in Figure 26a, the average hop delay for ATAW-TM-enhanced SDN-WISE is 331.60 ms, which is 2.4% higher than the average hop delay for SDN-WISE with promiscuous mode (323.55 ms) and 5.3% higher than the average hop delay for the standard SDN-

WISE (314.99 ms). Similarly, the average forwarding delay for ATAW-TM-enhanced SDN-WISE is 25.97 ms, which is 8.9% higher compared to SDN-WISE with promiscuous mode (23.83 ms) and 8.1% higher compared to standard SDN-WISE (24.02 ms) (Figure 26b). These slight increases are expected, as the incorporation of trust evidence collection and trust computation into the SDN-WISE introduces additional processing. Nevertheless, the relative increases remain below 9% for both hop delay and forwarding delay.

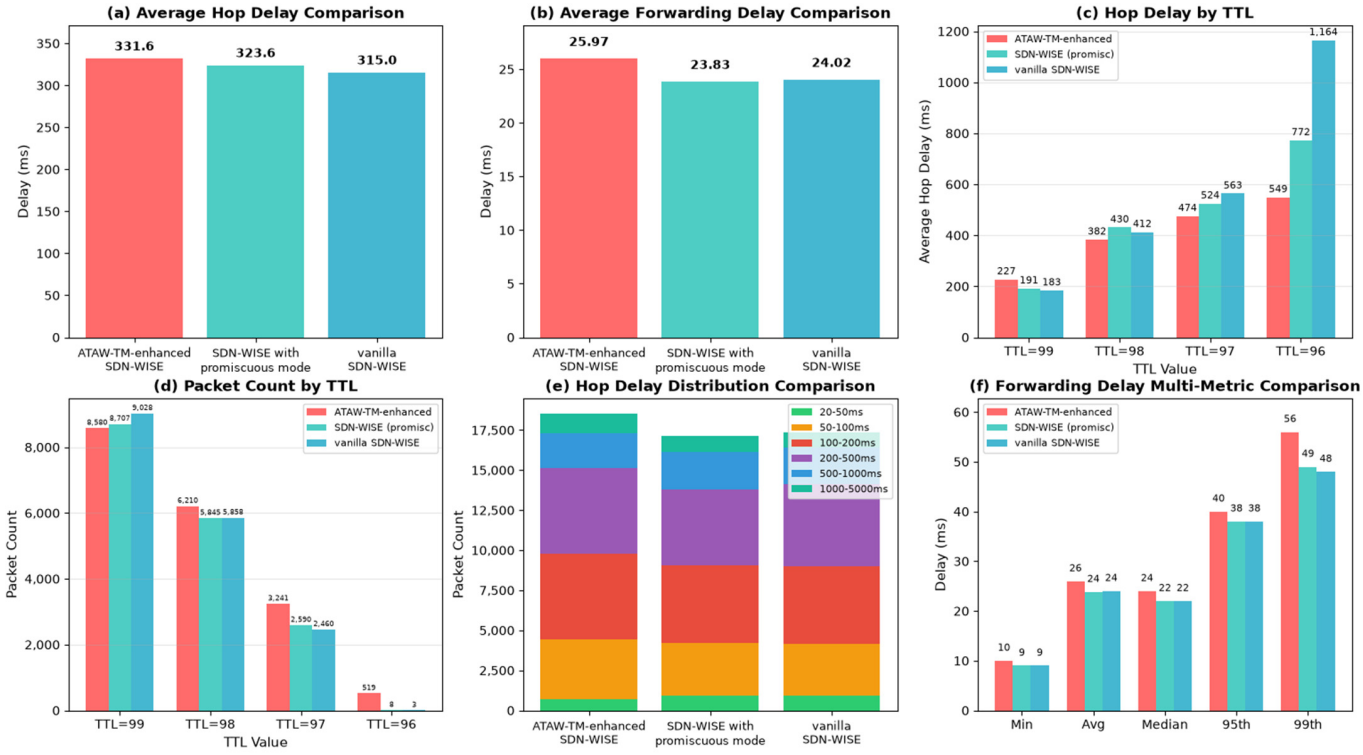


Figure 26. Performance comparison between ATAW-TM-enhanced SDN-WISE, SDN-WISE with nodes in promiscuous mode, and standard SDN-WISE.

As shown in Figure 26c, packets with TTL = 99 (originating farthest from the sink) experience the lowest per-hop delay (227.09 ms for ATAW-TM-enhanced SDN-WISE), while packets with TTL = 96 (closest to the sink) experience significantly higher delay (549.26 ms). This trend reflects increased congestion and radio interference near the sink, where multiple flows converge. Notably, the ATAW-TM-enhanced version maintains consistent performance across all TTL levels, with delays of 227.09 ms (TTL = 99), 382.99 ms (TTL = 98), 474.97 ms (TTL = 97), and 549.26 ms (TTL = 96). Figure 26d shows the packet count distribution across TTL values. The initial TTL value of the packets is set to 100, so packets with a TTL of 99 are those originally sent by the source, and the TTL decreases by 1 after each hop. As can be seen from the figure, the packet count distribution across TTL values remains largely consistent across the three experimental scenarios.

Figure 26e shows the stacked distribution of hop delays across six intervals. For all three protocols, most packets fall within the 100–200 ms and 200–500 ms ranges, each accounting for approximately 28–29% of the total number of packets. Our ATAW-TM-enhanced SDN-WISE protocol exhibits a slightly higher proportion of packets in the 1000–5000 ms range (6.7%) compared to the standard SDN-WISE (5.5%) and SDN-WISE with promiscuous mode (5.9%). This increase may be attributed to the additional trust computation overhead and the debouncing mechanism applied during low-traffic periods.

Figure 26f presents a multi-metric comparison of forwarding delays, showing the minimum, average, median, 95th percentile, and 99th percentile values across the three

protocols. The ATAW-TM-enhanced SDN-WISE protocol achieved a median forwarding delay of 24.00 ms, comparable to the other two protocols (both at 22.00 ms). The 99th percentile delay in the proposed trust-embedding approach is 56.00 ms, which is 14.3% higher than standard SDN-WISE (48.00 ms). In WSN-based IoT deployments, the minimum acceptable delay for data delivery typically varies between 0 and 250 ms, while the maximum acceptable delay is usually in the range of 250–1000 ms, depending on the application scenario [34]. The 99th percentile delay of 56 ms achieved by the proposed scheme falls well below these thresholds, confirming its suitability for real-time IoT scenarios.

3.6.2. Resource Consumption Comparison and Function Variants

ROM/RAM Usage

As shown in Table 10, the memory footprint varies depending on the build configuration and the mathematical functions used. In the debug build, the unmodified functions version of ATAW-TM-enhanced SDN-WISE requires 47.49 KB ROM and 9.12 KB RAM, while the lightweight functions version reduces the ROM footprint to 45.75 KB (retaining the same 9.12 KB RAM).

Table 10. Memory Usage Comparison.

Build	Version	ROM (KB)	RAM (KB)
Debug	ATAW-TM-enhanced SDN-WISE (unmodified functions)	47.49	9.12
Debug	ATAW-TM-enhanced SDN-WISE (lightweight functions)	45.75	9.12
Debug	SDN-WISE with promiscuous mode	35.02	8.82
Debug	Standard SDN-WISE	35.07	8.82
Debug off	ATAW-TM-enhanced SDN-WISE (lightweight functions)	40.16	8.99
Debug off	SDN-WISE with promiscuous mode	30.13	6.80
Debug off	Standard SDN-WISE	30.19	6.80

In the deployment-oriented (debug-off) build, the lightweight ATAW-TM-enhanced SDN-WISE requires only 40.16 KB ROM and 8.99 KB RAM.

Compared with the Standard SDN-WISE implementation, the additional memory overhead introduced by ATAW-TM in the debug build is approximately 10.68 KB ROM and 0.30 KB RAM. In the debug-off build, the additional overhead is 9.97 KB ROM and 2.19 KB RAM.

It is worth noting that the relative RAM overhead appears larger in the debug-off build because the compiler's aggressive optimization significantly reduces the bss and data sections of the standard SDN-WISE (from 8.82 KB to 6.80 KB). The ATAW-TM trust engine, however, requires a fixed allocation for the extended neighbor table and trust metrics, which cannot be entirely optimized away. Despite this relative increase, the absolute RAM usage (8.99 KB) remains well within the 10 KB constraint of the Sky mote (MSP430F1611) platform, demonstrating the feasibility of the proposed integration for resource-constrained deployments.

Energy

Figure 27 shows the remaining energy level of all ordinary sensor nodes after 30 min of operation. The differences among the three protocol versions are small, indicating that the trust-evidence collection and trust-computation mechanisms introduce only a modest additional energy cost.

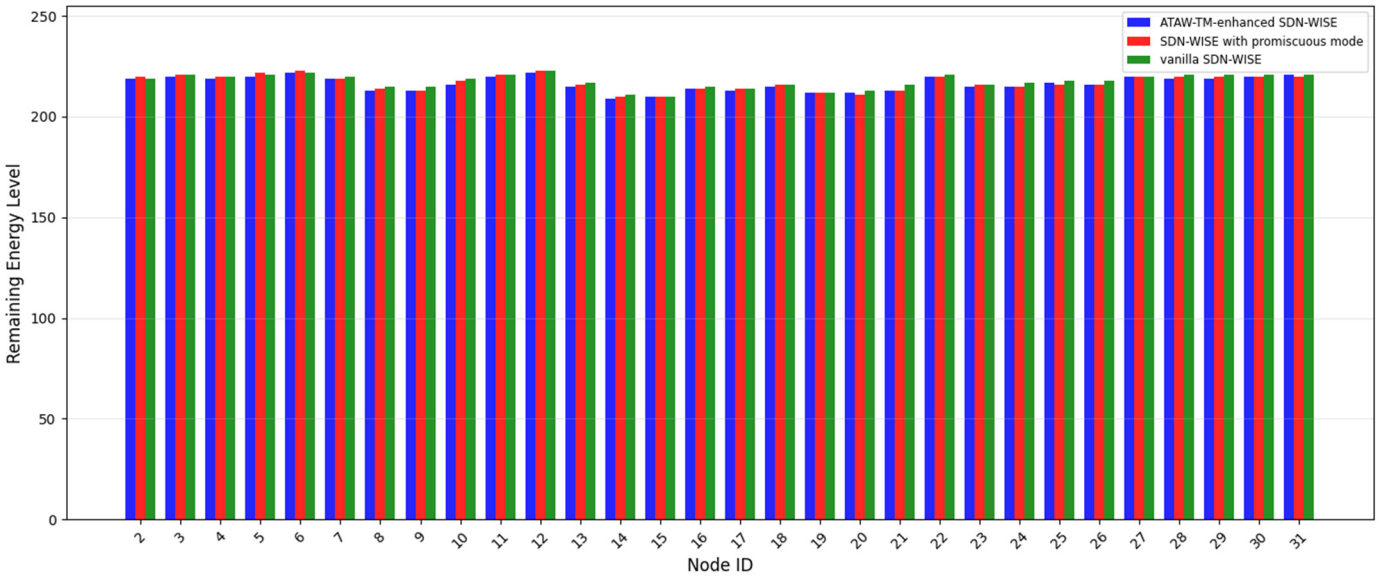


Figure 27. Remaining node energy after 30 min of operation.

Trust Computation Time

To quantify the computational overhead of ATAW-TM, we recorded the execution time required for a node to complete one trust-evaluation cycle for all neighbors currently stored in its neighbor table. We compared the lightweight functions against the unmodified functions (which utilize standard exp() and log() operations). As shown in Table 11, a total of 538 trust-computation records were collected for the lightweight version, with an average computation time of 1.87 s. In contrast, the unmodified functions required an average of 2.19 s across 503 records, with a higher maximum latency (3.48 s). This demonstrates that the lightweight functions effectively reduce the execution time by avoiding software floating-point library calls.

Table 11. Trust Computation Time Comparison between Lightweight and Unmodified Functions.

Metric	Lightweight Functions	Unmodified Functions
Number of Samples	538	503
Minimum Computation Time	0.96 s	0.83 s
Average Computation Time	3.24 s	3.48 s
Maximum Computation Time	1.87 s	2.19 s

3.6.3. Lightweight vs. Unmodified Functions on Detection and Robustness

Beyond computational savings, the function variants exhibit a better trade-off between sensitivity and robustness in trust evolution. As illustrated in Figure 28a,b for the LDoS attack scenario, the lightweight functions demonstrate faster convergence and significantly lower volatility across all nodes compared to the unmodified functions.

In the unmodified functions variant (Figure 28a), the trust values of affected nodes (e.g., Node 13 and Node 14) exhibit severe and prolonged fluctuations. This occurs because the unmodified mathematical formulations (e.g., standard exp() and log()) are highly sensitive to transient network anomalies, which inherently leads to over-penalization and reduced robustness. Conversely, the lightweight approximations act as a smoothing filter (Figure 28b). They effectively mitigate the impact of short-term noise while maintaining the same detection effectiveness (identifying the attacker Node 12 at a nearly identical time). This smoother trust evolution ensures a faster convergence to a stable state; this en-

hances the robustness of the system by decreasing the potential of misclassifying genuinely benign nodes.

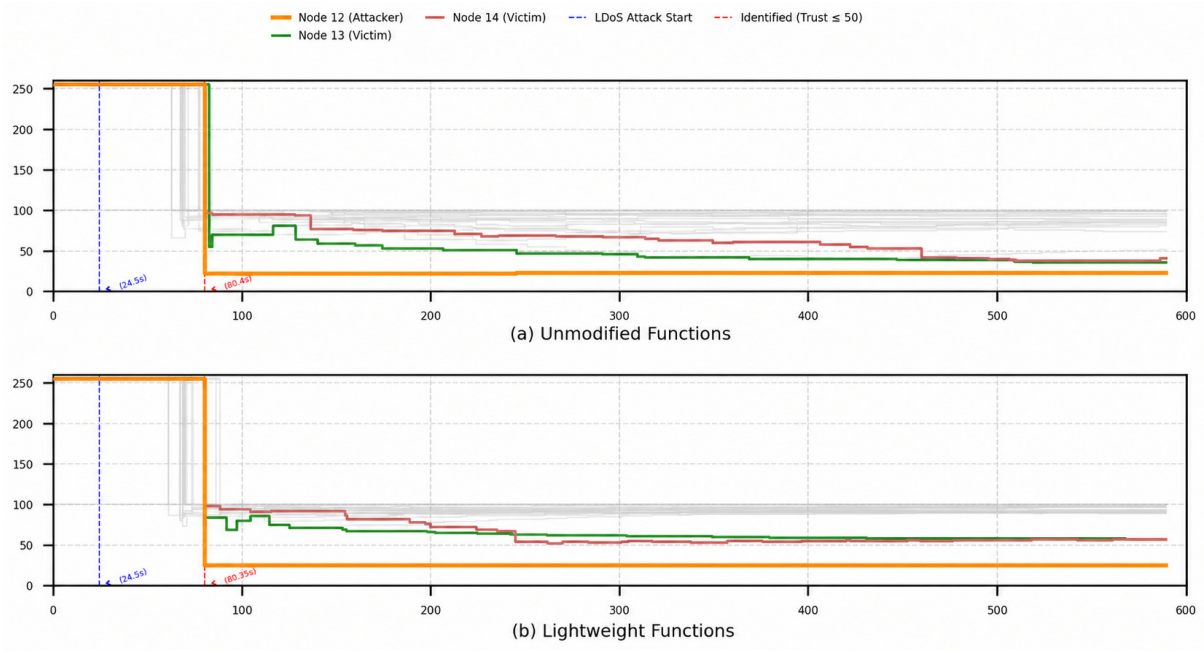


Figure 28. Trust evolution comparison under an LDoS attack: (**a**) Unmodified functions exhibit higher sensitivity but severe volatility and slower convergence; (**b**) Lightweight functions demonstrate smoother trust degradation, faster convergence, and better robustness while maintaining consistent detection accuracy. The grey lines represent the trust values of the remaining normal nodes in the network (excluding the attacker and victim nodes).

4. Discussion

4.1. Overall Detection Performance

The experimental results reported in the preceding section indicated that ATAW-TM-enhanced SDN-WISE successfully addresses all evaluated attack scenarios through attack detection, abnormality identification, or mitigation, depending on the characteristics of the attack. However, in the grey hole and sinkhole attack scenarios, the attacker's trust value eventually stabilizes at a level that is lower than the trust values of the legitimate nodes but still above the detection threshold. Additionally, flooding-type attacks can cause victim nodes to be incorrectly flagged as malicious due to the low precision of the detection (PR as low as 0.57). Even though the trust values of some of the malicious nodes did not fall below the predefined detection threshold of 50 during the observation period, the temporal evolution of trust values remained stable and highly discriminative. As illustrated by the trust-evolution plots presented in Appendix A, the trust values of malicious nodes consistently exhibited a downward trend over time before converging to levels lower than those of legitimate nodes. In contrast, the trust values of legitimate nodes gradually increased and stabilized at significantly higher values. This observation suggests that the proposed trust mechanism is capable of effective long-term differentiation between malicious and legitimate nodes, even in cases where a malicious node does not meet the predefined threshold criterion and is not detected as compromised.

If integrated with a routing protocol, trust values could be used directly as a routing metric. Routing decisions could preferentially select nodes with higher trust values while avoiding nodes with persistently lower trust values. Consequently, even malicious nodes that do not reach the detection threshold could still be excluded from preferred

routing paths due to their relatively low trust values, thereby reducing their influence on network operation.

4.2. Factors Affecting TTD

The experimental results presented in Table 8 show substantial variation in the TTD among different attack types. This variation can be explained primarily by two factors: (i) the temporal accumulation effect of trust history in ATAW-TM and (ii) the visibility of malicious behavior to neighboring nodes.

4.2.1. Effect of Trust-History Accumulation

The first factor is related to the trust update mechanism used by ATAW-TM. As shown in Equations (4) and (5),

$$\alpha^{new} = \alpha + P_{Co} \qquad \beta^{new} = \beta + 1 - P_{Co} \tag{4}$$

$$T = \frac{\alpha^{new}}{\alpha^{new} + \beta^{new}} \tag{5}$$

the parameters α and β are updated cumulatively based on both historical values and newly calculated cooperation probabilities (P_{Co}). Consequently, the trust value (T) is not determined solely by current behavior but also reflects historical interactions. When a node has accumulated a long period of cooperative behavior before launching an attack, its trust value is supported by a large amount of positive historical evidence. Even after malicious behavior begins, several trust evaluation cycles may be required before the trust value falls below the detection threshold.

This effect can be observed by comparing the LDoS and on–off attack scenarios (Appendix A, Figures A6 and A14, respectively). In both cases, the attacker generates low-rate flooding traffic at the same transmission rate. However, in the on–off attack, malicious behavior is initiated later, after a longer period of normal operation. As a result, the attacker accumulates a higher trust value before the attack starts, which increases the time required for trust degradation and leads to a longer TTD.

Although the adaptive aging mechanism already accelerates trust decay when negative behavior is observed, historical evidence still influences the α and β parameters directly. Future work could investigate incorporating an additional aging mechanism into the α and β update process itself. Such an extension could gradually discount older observations and increase sensitivity to recently observed malicious activities, thereby shortening detection time for attacks launched after long periods of cooperative behavior.

4.2.2. Effect of Behavior Visibility

The second factor is the extent to which malicious behavior can be observed by neighboring nodes. Some attacks generate behavior that is visible to nearly all neighbors within radio range. For example, flooding and LDoS attacks produce excessive packet transmissions that can be overheard by every neighboring node. Consequently, multiple observers independently collect negative evidence and report reduced trust values, causing the attacker's aggregated trust value to decline rapidly.

In contrast, other attacks expose malicious behavior only to a subset of neighboring nodes. Selective forwarding in the 51-node topology provides a representative example, as shown in Figure 29. When the attacker (Node 51) drops packets from a specific source (Node 23), as shown in red arrows, only neighbors located within the communication range of the forwarding path (Node 44, Node 45, and Node 20) can observe the forwarding failure. Other neighboring nodes (Node 15 and Node 50) may not even know that the attacker was expected to forward those packets and therefore cannot contribute negative

evidence. Because fewer observers participate in trust degradation, the aggregated trust value decreases more slowly, resulting in longer detection times.

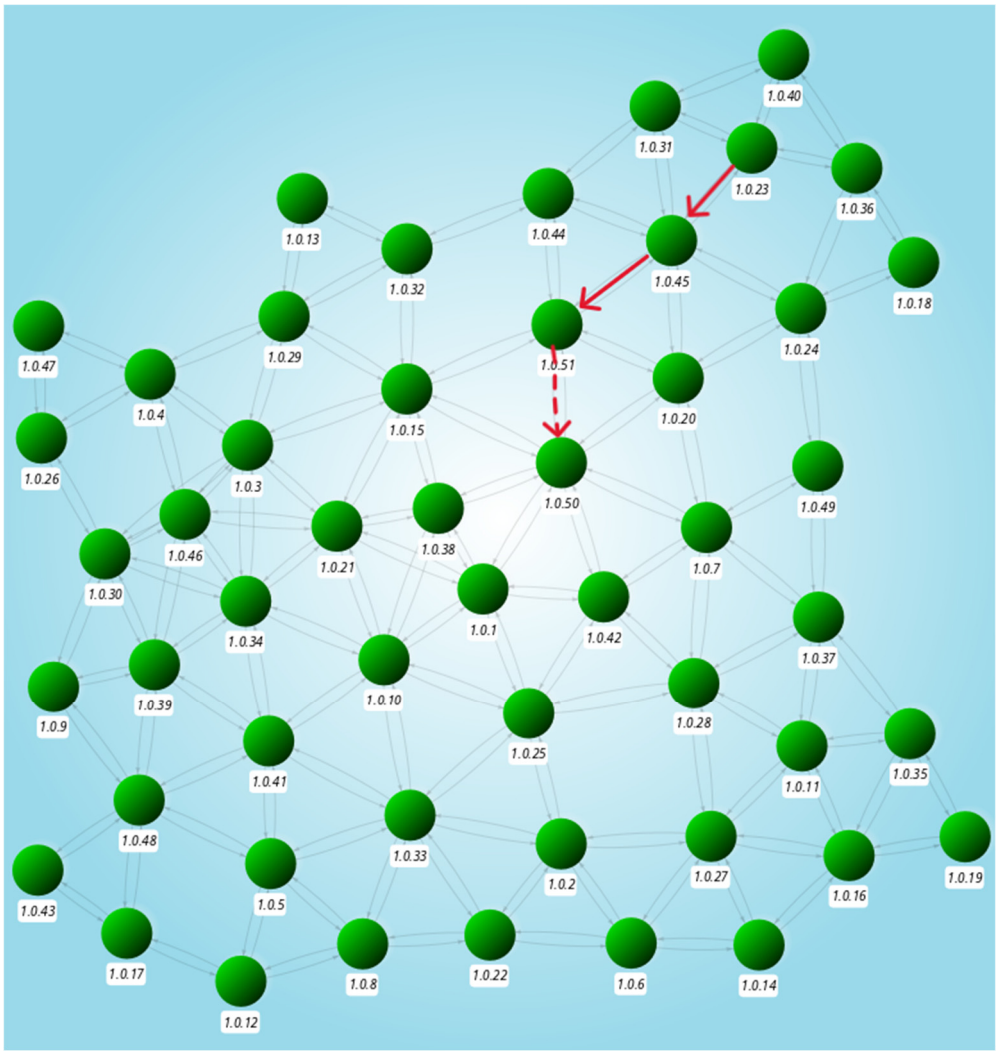


Figure 29. The attack path of a selective forwarding attack by Node 51.

4.3. False Positives and Victim Differentiation

An important observation from Table 8 is that in some attack scenarios (e.g., flooding, LDoS) a small number of legitimate nodes are incorrectly identified as malicious. These nodes are not attackers but attack victims: their trust values degrade because they are affected by the abnormal network conditions created by neighboring malicious nodes. For flooding-type attacks (flooding, LDoS, New-Flow, Hybrid, and DDoS), a possible victim-differentiation approach would be to use the trust metrics already collected by ATAW-TM. Although both attackers and victim nodes may experience trust degradation, attackers typically exhibit abnormal values in the DSR and/or CSR metrics due to excessive packet generation. In contrast, victim nodes usually maintain normal packet-generation behavior while suffering from congestion and forwarding failures induced by attack traffic. Consequently, once a node is identified as suspicious, neighboring nodes could examine the DSR and CSR evidence to distinguish between attackers and victims. Packets originating from confirmed attackers could be selectively dropped or rate-limited, reducing the impact of the attack and preventing further propagation of false trust degradation. (The proposed differentiation approach has not been experimentally evaluated in this study.) For the vampire attack, although false positives may occur, the identified nodes can still be isolated.

Once a false-positive node is isolated, the routing loop is disrupted, causing the vampire attack to fail.

Another issue is that in the case of the control-plane blackhole attack, the proposed approach initially identifies only an anomaly whereby the trust values of a group of nodes remain unavailable or cannot be updated for an extended period. However, the controller could exploit its global view of the network topology to perform attacker inference. When a group of nodes exhibits prolonged absence of trust-value updates, the controller may be able to identify the upstream node that lies closest to the sink along the affected communication paths. Because trust reports from downstream nodes must pass through this node before reaching the controller, it represents the most likely attack source. This approach could enable the controller to distinguish the malicious forwarding node from the downstream victim nodes whose trust values remain unavailable (this approach has not been empirically investigated in this study).

Finally, the results show that the trust model allows the attacker nodes in grey hole and sinkhole attacks to recover their trust levels after a sufficiently long period of time, as in both cases, the attacks do not terminate all cooperative behavior. A grey hole attacker drops packets only intermittently and continues to cooperate during the rest of the time, while a sinkhole attacker continues to forward the packets it is expected to forward; it is the large volume of traffic attracted to it that degrades its forwarding capability. Therefore, after a sufficiently long observation period, the trust value of the attacker nodes stabilizes at a level that is consistently lower than that of legitimate nodes, reflecting their reduced cooperation level. As the stable value may still remain above the detection threshold, complementary mechanisms such as linking routing decisions to trust values may be integrated into the trust model: nodes with relatively low trust values could be avoided as next hops, even if their trust levels remain above the detection threshold (the proposed integration was not evaluated experimentally in this study).

4.4. Adaptability to Traffic Variations

The proposed approach exhibits strong adaptability to varying network traffic rates without requiring manual threshold tuning for different rate conditions. This feature significantly reduces deployment complexity and enhances generalizability across diverse operational scenarios. Nevertheless, all evaluations were conducted using a CBR traffic model. CBR is a widely adopted traffic model in WSN performance evaluations [34]. It has been used as a representative real-time traffic pattern in cognitive radio sensor networks [29] and as a baseline application model in underwater WSN studies [35].

4.5. Computational Overhead and Resource Consumption

ATAW-TM-enhanced SDN-WISE adds multi-type DoS attack detection capabilities without sacrificing significant network performance. The average forwarding delay increases by less than 2 ms compared to the standard SDN-WISE. The modest overhead increase can be attributed primarily to the trust evidence collection process, as the trust computation itself is performed asynchronously and does not contribute to forwarding delay. As shown in Figure 26a, enabling promiscuous mode alone (without evidence collection or trust computation) increases single-hop delay, reflecting the inherent burden of promiscuous reception on the radio stack. Adding evidence collection and trust computation further increases single-hop delay, indicating the additional CPU load introduced by these processes. We also observe that single-hop delay increases as TTL decreases (i.e., as packets approach the sink node). This trend reflects higher contention and interference in areas of traffic convergence, where multiple flows compete for channel access. This observation aligns with expected network behavior. The data about memory usage, energy

consumption, and trust-computation time further indicates that the proposed integration remains feasible on MSP430-class sensor hardware. The RAM footprint remains below 10 KB, the ROM footprint remains below 48 KB, and the average trust-computation time is approximately 1.9 s per 30 s evaluation period.

Although the proposed integration remains within the hardware limits of the Sky mote, the RAM footprint (8.99 KB in the debug-off build) is close to the 10 KB capacity of the MSP430F1611 microcontroller. As discussed in Section 5, a decoupled architecture (e.g., a sidekick module) or more aggressive memory optimizations (e.g., dynamic allocation or smaller data types) could be explored in future work to alleviate this constraint.

4.6. Impact of Aging and Debouncing Mechanisms

The experimental results highlight an important design consideration for trust management in resource-constrained wireless sensor networks: small changes in the design of trust update mechanisms can lead to substantial improvements in overall system performance.

The aging mechanism provides an example of how a minor design adjustment can produce a substantial performance improvement. We evaluated two aging strategies: applying the aging factor from the first trust evaluation round and applying it from the second evaluation round onward. Although the difference between the two strategies is small from an implementation perspective, delaying the application of the aging factor until the second evaluation round substantially accelerates the initial convergence of legitimate-node trust values. In our experiments, the trust values of all sensor nodes converged in approximately 200 s, compared with approximately 800 s when aging was applied from the first evaluation round. This corresponds to a reduction of approximately 75% in the convergence time.

The debouncing mechanism is similarly important for practical deployment. In a real WSN or SDWSN, sensor nodes may not start simultaneously, and legitimate nodes may temporarily generate little or no traffic because of asynchronous startup, intermittent sensing, or application-specific traffic patterns. If the absence of trust evidence is immediately interpreted as non-cooperative behavior, legitimate nodes may receive unstable or artificially low trust values during network initialization or low-traffic periods. Our experiments show that the debouncing mechanism helps prevent such fluctuations by treating sufficiently low traffic as insufficient evidence for negative trust updates. When debouncing was disabled, the trust values exhibited substantial fluctuations and converged considerably more slowly, with legitimate nodes reaching the trust threshold only at around 400 s in the evaluated scenario. This observation is particularly relevant to practical deployment because low traffic does not necessarily imply malicious behavior. A trust model designed primarily around continuously available evidence may perform well in a controlled simulation but become unstable when deployed in networks with asynchronous node activation, intermittent communication, or heterogeneous traffic patterns. The debouncing mechanism addresses this issue by preventing temporary or insufficient evidence from immediately influencing trust values.

4.7. Comparison with Existing Trust Models

To position this study within the context of extant research, Table 12 provides a comparison of ATAW-TM-enhanced SDN-WISE and three other models proposed in the literature: ETMRM [18], TSW [17], and the original ATAW-TM [19].

While ATAW-TM was evaluated primarily through theoretical analysis, this study integrates ATAW-TM into the SDN-WISE protocol stack and includes methods for trust-evidence collection, node-level trust computation, controller-level trust aggregation, and attack recognition that were developed within a Contiki/Cooja-based SDWSN simulation environment. Overall, the ATAW-TM-enhanced SDN-WISE protocol architecture was eval-

uated against a broader set of attack scenarios, including 14 DoS attack types and one data-tampering attack, providing the most comprehensive attack evaluation among the compared approaches.

Table 12. Comparison with Existing Trust-Based Approaches (✓ indicates the feature is supported; ▫ indicates it is not supported).

Feature	**ETMRM [18]**	**TSW [17]**	**ATAW-TM [19]**	**This Work (ATAW-TM-Enhanced SDN-WISE)**
Network	SDWSN	SDWSN	SDWSN	SDWSN
Number of Trust Metrics	Forwarding (data and control respectively), New-Flow, 3 in total.	Forwarding (data and control respectively), Sending Rate (data and control respectively), New-Flow, Reliability, 6 in total.	DSR, CSR, DRR, CRR, DFR, CFR, ECR, DA, 8 in total.	DSR, CSR, DRR, CRR, DFR, CFR, ECR, DA, 8 in total.
Weight Assignment	Fixed	Fixed	Entropy-based Automatic Weighting	Entropy-based Automatic Weighting
Aging Mechanism	None	Fixed	Adaptive Aging Factor	Adaptive Aging Factor
Link Quality Awareness	✗	✗	✓	✓
Flooding Detection Requires Traffic-Rate Threshold	✓	✓	✗	✗
DoS Attack Coverage	Grey hole, Blackhole, Selective Forwarding Attack, New-Flow Attack.	Blackhole Attack, Selective Forwarding Attack, DoS Attack, Good-Mouthing Attack, On–Off Attack, Hybrid DoS Attacks, New-Flow Attack.	14 DoS Attack Types	14 DoS Attack Types
Data Tampering Detection	✗	✗	✓	✓
Protocol-Level Integration	✓	✗	✗	✓
Validation Environment	Cooja (Contiki 2.7)	MATLAB (version not specified)	Theoretical Analysis	Cooja (Contiki-3.x-3330-g719f71230)
Open-Source Implementation Available	✗	✗	✗	✓ (Supplementary Materials)

The proposed solution addresses several significant limitations of ETMRM [18] and TSW [17]. Both approaches employ fixed trust-metric weights and static aging factors, which may become suboptimal as network conditions and attack behaviours evolve. In contrast,

the proposed model utilizes entropy-based automatic weighting and an adaptive aging mechanism, enabling trust evaluations to adjust dynamically to observed node behaviour. Consequently, detection effectiveness can be maintained under changing network conditions without requiring manual parameter reconfiguration.

Furthermore, both ETMRM and TSW depend on traffic-rate-based thresholds for detecting flooding-type attacks. Such thresholds typically require calibration for different traffic rates. The ATAW-TM-enhanced SDN-WISE protocol framework eliminates this dependency by evaluating node behavior relative to the behavior of neighboring nodes rather than against fixed traffic-rate thresholds. This improves adaptability and reduces deployment complexity while maintaining consistent detection performance across different traffic conditions.

Finally, neither ETMRM nor TSW considers data accuracy as a trust metric. As a result, these approaches are unable to detect data tampering attacks directly. By integrating data accuracy into the trust evaluation process, the proposed architecture extends attack coverage beyond DoS attacks and enables the detection of malicious data manipulation, thereby improving the comprehensiveness of trust-based security assessment.

5. Conclusions

This study proposes a novel trust model (ATAW-TM) integration in the SDN-WISE protocol stack that enables protocol-native detection and mitigation of multiple types of DoS attacks and data tampering attacks. The implementation includes a lightweight embedded adaptation of the original ATAW-TM to adapt the model to the resource-constrained sensor nodes.

The primary contribution of this work is the protocol-level implementation and validation of the ATAW-TM trust model. To enable the deployment of the original ATAW-TM model on resource-constrained sensor nodes, three lightweight standard functions were adopted to replace the corresponding exp(), and log() functions, in cooperation-probability computation, entropy-based weight calculation, and trust aging.

Second, this study contributes a comprehensive evaluation framework that rk comprising a significant range of attacks on SDWSNs (14 DoS attacks and the data tampering attack); the framework can be used to test the performance of new security-enhancing integrations.

The evaluation results indicate that the proposed approach can successfully address all evaluated attack scenarios through attack detection, abnormality identification, or mitigation, while remaining feasible for deployment on MSP430-class sensor devices. The model adapts to variations in network traffic conditions without requiring manual threshold tuning. The experimental results show that protocol-native trust management can provide practical security enhancements for resource-constrained SDWSNs without sacrificing protocol efficiency.

Despite these positive results, this study has several limitations that should be acknowledged. First, several specific scenarios (discussed in detail in Section 3) present a challenge to the trust model. In the case of the grey hole and sinkhole attacks, the trust value of the attacker node may stabilize at a level that may be lower than that of the trust values of legitimate nodes but is still above the detection threshold (i.e., the attacker node is flagged as benign, a false negative). Furthermore, in the case of some control-plane attacks (e.g., blackhole), vampire attacks, and flooding-type attacks (flooding, LDoS, New-Flow), the model cannot reliably distinguish between malicious nodes and legitimate victims. As victim nodes may not re4ciecve receive trust updates or experience degraded forwarding performance due to malicious traffic flooding, their trust values may become invalid or drop below the predefined threshold (leading to false positives). This limitation is inherent to trust-based systems under extreme network stress [36]. It can be addressed by adding a

complementary mechanism based on machine learning (ML) and deep learning (DL) for anomaly detection at the controller [37]. Alternatively, the lightweight method discussed in Section 4.3 can be used to achieve more robust differentiation.

Second, modelling network traffic using a CBR approach may not capture the burstiness of real-world sensor traffic fully [30]. While CBR is used in sensor network models [35], CBR-based experimental settings may lead to unrealistic performance metrics across all layers of the protocol stack. The adaptability of ATAW-TM-enhanced SDN-WISE should be validated further by applying a more comprehensive range of traffic models, including bursty, Poisson, and event-driven models [38].

Another limitation is the design decision to embed the trust model directly into the SDN-WISE protocol stack to minimize communication overhead and avoid additional control-plane traffic, which is a critical consideration for resource-constrained sensor motes. While this tight integration enables lightweight and timely trust evaluation, it inherently couples the detection logic with the protocol implementation. Consequently, updating or replacing the trust evaluation method would require modifications to the core protocol code, potentially increasing development and maintenance complexity.

Finally, the memory footprint of the integrated trust model is close to the hardware limits of the Sky mote; Section 4.5 discusses this constraint and proposes mitigation strategies.

Building on the findings of this study, we propose several directions for further investigation and research that may improve model reliability and robustness.

(i) Integration with routing and data fusion: The trust values produced by ATAW-TM-enhanced SDN-WISE can be used to optimize both routing protocols and data fusion algorithms. In routing, trust values can guide path selection to favor trustworthy nodes and circumvent malicious ones. In data fusion, measurements from nodes with higher trust can be assigned a greater weight, enhancing the fidelity of the fused result while mitigating the influence of compromised or malfunctioning nodes.

(ii) Evaluation under diverse traffic models: Future work should evaluate ATAW-TM-enhanced SDN-WISE under a broader range of traffic models, including bursty, Poisson, and event-driven traffic patterns, and validate the proposed approach on physical sensor platforms. Such studies would provide additional insight into the robustness and deployment readiness of the system under realistic operating conditions.

(iii) Distinguishing between malicious nodes and victims: Some potential attacker-identification strategies for flooding-type attacks and control-plane blackhole attacks were proposed in Section 4.3. Future work will implement and evaluate these differentiation mechanisms and investigate their effectiveness in reducing false-positive rates.

(iv) Real-world deployment: While the evaluation was conducted in a simulated environment (Cooja/Contiki), deploying ATAW-TM-enhanced SDN-WISE on physical hardware (e.g., TelosB, CC2420 motes) would provide insights into real-world energy consumption, memory footprint, and timing behavior.

(v) To address the coupling-related limitation, future work could explore a decoupled architectural alternative, where trust evaluation is delegated to an optional sidekick module that runs alongside the protocol stack. Such a sidekick would operate in a loosely coupled manner, performing reputation checks on demand and communicating with the protocol core through well-defined interfaces. This design would enable independent evolution of detection methods, allowing new algorithms to be integrated without modifying the underlying SDN-WISE implementation. The sidekick could also be enabled or disabled as needed, offering flexibility across different deployment scenarios. However, such decoupling would introduce additional inter-module communication overhead and memory footprint, which would need to be carefully evaluated against the resource constraints of low-power sensor platforms. We

consider this a promising direction for extending the adaptability and maintainability of trust-based security solutions in SDWSNs.

Supplementary Materials: The following supporting information can be downloaded at https://drive.google.com/drive/folders/1et0Snm9EPGTnIZbrx1qwYfLo72QGAmcN (accessed on 25 October 2026): Code and data.

Author Contributions: Conceptualization, L.W. and K.P.; methodology, L.W. and K.P.; validation, L.W.; formal analysis, L.W.; investigation, L.W.; resources, M.L.Y.; data curation, L.W.; writing—original draft preparation, L.W., M.L.Y. and K.P.; writing—review and editing, L.W., M.L.Y. and K.P.; visualization, L.W.; supervision, M.L.Y. and K.P.; project administration, M.L.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The source code of the ATAW-TM-enhanced SDN-WISE implementation, attack implementation code, Contiki node binaries, simulation configuration files, node-side logs, controller-side logs, and Python analysis scripts used to generate reported results are available as Supplementary Materials.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ATAW-TM	Adaptive, Threshold-Free, and Automatically Weighted Trust Model
CBR	Constant Bit Rate
CFR	Control Packet Forwarding Rate
CRR	Control Packet Receiving Rate
CSR	Control Packet Sending Rate
DA	Data Accuracy
DDoS	Distributed Denial-of-Service
DFR	Data Packet Forwarding Rate
DoS	Denial-of-Service
DRR	Data Packet Receiving Rate
DSR	Data Packet Sending Rate
ECR	Energy Consumption Rate
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
LDoS	Low-rate Denial-of-Service
LQI	Link Quality Indicator
MAC	Medium Access Control
MCU	Microcontroller Unit
ML/DL	Machine Learning/Deep Learning
MSP430	Mixed-Signal Processor (Texas Instruments)
RAM	Random Access Memory
ROM	Read-Only Memory
RPL	Routing Protocol for Low-Power and Lossy Networks
SDN	Software-Defined Networking
SDN-WISE	A stateful SDN solution for Wireless Sensor and Actuator Networks
SDWSN	Software-Defined Wireless Sensor Network
TTL	Time-to-Live
TTD	Time-to-Detect
WSN	Wireless Sensor Network

Appendix A

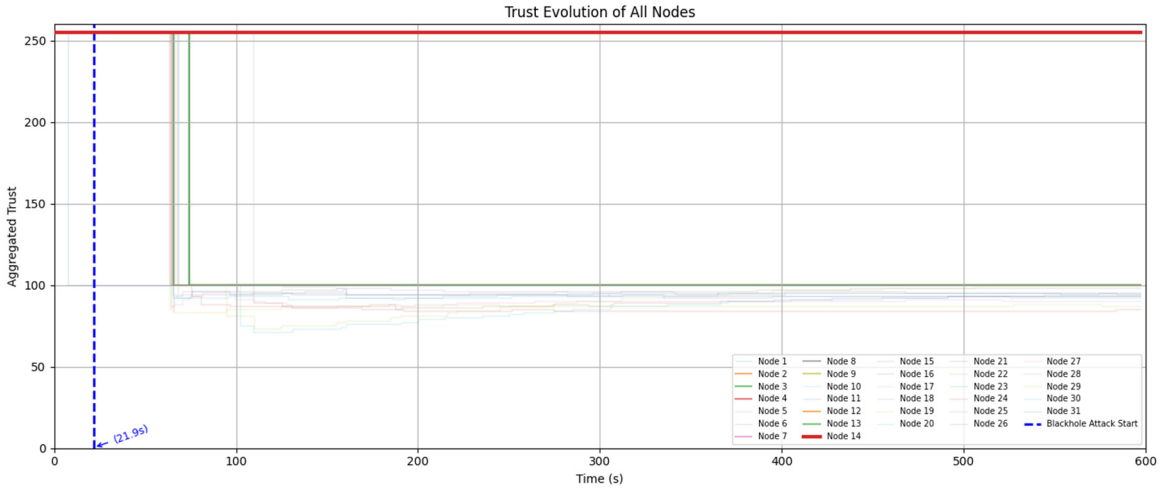


Figure A1. The evolution of the aggregated trust values of all nodes under a blackhole attack by Node 14 targeting the control plane.

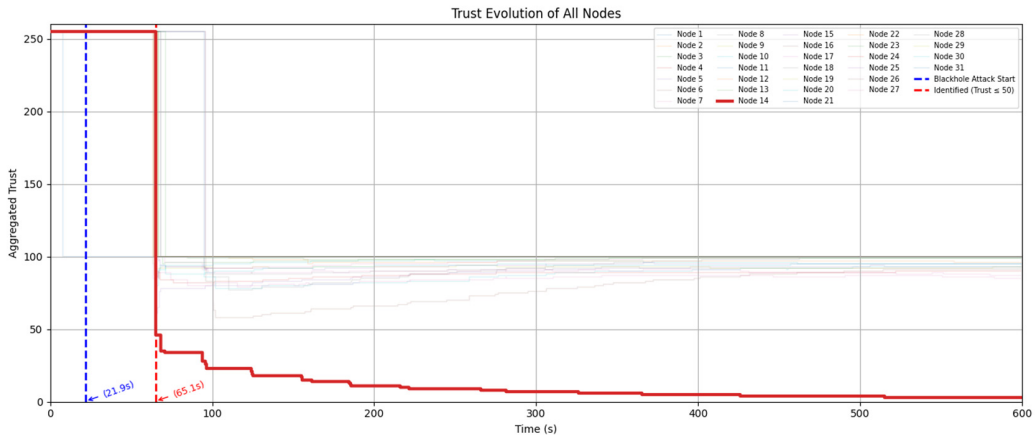


Figure A2. The evolution of the aggregated trust values of all nodes under a blackhole attack by Node 14 targeting the data plane.

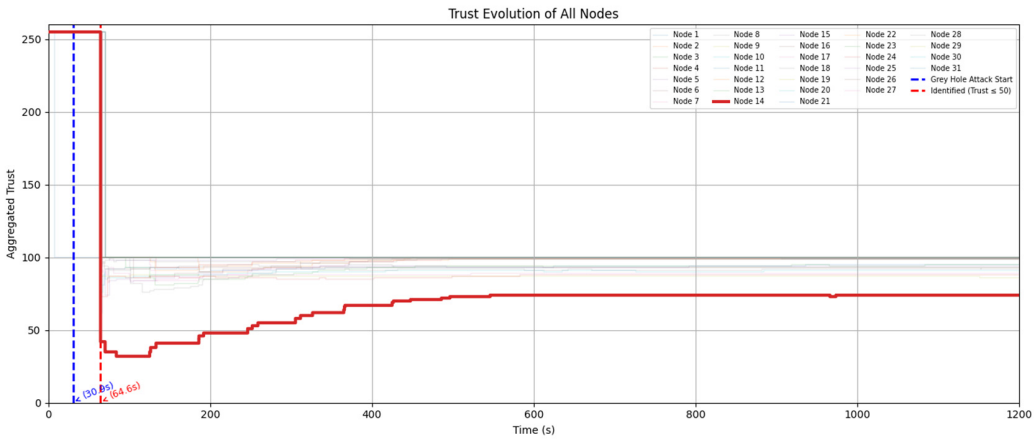


Figure A3. The evolution of the aggregated trust values of all nodes under a grey hole attack by Node 14.

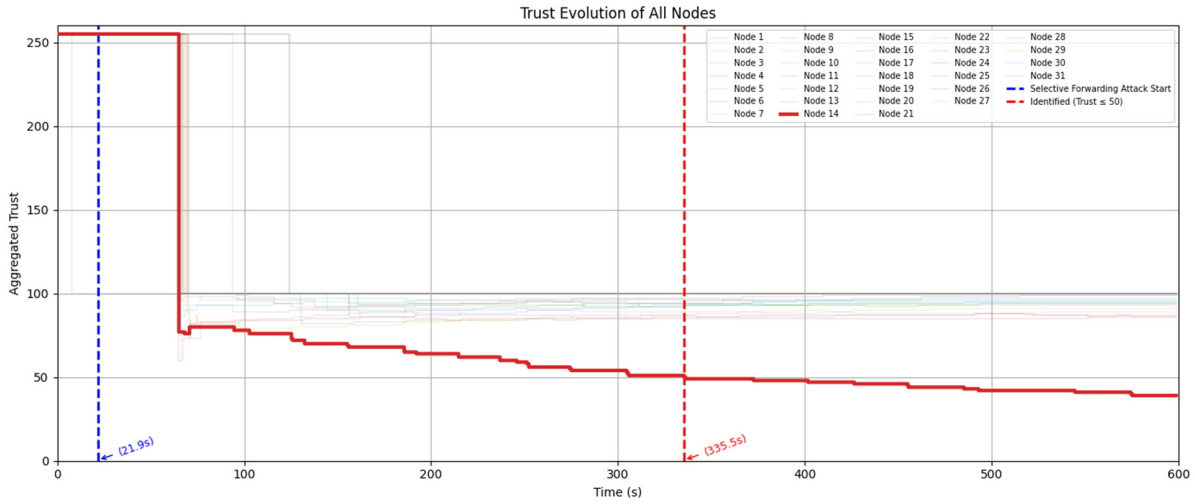


Figure A4. The evolution of the aggregated trust values of all nodes under a selective forwarding attack by Node 14.

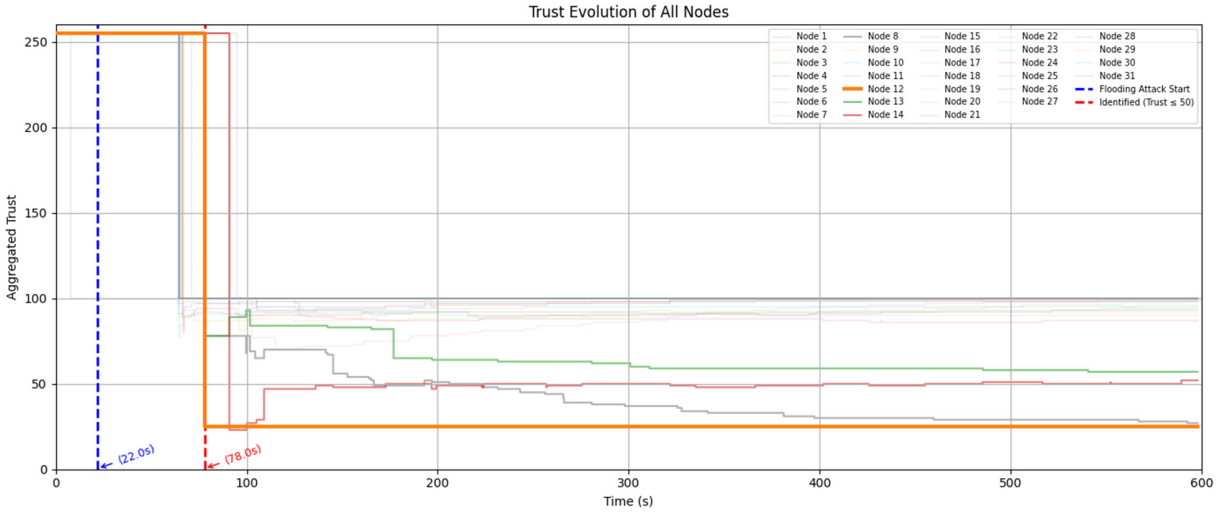


Figure A5. The evolution of the aggregated trust values of all nodes under a flooding attack by Node 12.

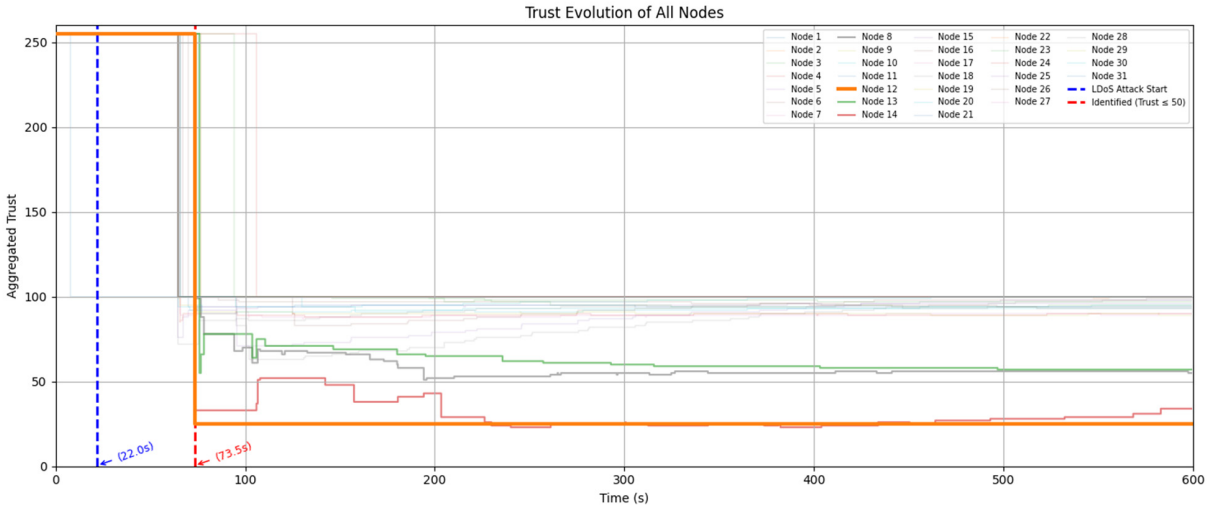


Figure A6. The evolution of the aggregated trust values of all nodes under an LDoS attack by Node 12.

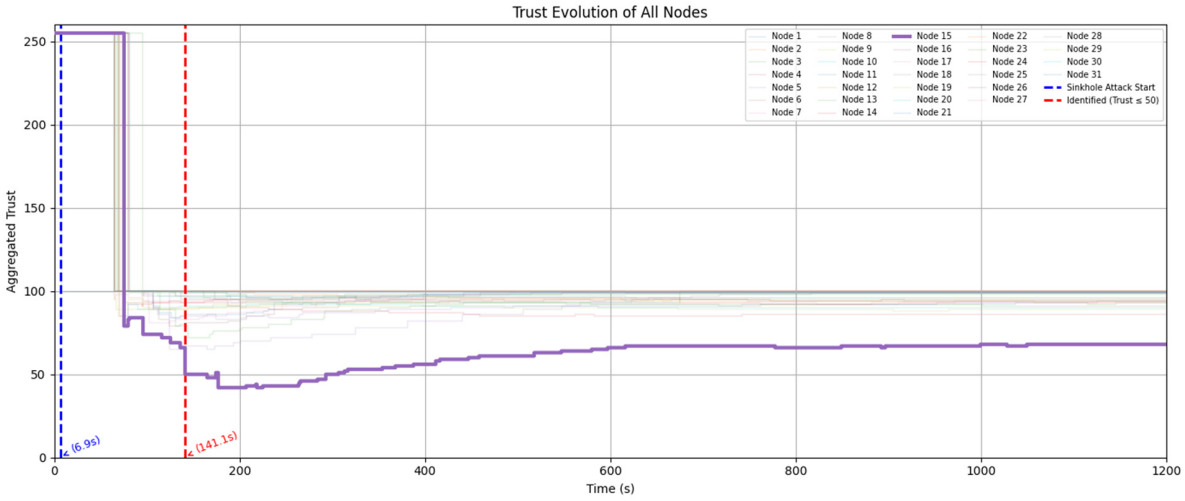


Figure A7. The evolution of the aggregated trust values of all nodes under a sinkhole attack by Node 15.

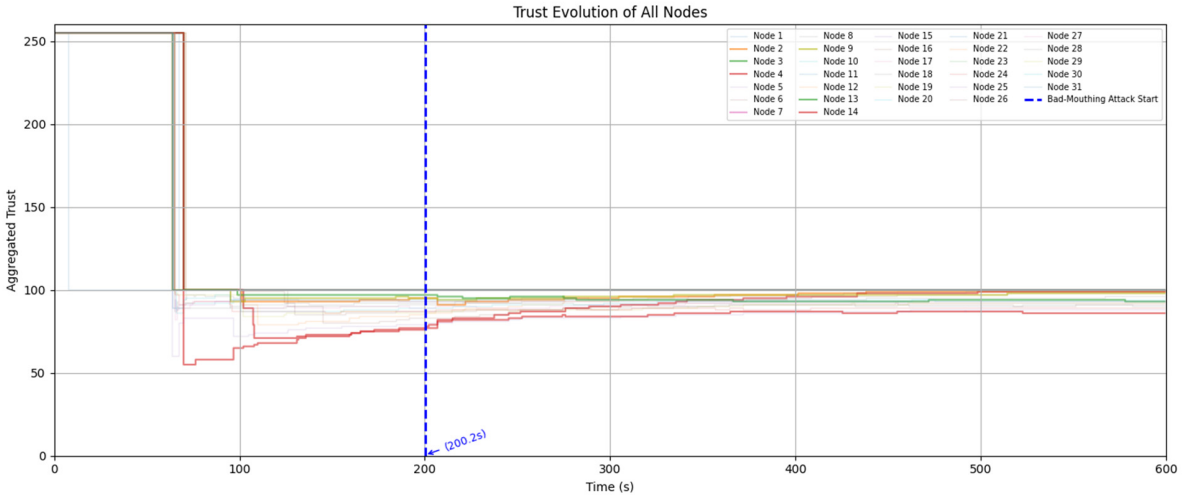


Figure A8. The evolution of the aggregated trust values of all nodes under a bad-mouthing attack by Node 8.

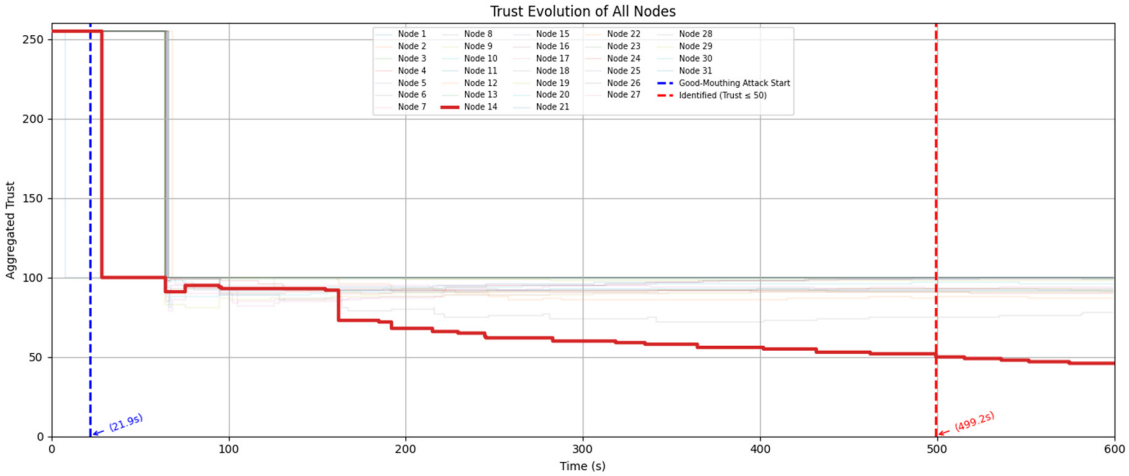


Figure A9. The evolution of the aggregated trust values of all nodes under a good-mouthing attack, where Node 8 conceals the selective forwarding attack of Node 14.

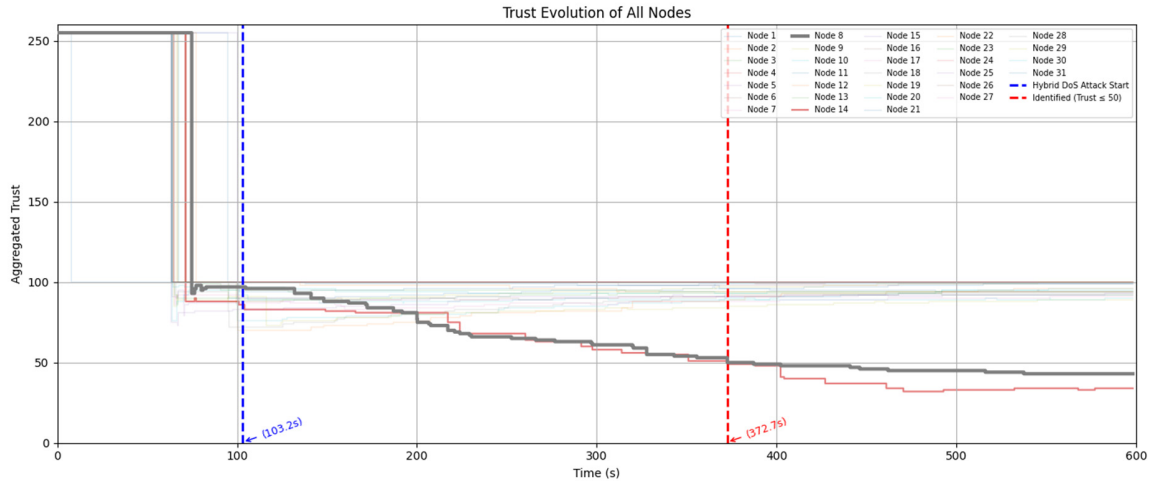


Figure A10. The evolution of the aggregated trust values of all nodes under a hybrid attack by Node 8.

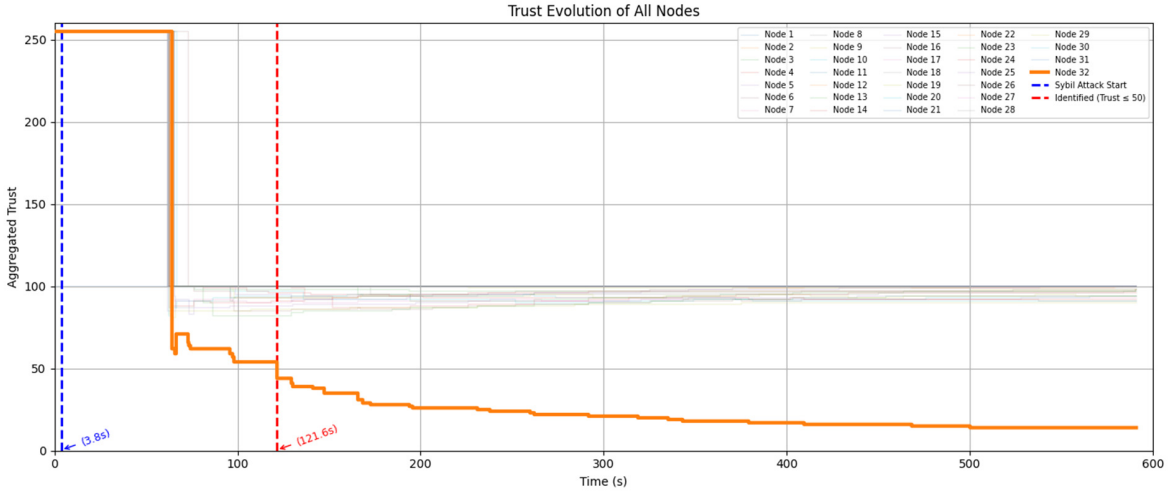


Figure A11. The evolution of the aggregated trust values of all nodes under a sybil attack with a forged identity (Node 32).

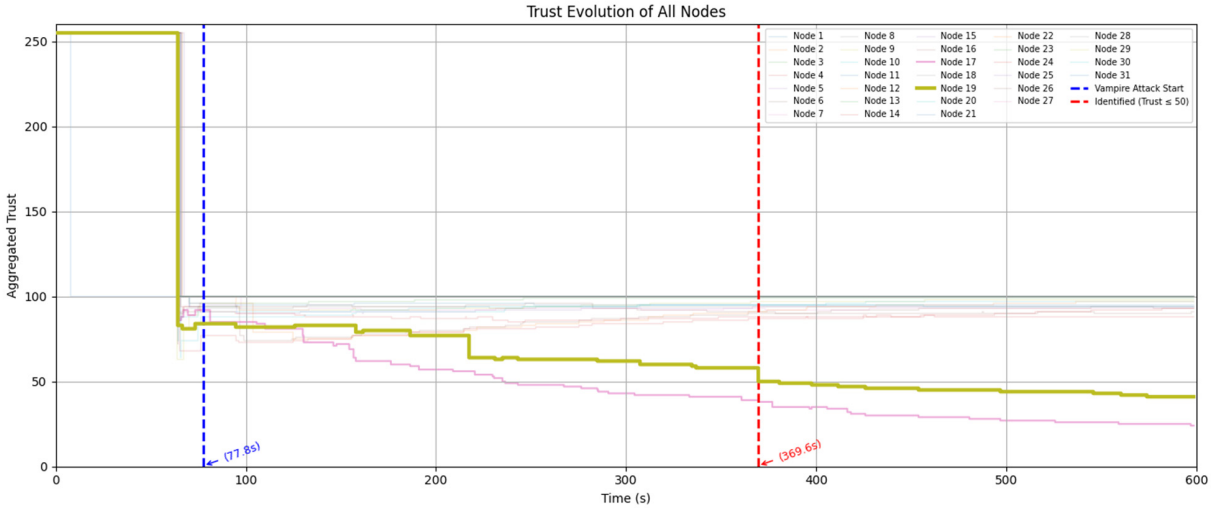


Figure A12. The evolution of the aggregated trust values of all nodes under a vampire attack by Node 19.

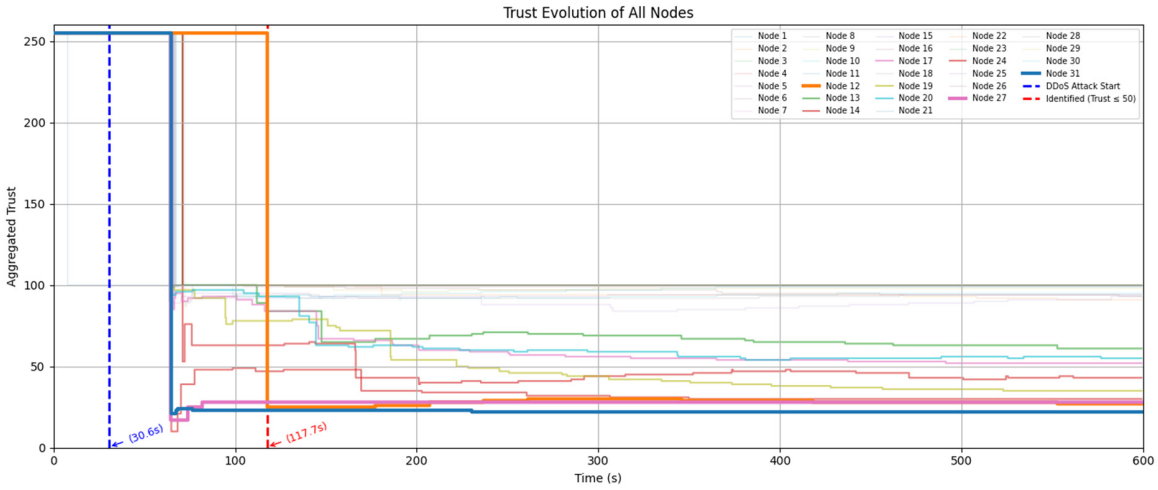


Figure A13. The evolution of the aggregated trust values of all nodes under a DDoS attack by Node 12, 27, and 31.

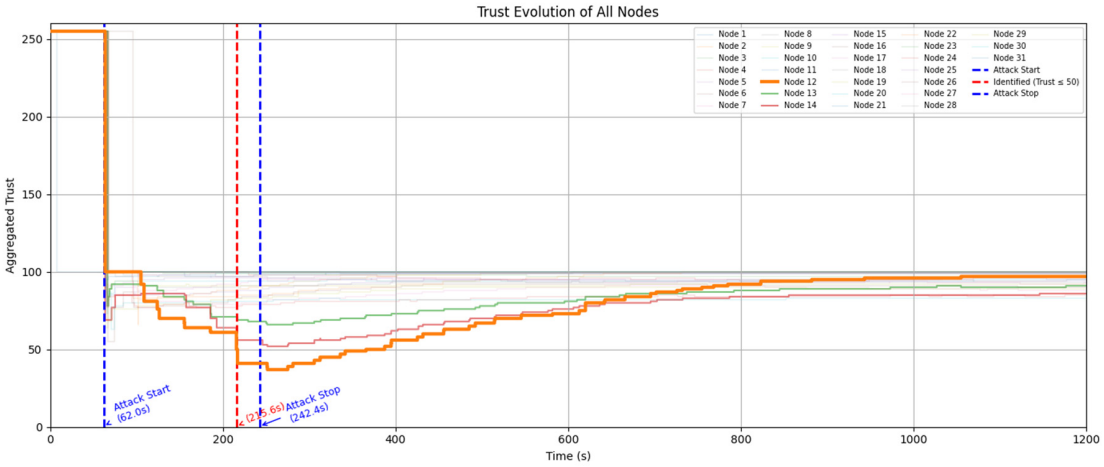


Figure A14. The evolution of the aggregated trust values of all nodes under an on–off attack by Node 12.

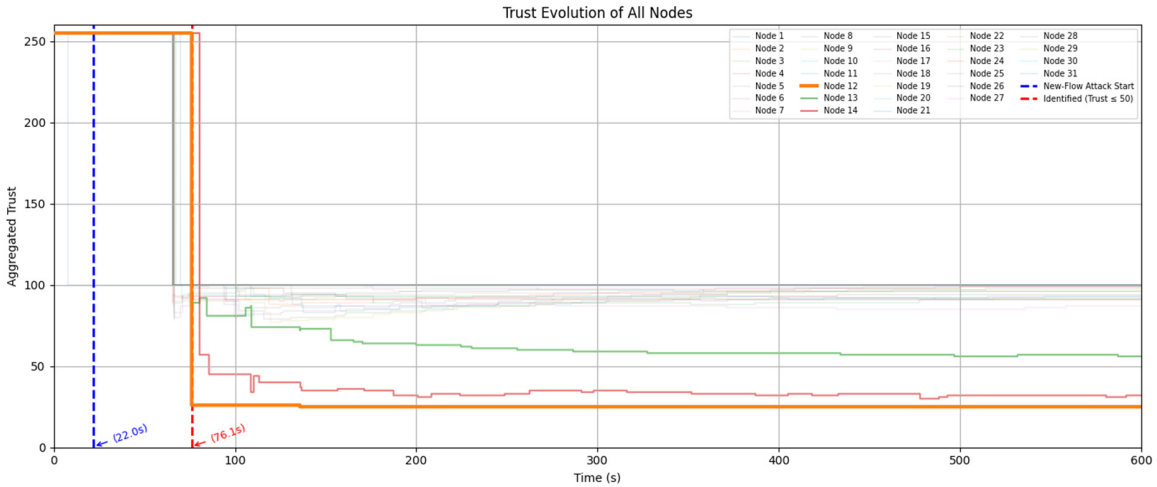


Figure A15. The evolution of the aggregated trust values of all nodes under a new flow attack by Node 12.

https://doi.org/10.3390/electronics15215084

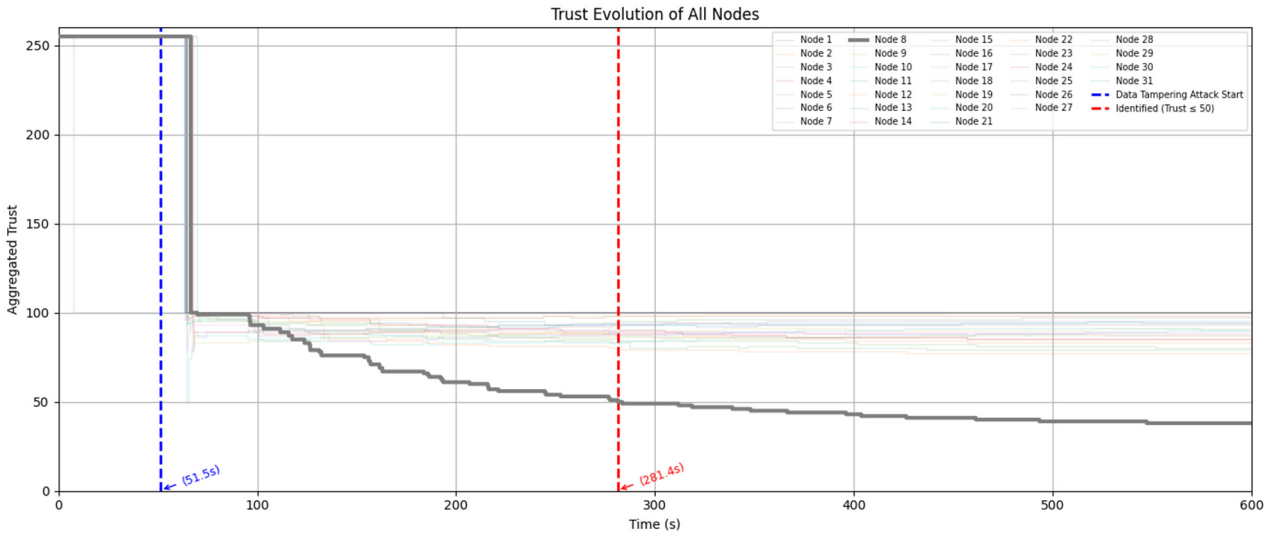


Figure A16. The evolution of the aggregated trust values of all nodes under a data tampering attack by Node 8.

References

1. Kumar, N.; Beniwal, R. Energy-Efficient Techniques in IoT-based Software-Defined Wireless Sensor Networks: A Systematic Review. In Proceedings of the 2025 7th International Conference on Energy, Power and Environment (ICEPE), Sohra, India, 9–11 May 2025; pp. 1–6.
2. Najjar, S.; David, M.; Derigent, W.; Zouinkhi, A. Dynamic reconfiguration of wireless sensor networks: A survey. *Comput. Netw.* **2025**, *262*, 111176. [CrossRef]
3. Sharma, V.; Beniwal, R.; Kumar, V. Multi-level trust-based secure and optimal IoT-WSN routing for environmental monitoring applications. *J. Supercomput.* **2024**, *80*, 11338–11381. [CrossRef]
4. Mohammadzadeh, Z.; Saeidnia, H.R.; Lotfata, A.; Hassanzadeh, M.; Ghiasi, N. Smart city healthcare delivery innovations: A systematic review of essential technologies and indicators for developing nations. *BMC Health Serv. Res.* **2023**, *23*, 1180. [CrossRef] [PubMed]
5. Li, P.; Kang, R. Optimizing the Design and Application of Wireless Sensor Networks in Smart Cities. In *Smart Infrastructures in the IoT Era*; Al-Turjman, F., Ed.; Springer Nature: Cham, Switzerland, 2025; pp. 861–875.
6. Sharma, T.K.; Dhanda, S.S.; Panwar, D.; Singh, B.; Jindal, P. Software-Defined Networking in the Internet of Things. In *Understanding Internet of Things: Evolution, Enabling Technologies and Challenges*; Springer: Berlin/Heidelberg, Germany, 2026; pp. 85–97.
7. Galluccio, L.; Milardo, S.; Morabito, G.; Palazzo, S. SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 26 April–1 May 2015; pp. 513–521.
8. Anadiotis, A.-C.; Galluccio, L.; Milardo, S.; Morabito, G.; Palazzo, S. SD-WISE: A Software-Defined WIreless SEnsor network. *Comput. Netw.* **2019**, *159*, 84–95. [CrossRef]
9. Wang, L.; Petrova, K.; Yang, M.L. Trust Models in Wireless Sensor Networks for Defending Against Denial-of-Service Attacks: A Literature Review. *Appl. Sci.* **2025**, *15*, 3075. [CrossRef]
10. Mohanapriya, R.; Ramu, V.; Arunachalam, K.P.; Kadu, R.K. Optimized storage and cybersecurity in blockchain-assisted wsns through malicious node detection and data enhancement using a deep spatial pyramid diffractive neural network. *Int. J. Inf. Technol.* **2026**, 1–11. [CrossRef]
11. Al Ebrahim, E.A.; Badawieh, A. Machine Learning-Based Detection and Prevention of DoS and DDoS Attacks in SDWSN. *Iraqi J. Electr. Electron. Eng.* **2026**, *22*, 114–127. [CrossRef]
12. Chu, A.; Zhang, T.; Wang, C. LEACH Protocol Evolution in WSN: A Review of Energy Consumption Optimization and Security Reinforcement. *Sensors* **2026**, *26*, 2272. [CrossRef] [PubMed]
13. D'aniello, G.; Fotia, L. Blockchain and AI-based methods for trust management in IoT: A comprehensive survey. *Internet Things* **2025**, *34*, 101755. [CrossRef]
14. Kumavat, K.S.; Gomes, J. Multi-layer DDoS attacks detection with mitigation in IoT-enabled sensor network. *Cybersecurity* **2025**, *8*, 72. [CrossRef]
15. Airehrour, D.; Gutierrez, J.A.; Ray, S.K. SecTrust-RPL: A secure trust-aware RPL routing protocol for Internet of Things. *Future Gener. Comput. Syst.* **2019**, *93*, 860–876. [CrossRef]

16. Khan, T.; Singh, K. RTM: Realistic Weight-Based Reliable Trust Model for Large Scale WSNs. *Wirel. Pers. Commun.* **2023**, *129*, 953–991. [CrossRef]
17. Bin-Yahya, M.; Alhussein, O.; Shen, X. Securing Software-Defined WSNs Communication via Trust Management. *IEEE Internet Things J.* **2022**, *9*, 22230–22245. [CrossRef]
18. Wang, R.; Zhang, Z.; Zhang, Z.; Jia, Z. ETMRM: An Energy-efficient Trust Management and Routing Mechanism for SDWSNs. *Comput. Netw.* **2018**, *139*, 119–135. [CrossRef]
19. Wang, L.; Yang, M.L.; Petrova, K. ATAW-TM: An Adaptive, Threshold-Free, and Automatically Weighted Trust Model for Mitigating Multiple Types of Denial-of-Service Attacks in Software-Defined Wireless Sensor Networks. *Electronics* **2025**, *14*, 4933. [CrossRef]
20. SDN-WISE Lab. sdn-wise-contiki. Available online: https://github.com/sdnwiselab/sdn-wise-contiki (accessed on 17 September 2025).
21. SDN-WISE Lab. sdn-wise-java. Available online: https://github.com/sdnwiselab/sdn-wise-java (accessed on 17 September 2025).
22. Wu, X.; Huang, J.; Ling, J.; Shu, L. BLTM: Beta and LQI Based Trust Model for Wireless Sensor Networks. *IEEE Access* **2019**, *7*, 43679–43690. [CrossRef]
23. Liang, X.; Gao, C.; Jing, W.; An, R. Iterated rational quadratic kernel-high-order unscented Kalman filtering algorithm for spacecraft tracking. *Def. Technol.* **2025**, *45*, 238–250. [CrossRef]
24. Simpson, E.H. Measurement of diversity. *Nature* **1949**, *163*, 68. [CrossRef]
25. Rényi, A. On measures of entropy and information. In Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, USA, 20 June–30 July 1960; Volume 1: Contributions to the Theory of Statistics, pp. 547–562.
26. Gu, D.; Hu, H. Rényi entropy based target tracking in mobile sensor networks. *IFAC Proc. Vol.* **2011**, *44*, 13558–13563. [CrossRef]
27. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 249–256.
28. Wijetunge, S. Performance Analysis of IEEE 802.15. 4 Based Wireless Sensor Networks. Ph.D. Thesis, University of Western Sydney, Penrith, Australia, 2013.
29. Feng, S.; Zhao, D. Supporting real-time CBR traffic in a cognitive radio sensor network. In Proceedings of the 2010 IEEE Wireless Communication and Networking Conference, Sydney, Australia, 18–21 April 2010; pp. 1–6.
30. Karpinski, S.; Belding, E.M.; Almeroth, K.C. Towards realistic models of wireless workload. In Proceedings of the 2007 5th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks and Workshops, Limassol, Cyprus, 16–20 April 2007; pp. 1–5.
31. Koyama, A.; Tanigawa, Y.; Tode, H. Transmission Timing Control among Both Aperiodic and Periodic Flows for Reliable Transfer by Restricted Packet Loss and within Permissible Delay in Wireless Sensor Networks. *IEICE Trans. Commun.* **2023**, *106*, 817–826. [CrossRef]
32. Dunkels, A.; Voigt, T. Contiki-a lightweight and flexible operating system for tiny networked sensors. In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, Tampa, FL, USA, 16–18 November 2004; pp. 455–462.
33. Dunkels, A.; Osterlind, F.; Tsiftes, N.; He, Z. Software-based on-line energy estimation for sensor nodes. In Proceedings of the 4th Workshop on Embedded Networked Sensors, Cork, Ireland, 25–26 June 2007; pp. 28–32.
34. Borges, L.M.; Velez, F.J.; Lebres, A.S. Survey on the characterization and classification of wireless sensor network applications. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1860–1890. [CrossRef]
35. Sathish, K.; Ravikumar, C.V.; Srinivasulu, A.; Gupta, A.K. Performance Analysis of Underwater Wireless Sensor Network by Deploying FTP, CBR, and VBR as Applications. *J. Comput. Netw. Commun.* **2022**, *2022*, 7143707:7143701–7143707:7143730. [CrossRef]
36. Pandey, U.; Verma, R.K.; Chaurasia, B.K. HTBIDS: A hybrid trust and feature fusion-based framework for robust intrusion detection in adversarial wireless sensor networks. *Ad Hoc Netw.* **2026**, *192*, 104334. [CrossRef]
37. Kumar, N.; Sangal, A.L. Evolution of ML and AI Techniques for DDoS Detection and Mitigation in SDN: A Comprehensive Review. In Proceedings of the 2026 International Conference on Intelligent Processing, Hardware, Electronics, and Radio Systems (CIPHER), Jalandhar, India, 13–15 February 2026; pp. 1–7.
38. Schaust, S.; Drozda, M.; Szczerbicka, H. Impact of packet injection models on misbehaviour detection performance in wireless sensor networks. In Proceedings of the 2007 IEEE International Conference on Mobile Adhoc and Sensor Systems, Pisa, Italy, 8–11 October 2007; pp. 1–6.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.