Mathematical Biosciences and Engineering

MBE, 21(2): 3335–3363.
DOI: 10.3934/mbe.2024148
Received: 04 December 2023
Revised: 13 January 2024
Accepted: 16 January 2024
Published: 04 February 2024

http://www.aimspress.com/journal/MBE

Research article

Secure multi-path routing for Internet of Things based on trust evaluation

Jingxu Xiao[1,*], Chaowen Chang[1], Yingying Ma[2], Chenli Yang[1] and Lu Yuan[1]

[1] Information Engineering University of the Army Strategic Support Force, Zhengzhou 450004, China
[2] Zhengzhou University of Technology, Zhengzhou, Henan, 450004, China

* **Correspondence:** Email: xiaojingxu2301@163.com.

Abstract: In the realm of the Internet of Things (IoT), ensuring the security of communication links and evaluating the safety of nodes within these links remains a significant challenge. The continuous threat of anomalous links, harboring malicious switch nodes, poses risks to data transmission between edge nodes and between edge nodes and cloud data centers. To address this critical issue, we propose a novel trust evaluation based secure multi-path routing (TESM) approach for IoT. Leveraging the software-defined networking (SDN) architecture in the data transmission process between edge nodes, TESM incorporates a controller comprising a security verification module, a multi-path routing module, and an anomaly handling module. The security verification module ensures the ongoing security validation of data packets, deriving trust scores for nodes. Subsequently, the multi-path routing module employs multi-objective reinforcement learning to dynamically generate secure multiple paths based on node trust scores. The anomaly handling module is tasked with handling malicious switch nodes and anomalous paths. Our proposed solution is validated through simulation using the Ryu controller and P4 switches in an SDN environment constructed with Mininet. The results affirm that TESM excels in achieving secure data forwarding, malicious node localization, and the secure selection and updating of transmission paths. Notably, TESM introduces a minimal 12.4% additional forwarding delay and a 5.46% throughput loss compared to traditional networks, establishing itself as a lightweight yet robust IoT security defense solution.

Keywords: IoT security; software-defined networking; packet security verification; multi-path routing; path scoring

1. Introduction

The Internet of Things (IoT) refers to the interconnection of physical devices to a network through information-sensing devices, creating a comprehensive network system that enables seamless communication among people, devices, and things at any time and any location [1]. IoT technology digitizes the physical world, effectively integrates decentralized information resources, and facilitates digital information interaction between things. Today, IoT has found widespread applications in various fields such as industrial manufacturing, remote healthcare, smart homes, and connected vehicles [2]. Predictions indicate that by 2025, approximately 750 billion devices will be connected to the IoT, playing a crucial role in people's lives and work. However, despite the convenience IoT brings, it also faces increasingly severe security challenges [3].

The application of edge computing in the IoT effectively addresses issues such as data transmission latency and bandwidth consumption caused by the distance of cloud data centers from IoT terminal devices. Edge computing deploys computing, storage, and network services closer to the data source, utilizing edge nodes such as edge computing nodes and fog computing nodes (this paper considers fog computing as an extension of edge computing). By executing local computing tasks, edge nodes only send essential data to the cloud, effectively reducing data transmission latency and improving real-time capabilities [4]. Edge nodes ensure the security of incoming IoT data by establishing access control mechanisms or authenticating devices accessing the IoT [5]. Through data encryption and security verification, the confidentiality and integrity of data transmission can be ensured [6]. However, these solutions often overlook security threats in communication links between edge nodes and cloud servers or in scenarios of data transmission between edge nodes in multi-access edge computing (MEC) [7]. The presence of malicious nodes in the link can result in actions such as data theft, tampering, or deletion, impacting the secure transmission of IoT data [8]. While commonly used end-to-end data security verification solutions in the IoT can identify abnormal data, they cannot locate abnormal links and malicious nodes in the link, allowing malicious nodes to continuously threaten the secure transmission of data. Additionally, if path anomalies occur and are not promptly detected and replaced, it can lead to the malfunction of IoT devices and widespread network failures. Therefore, the key to ensuring the secure and stable transmission of IoT data lies in the implementation of the selection of secure transmission paths between edge nodes and the cloud, the localization of malicious nodes, and the detection and replacement of abnormal links.

Software-defined networking (SDN) [9] is a technology that separates the control plane from the data plane in a network, with the core concept of enabling flexible configuration and management through software. Javanmardi et al. [10] summarized the application architecture of SDN in the IoT fog networks and the SDN-based security defense mechanism for IoT-fog. They designated fog nodes and fog gateways as the data plane of SDN, while cloud data centers and cloud gateways constituted the SDN control plane. Leveraging the centralized management features of SDN, the control plane facilitates the observation and monitoring of all nodes in the network. Fog gateways process all flows using flow rules injected by the cloud gateway, enabling flexible management of network configurations and enhancing the operational and security aspects of IoT-fog. SDN, with its programmability and centralized management of network nodes and data flows, effectively addresses the challenges in IoT security defense mechanisms, thereby improving the performance of security defense solutions. For instance, it facilitates the management and secure access control of a large number of IoT devices, enables easy acquisition of IoT traffic for anomaly detection, and provides an integrated management environment for password-based security solutions. However, existing

security solutions that integrate SDN with IoT lack research on secure data transmission paths [11]. Therefore, investigating how SDN can be utilized to achieve the selection of secure routing in the IoT is a worthwhile research direction.

To address the security threats faced by data during transmission between edge nodes and the cloud, as well as between edge nodes, and to ensure the security of data transmission paths, we propose a trust evaluation based secure multi-path routing scheme for the IoT. By applying SDN architecture to IoT, we incorporate the "distrust and always verify" principle from zero trust [12]. Through the continuous security verification of data packets, we assess the trustworthiness of links and nodes within the links. Based on these evaluations, we establish a secure and reliable multi-path routing mechanism, enabling the selection and dynamic updating of secure paths. This ensures the secure and stable transmission of data across the network links. The following are the main contributions of this paper:

1) Proposing a trust evaluation based secure multi-path routing scheme for IoT, this approach achieves the detection of abnormal links through secure validation of data packets. Real-time trust scoring of switch nodes enables secure multi-path selection and the localization of malicious nodes. Finally, by devising dynamically adjustable multi-path routing policies, the method ensures flexible, secure, and stable data transmission within the IoT.

2) Designing a reinforcement learning-based dynamic multipath routing algorithm, using node trust scores as evaluation parameters for multi-objective reinforcement learning. The algorithm adjusts the real-time generation strategy of multipath routing based on path scores, achieving the dynamic generation and updating of secure multipaths.

3) Validating the effectiveness and superiority of the proposed solution and algorithm through simulation experiments. The results indicate that the proposed methods can effectively enhance the security and reliability of the IoT, with minimal impact on performance overhead.

The structure of this paper is organized as follows: Section 2 reviews the research schemes related to this work. Section 3 provides a detailed introduction to the proposed trust evaluation based secure multi-path routing scheme for IoT. Section 4 conducts simulations of the proposed scheme and analyzes the results. Section 5 summarizes the main content of this paper and outlines future research directions.

2. Related works

The related solutions are built upon SDN to achieve secure protection and anomaly detection for IoT data traffic. Kamoun-Abid et al. [13] established a firewall structure combining the IoT cloud layer and fog layer. Due to the proximity of fog nodes to IoT devices, they efficiently filter traffic based on rules, effectively reducing the hops of illegal data packets in the IoT. Sadiq et al. [14] integrated SDN to build a firewall for defending against distributed denial of service (DDoS) attacks in the IoT. While the firewall incurs lower costs, it is ineffective against internal attacks and 0-day attacks. Dhawan et al. [15] introduced the Sphinx scheme, obtaining traffic information from switches to create specific flow information flowcharts in the network. They effectively monitor network traffic based on the features of these flowcharts. Nguyen et al. [16] used a distributed controller to obtain IoT traffic information, alleviating the issue of a single point of failure. They combined deep learning to construct three different levels of intrusion detection systems to detect anomalous traffic. Pourvahab et al. [17] utilized blockchain for forensic analysis of data in the IoT, ensuring device authentication and data integrity. Additionally, each SDN controller incorporates a classifier for recognizing malicious packets and devices using a multi-fuzzy neural network algorithm. However, these anomaly detection solutions require real-time monitoring and statistics of IoT traffic, leading to increased network overhead.

Moreover, anomaly detection may introduce some latency, posing challenges in real-time blocking of malicious traffic attacks.

The related solutions integrate SDN with cryptography, ensuring the security of data transmission and the reliability of links through secure verification of data packets in the context of the IoT. The LPV, Lightweight Packet Forwarding Verification, [18] scheme adds externally transparent labels to data packets in the network. Ingress and egress switches calculate the summary information for packets with special labels and send it to the controller. Through the verification of the summary information, the security of data transmission is ensured. Xie et al. [19] applied blockchain technology to vehicular networks, where each vehicle contains road information. Surrounding vehicles score this information for its authenticity, ensuring information accuracy. Li et al. [20] constructed a blockchain-based IoT zero-trust architecture. This architecture establishes an identity authentication model between edge computing nodes and IoT devices. Through blockchain, data and behaviors are recorded, achieving zero-trust verification and dynamic authorization for user behavior. Although blockchain can authenticate IoT data, it generates a large amount of information exchange between nodes and is not suitable for certain IoT environments with high real-time communication requirements. P4Label [21] is a P4-based packet forwarding verification mechanism. It establishes identity-based signatures for data packets, detecting malicious tampering or forgery through signature verification. However, P4Label adds a longer header length to the data packets and incurs significant key storage overhead. The SDNsec [22] scheme encodes packets with information about the transmission path. Switches in the network verify the encoding, and if the packet violates path rules, the switch discards it, ensuring the consistency of the forwarding path. However, this scheme cannot locate malicious nodes. Latif et al. [23] integrated blockchain with SDN and proposed a clustered structure for the IoT network routing protocol. By constructing multiple SDN domains and establishing both a public blockchain and private blockchains responsible for inter-domain and intra-domain data security, this solution is designed to be managed by the SDN controller and maintained by each IoT device within the cluster. However, this approach only achieves secure verification between IoT data points and does not address the security of data transmission paths between two points. Zeng et al. [24] employed blockchain to protect secure routing in SDN-enabled multi-controller IoT networks. The SDN controller abstracts the inter-domain topology structure and uploads it to smart contracts, which are then verified by other controllers based on a voting mechanism to ensure secure routing between multiple domains. Nevertheless, this solution only guarantees the security and trustworthiness of controllers and network topology and does not effectively detect or locate malicious switch nodes in the data plane.

In the IoT, the selection of transmission paths is crucial. Strategies such as defining path selection, updates, and replacements can effectively enhance network service quality, data transmission security, and resilience to risks. Yan et al. [25] proposed an SDN-based quality of service (QoS) assurance scheme that utilizes queuing mechanisms and multipath routing technology to ensure the QoS of different types of traffic. The scheme can quickly provide replacement paths when faulty paths occur. Alqahtani et al. [26] introduced a multipath QoS routing protocol based on route stability. The protocol generates two optimal paths between source and destination nodes using different route metrics. It uses both the primary and backup paths for data transmission to improve network stability. Multipath quick UDP Internet connections (MPQUIC) [27] is a routing protocol that supports multipath transmission. In comparison to the multi-path TCP (MPTCP) protocol, MPQUIC employs dynamic path selection strategies to choose the best path in real-time based on network conditions and data transmission requirements. However, MPQUIC, using the UDP protocol, cannot address security threats such as IP spoofing and tampering of packets. Pu et al. [28] proposed an interference-resistant multipath routing protocol called JarmRout. The protocol generates multiple paths based on link quality, traffic load, and

spatial distance, addressing the impact of local failures on data transmission in unmanned aerial vehicle self-organizing networks and improving network resilience. Jin et al. [29] established a resilient and secure microgrid, using SDN for visualized supervisory control of the communication network. In the face of network attack threats, the system achieves self-healing management by generating secure alternative paths. Li et al. [30] presented a secure and reliable data transmission solution in industrial IoT. A centralized controller periodically analyzes data packet validation reports sent by terminal hosts to identify malicious switch nodes involved in packet modification attacks. Upon discovering a malicious switch node, the controller selects a new secure path through route algorithm updates for communication restoration. However, this scheme requires obtaining validation information from all terminals, imposing a significant burden on the controller. Ren et al. [31] proposed an endogenous security SDN network architecture based on multipath elastic routing (MRR). The architecture offers three data forwarding modes: multipath comparison, multipath weighting, and multipath random. In the multipath comparison mode, security verification of paths is achieved by comparing the consistency of data. In the multipath weighting mode, load balancing is achieved by assigning weights to paths. In the multipath random mode, low-cost paths are randomly selected for data transmission. However, MRR can only detect anomalous nodes in the multipath comparison forwarding mode and other modes do not include the security verification process for paths. Guo et al. [32] proposed a QoS-aware secure routing protocol, DQSP, based on deep reinforcement learning to address the vulnerability of routing protocols in SDN-IoT. DQSP achieves self-learning through interactions with the environment, ensuring QoS while avoiding routing plans targeting malicious nodes. However, DQSP only considers gray hole attacks and DDoS attacks during the implementation of secure routing. In cases of abnormal behavior such as data tampering and unusual forwarding paths in the switch nodes, DQSP cannot provide security functionality.

Table 1. Functional comparison of relevant solutions.

Scheme	Data security	Anomaly localization	Path selection	Path security
Scheme [13–17,19–21,23]	√	×	×	×
Scheme [18]	√	√	×	×
Scheme [22]	√	×	×	√
Scheme [24]	×	×	×	√
Scheme [25-28]	×	×	√	×
Scheme [29,30]	√	√	×	√
Scheme [31]	√	√	√	×
Scheme [32]	√	×	√	×
Proposed scheme	√	√	√	√

Table 1 compares the functionalities implemented by our proposed solution with those of related approaches. In the table, √ indicates the presence of the corresponding functionality, while × indicates the absence of that functionality. The shortcomings identified in the above-mentioned research include the separation of studies on data security and path security, lack of consideration for security factors in the process of selecting data transmission paths, and the absence of effective means to handle abnormal links and promptly switch to secure transmission paths. Therefore, there is a pressing need for an IoT routing selection and updating strategy that comprehensively addresses both data and path security verification, dynamically generates secure paths based on the security status of

the links, and facilitates effective handling of abnormal links and timely replacement of secure transmission paths.

3. Trust evaluation based secure multi-path routing for IoT

This section will provide a detailed exploration of the trust evaluation based secure multi-path routing for IoT (TESM), covering aspects such as model architecture, problem statement, packet security verification, secure multi-path routing, and anomaly handling mechanisms. To enhance clarity in describing the solution, definitions for the identifiers presented in Table 2 are provided.

Table 2. Definition of identifiers.

Identifier	Definition
P	Forwarded data packet
H	Packet header
PL	Packet payload
FlowID	Data flow identifier
S_i	Switch node identifier
count_in	Incoming packet count
count_out	Outgoing packet count
$score\_d_{S_i}$	Trust score based on data of S_i
$score\_p_{S_i}$	Trust score based on path of S_i
K_i	Session key between controller and S_i
$H_1()$,$H_2()$,$H_3()$	Hash functions
$MAC_K()$	Message authentication code generation function based on key K

3.1. TESM architecture

To address security threats in data transmission between edge nodes and between edge nodes and the cloud data center, we propose a secure multi-path routing solution based on trust evaluation, TESM. TESM deploys an SDN controller to manage data forwarding between IoT edge nodes or between edge nodes and the cloud data center. In the following description, we will focus on the scenario where data is transmitted between edge nodes. In this context, network devices with data forwarding capabilities in the IoT are defined as switching nodes. TESM transforms the switch nodes along the data transmission links between edge nodes into SDN-enabled entities capable of communicating with the controller. Comprising security verification, multi-path routing, and anomaly handling modules, TESM ensures secure data validation, detection and localization of malicious nodes, and dynamic selection of secure multiple paths through collaborative interactions among these modules. This enhances the security and stability of data transmission between edge nodes.

The architecture of TESM is illustrated in Figure 1, where SN and DN denote edge nodes and S_1~S_6 represent the switch nodes within the link. The security verification module, multi-path routing module, and anomaly handling module are positioned within the controller. Below, we provide individual descriptions of the functions of each module.

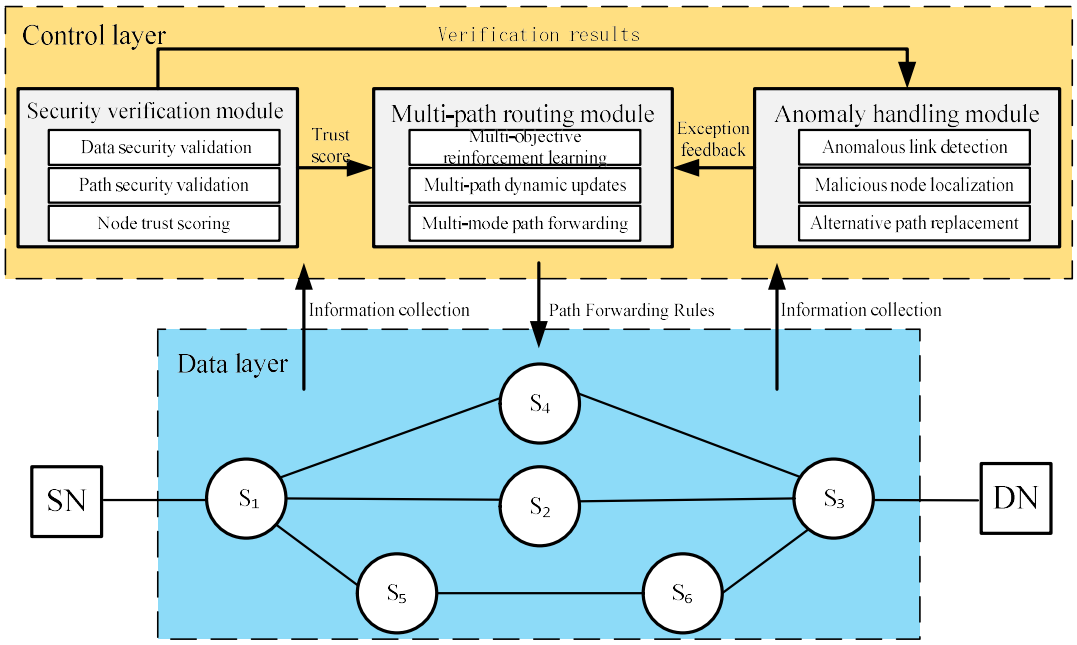


Figure 1. TESM architecture diagram.

The security verification module performs real-time monitoring of data traffic within the data layer and conducts secure validation, ensuring the safety and integrity of data transmission. Simultaneously, it gathers statistics on the packet forwarding activity of relevant ports on switching devices, enabling the verification of paths and nodes. This process guarantees the consistency of forwarding paths and the trustworthiness of nodes. Based on the verification outcomes, the security verification module assigns trust scores to nodes, serving as a basis for path selection in the multi-path routing module. Additionally, the security verification module forwards information about paths and nodes that did not pass the security validation to the anomaly handling module for further investigation and resolution.

The multi-path routing module constructs secure multi-path routing based on real-time network topology information, incorporating bandwidth, latency, and node trust scores. Using multi-objective reinforcement learning, the controller deploys multi-path forwarding rules to switch nodes, thereby enhancing the quality of service and security of IoT. This module dynamically adjusts paths in response to changes in node status, ensuring the dynamic updating of forwarding paths. It also adapts data forwarding rules based on network conditions, thereby improving IoT stability and network resilience.

The anomaly handling module detects and isolates paths that fail security validation through a more granular security verification method for the links associated with abnormal nodes. It updates paths in the multi-path routing module, distributing suspicious abnormal nodes across different paths to enhance the efficiency of anomaly detection. Using a trust score mechanism, the anomaly handling module locates and removes abnormal nodes from the multi-path routing, ensuring secure and trustworthy paths. In the event of detecting abnormal links, the module replaces the abnormal path with a secure alternative path, promptly deploying updated flow rules to enhance network self-healing and achieve self-recovery in the face of security threats in the IoT.

The implementation of TESM relies on the following security assumptions: the SDN controller is secure, the process through which the controller issues rules to switching devices is secure, and the boundary switch nodes are secure and trustworthy.

3.2. Problem statement

In the deployed IoT with an SDN architecture, the controller facilitates data forwarding along specified paths by deploying flow tables to switch nodes. The secure transmission of data relies on the security of the chosen paths. The presence of malicious nodes in the path can lead to abnormalities in both data and paths, posing a threat to the secure transmission of data between edge nodes. Figure 2 illustrates the security threats posed by malicious nodes to data transmission. Here, S_1–S_6 represent switch nodes and data is transmitted from the source edge node SN to the destination edge node DN, following the designated path S_1-S_2-S_3. In this scenario, data flow is sequentially forwarded through S_1, S_2, and S_3. Assuming S_2 is a malicious node, it possesses the capability to alter both the data content and the rules for forwarding data packets.

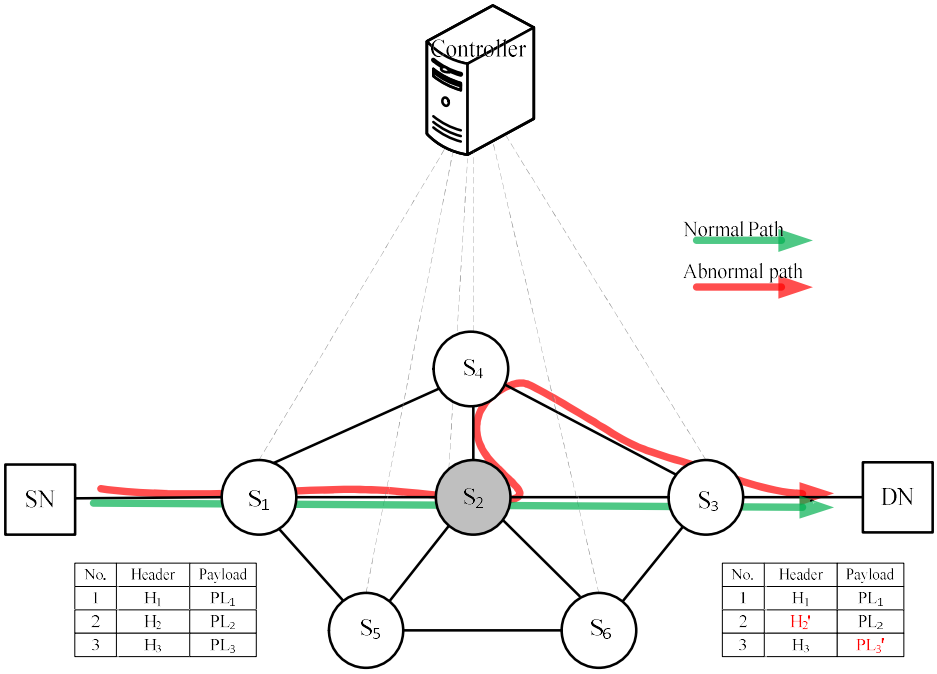


Figure 2. Security threats of malicious nodes to data transmission.

Malicious packet modification attacks involve altering the content of data packets to achieve objectives such as theft, attacks, and destruction. In Figure 2, the malicious node S_2, through header manipulation ($H_2 \rightarrow H_2'$), can render security defense systems ineffective in detecting and tracing the illicit data packet. Tampering with matching items can lead to security threats such as masquerade attacks, denial of service attacks, and man-in-the-middle attacks. By tampering with the payload of data packets ($PL_3 \rightarrow PL_3'$), malicious nodes deceive the target nodes, especially when the manipulated content includes malicious code, leading to security threats like device control, information theft, and expanding the scope of IoT infections.

Anomalies in the forwarding path involve modifying the data's forwarding path to intercept data, bypass protective measures, or change the destination address. In Figure 2, the malicious node S_2, contrary to the rules, forwards the data packet to S_4 before transmitting it to S_2. S_4, through operations like copying or stealing data packets, compromises the security of IoT data, leading to data leakage.

The security of the transmission path is crucial for enhancing data security and network trustworthiness. Figure 3 illustrates multiple forwarding paths in the network, with S_1-S_2-S_3 as the designated path and S_4 as a potential malicious node. When the S_1-S_2-S_3 path encounters a failure, an alternative path needs to be selected. Considering factors such as bandwidth and latency, and assuming equal performance at each node, the S_1-S_4-S_3 path might be considered the optimal alternative path due to fewer switch hops, enabling faster data transmission. However, since S_4 has a higher likelihood of being a malicious node, data transmission along this path poses a higher security risk. Therefore, combining security and trustworthy analysis of nodes, the S_1-S_5-S_6-S_3 path emerges as the most suitable alternative, effectively enhancing the security and reliability of data transmission. The same conclusion can be reached in the case of using multipath routing. If two paths are selected for data transmission, ensuring secure forwarding involves choosing the S_1-S_2-S_3 and S_1-S_5-S_6-S_3 paths, avoiding the selection of shorter but less secure paths like S_1-S_4-S_3.

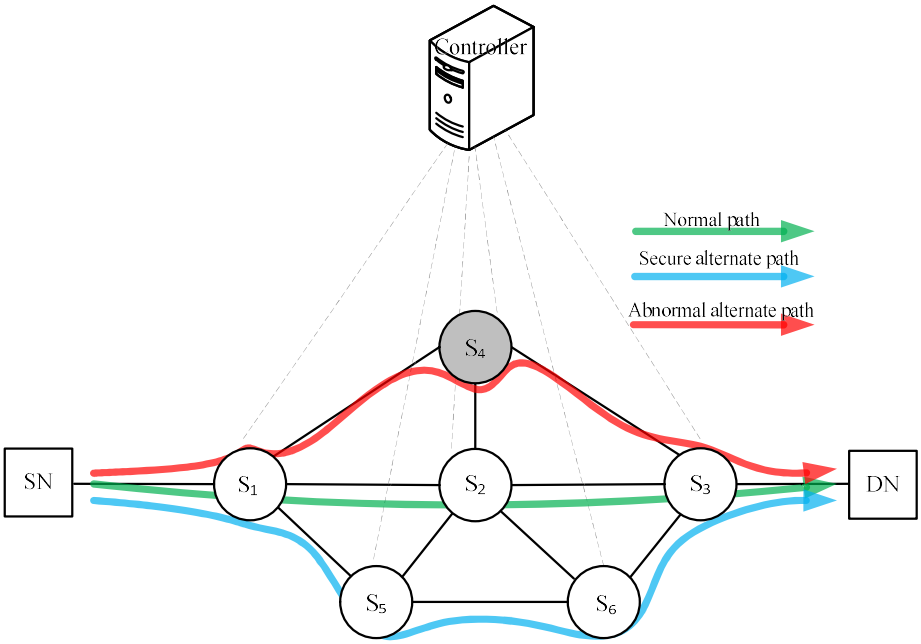


Figure 3. Selection of secure forwarding paths.

To address the aforementioned security threats and achieve secure data transmission between edge nodes, we aim for TESM to provide comprehensive security assurance for IoT data, nodes, and paths. Specifically, TESM must continuously verify the security of data and forwarding paths to ensure data integrity and path security. Additionally, TESM should perform real-time detection and localization of malicious nodes within the links. Finally, TESM should facilitate the selection of secure paths, enabling timely path updates when abnormal paths are detected to enhance the self-healing capability of the IoT.

3.3. Data packet security verification

In the TESM architecture, the security verification module performs security checks on data packets to detect anomalies and verify the integrity of the links. Based on the verification results, it

generates trust scores for the switch nodes in the transmission path. The security verification process is illustrated in Figure 4.

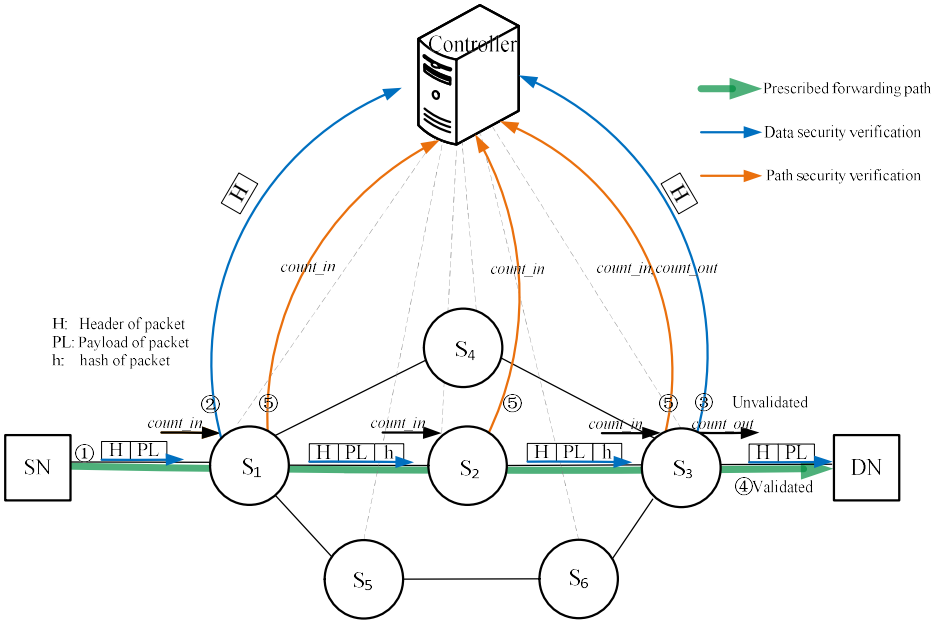


Figure 4. Data packet security verification flowchart.

1) The source node transmits data to the ingress node S_1.

2) If S_1 receives this type of data packet for the first time, it transmits the packet header (H) to the controller. The controller, based on the header information, formulates forwarding rules for the data packet. Subsequently, S_1 calculates the hash value (h) of the data packet using Eq (1) and forwards h along with the data packet. The hash function $H_1()$ is exclusively shared between the controller and the boundary switch nodes to prevent tampering of both P and h by intermediate switch nodes simultaneously.

$$h=H_1(H||PL) \tag{1}$$

3) Egress switch node S_3 calculates the hash value of the received data packet using Eq (1) and compares it with the hash value (h) within the packet. If the values are not equal, indicating that the data packet has not passed security validation (i.e., the data content has been anomalously modified), S_3 refrains from forwarding the packet and sends the packet header (H) to the controller to report the anomaly. When the calculated hash value matches the value within the data packet, it signifies the successful security validation of the data packet.

4) For data packets that pass security validation, egress switch node S_3 removes h and forwards the data packet to the destination node.

5) Each switch node along the forwarding path counts the packets flowing through, with *count_in* and *count_out* representing the number of incoming and outgoing packets at each switch node. Under the assumption of no natural packet loss in the links, the counts at each switch node along the same forwarding path should be equal. Any disparity indicates an anomaly in the forwarding path at that

specific switch node. Switch nodes periodically upload these statistical values to the controller, enabling trust score calculation for nodes and detection of anomalous nodes.

Node trust score is divided into two components: data-based trust score and path-based trust score. FlowID is employed to describe the type of data flow, and it is included in the data packet header, as depicted in Eq (2). Here, srcIP and dstIP refer to the source IP address and destination IP address from the original IP header of the data packet, while srcport and dstport represent the source and destination ports. The padding section can be customized by adding transmission information such as data business type and data priority, facilitating a fine-grained division of data flows. TESM specifically addresses data flows between edge nodes transmitted using the TCP/IP protocol. In cases where data between edge nodes employs alternative protocols, adjustments to Eq (2) should be made based on the specific protocol details. However, this aspect is not discussed further in this paper.

$$\text{FlowID} = \text{H}_2(\text{srcIP}||\text{dstIP} \,||\, \text{scrport}||\text{dstport}||\text{padding(optional)}) \tag{2}$$

If data packets with the same FlowID have the same forwarding path, then $count\_in_{S_i}^{\text{FlowID}}$ and $count\_out_{S_i}^{\text{FlowID}}$ respectively represent the statistics of the number of incoming and outgoing data packets of S_i for the FlowID type. The ratio of the number of data packets entering downstream exchange node S_{i+1} to the number of data packets entering upstream exchange node S_i in the specified forwarding path is calculated to describe the trust score of the node based on the path, as shown in Eq (3).

$$score\_p_{S_i}^{\text{FlowID}} = \frac{count\_in_{S_{i+1}}^{\text{FlowID}}}{count\_in_{S_i}^{\text{FlowID}}} \cdot 100\% \tag{3}$$

The trust score of the node based on data in the forwarding path where FlowID is located is shown in Eq (4).

$$score\_d_{S_i} = score\_d_{S_i}^{\text{FlowID}} = \frac{count\_out_{S_{out}}^{\text{FlowID}} + \Delta}{count\_in_{S_{in}}^{\text{FlowID}}} \cdot 100\% \tag{4}$$

S_{in} and S_{out} respectively represent the ingress and egress exchange nodes. When the ratio of the number of data packets flowing out of S_{out} to the number of data packets flowing into S_{out} is smaller, it indicates that there are more abnormal data packets and malicious exchange nodes in the path. At this point, the exchange nodes in the path receive a lower trust score. To eliminate the impact of abnormal forwarding paths or malicious discards on data trust scores, a compensation parameter Δ is added. The calculation method is shown in Eq (5). By supplementing the number of data packets that each exchange node in the forwarding path did not forward normally, the independence between the data-based trust score and the path-based trust score is ensured.

$$\Delta = \sum (1 - score\_p_{S_i}^{\text{FlowID}}) \cdot count\_in_{S_i}^{\text{FlowID}} \tag{5}$$

Equation (4) only considers the case where S_i forwards data on a single path. The appearance of malicious exchange nodes will cause the trust score of all nodes in the same forwarding path to decrease, and S_i in the same path has the same $score\_d_{S_i}$. To distinguish between malicious exchange

nodes and secure exchange nodes, TESM constructs multi-path routing to make S_i obtain different $score\_d_{S_i}^{FlowID}$ for different FlowID. At this point, the trust score of S_i is shown in Eq (6).

$$score\_d_{S_i} = w_d \cdot score\_d_{S_i}^0 + (1 - w_d) \cdot score\_d_{S_i}^{FlowID} \tag{6}$$

where w_d is the weight parameter and $score\_d_{S_i}^0$ is the trust score of S_i stored by the controller. Equation (6) can keep the trust score of malicious exchange nodes stably low while gradually increasing the trust score of normal nodes affected by malicious nodes in Eq (4). Similarly, $score\_p_{S_i}^{FlowID}$ in multiple forwarding paths is also updated using Eq (6).

3.4. Multipath secure routing

The multi-path routing module employs a multi-objective reinforcement learning approach to construct multiple secure paths for TESM. Multi-objective reinforcement learning is a type of reinforcement learning algorithm where an agent learns and discovers the optimal strategy by interacting with the environment to balance multiple objectives. In this context, the multi-path routing module is defined as an intelligent agent utilizing multi-objective reinforcement learning. Each switch node corresponds to a state in the reinforcement learning framework, and the process of moving from the current switch node to an adjacent switch node represents an action in the reinforcement learning context. Through continuous learning of the path selection strategy, the multi-path routing module ultimately acquires multiple secure paths between the source and destination nodes.

Algorithm 1 describes the process of generating secure multiple paths. Firstly, the topological structure of the network nodes is obtained and stored in a graph structure. Simultaneously, trust scores for each switch node are obtained from the security verification module. The notation Q is defined to represent the table storing the state-action value function, while *Path* and *Path_score* are used to store the generated multiple paths and their corresponding path scores. The algorithm initializes the source node S_s as the initial state. The choose_action() function employs an epsilon-greedy algorithm to select the next action in the current state. This involves randomly selecting the next action with a probability of e and choosing the action that yields a higher Q[(state, action)] in other cases. The *reward* signifies the real-time reward obtained after transitioning from the current switch node to an adjacent switch node. The reward() function is a linear combination of node trust scores and the number of path transitions. The weights w_1 to w_3 represent the respective weights for data trust scores, path trust scores, and the number of path transitions, as illustrated in Eq (7).

$$reward = w_1 \cdot score\_d + w_2 \cdot score\_p - w_3 \cdot 1 \tag{7}$$

The value function is a function used to assess the long-term returns for each pair of state-action. Continuous updates to the value function allow for the derivation of an optimal action policy for each switch node under the given objective. The update_q() function employs the iterative approximation of the optimal value function based on the Bellman equation in Q-learning [33]. Here, $\max(Q[(state', action')])$ represents the maximum value in all value functions for the next state, α is the learning rate that determines the convergence speed of Q, and γ is the discount factor

determining the importance of future rewards. The update_q() function updates Q[(state, action)] according to Eq (8).

$$Q[(state, action)] = Q[(state, action)] + \alpha[reward + \gamma \max(Q[(state', action')]) - Q[(state, action)]] \quad (8)$$

Subsequently, a new state is obtained, and the Q-table is continuously updated based on the updated state. The exploration process for the current path concludes when the destination node is reached or the path length exceeds the upper limit. The state is then restored to the initial state S_s, initiating a new round of the learning process. This cycle continues until the training iteration is reached, resulting in the acquisition and storage of the Q-table (Algorithm 1, lines 2–12).

Algorithm 1. Secure multipath routing generation.

Input: Network topology G, Source node S_s, Destination node S_D, Trust score of S_i
Output: Multipath routing *Path*, Multipath score *Path_score*
1.Begin
2. Q=[],*Path*=[],*Path_reward*=[],i=0
3. While not reach roop_num1:
4. *state*=S_s
5. while *state* !=S_D :
6. *action*= choose_action (*state*,actions[*state*],Q)
7. *reward* = reward (*state*)
8. Q[(*state*,*action*)]= update_q (*reward*, *state*, Q)
9. *state*=*state* + *action*
10. If *state* =S_D or overstep length:
11. break
12. save(Q)
13. While i < multipath_conut:
14. While not reach roop_num2:
15. *path* = choose_path(S_s,S_D,Q, *Path*)
16. *path _ score* = reward_path(*reward*, *path*)
17. *Path*[i] = max(*path*, *Path*[i])
18. *Path_score*[i] = max(*path _ reward*, *Path_score*[i])
19. Q = prun (Q, *Path*[i])
20. return *Path*,*Path_score*
21.End

Next, the multi-path routing module selects multiple paths based on the Q-table. Starting from the state S_s, the Q-table now contains the value functions obtained through reinforcement learning. To avoid local optima, the choose_path() function utilizes an epsilon-greedy algorithm to obtain multiple paths. The reward_path() function is employed to calculate the value score of the paths, as illustrated in Eq (9). Here, $reward\_node_i$ represents the reward for the switch nodes in the generated path, and the value score of the path is the sum of rewards for all nodes in the path.

$$path\_score = \sum reward\_node_i \quad (9)$$

The highest-scoring path is selected as the secure primary path. To facilitate multi-path data transmission and ensure alternative paths in case of primary path failure, the selected paths should ideally be as disjoint as possible. The prun() function, based on the chosen paths, updates the Q-table by reducing the Q-values for nodes in the selected paths, thereby avoiding or minimizing the occurrence of duplicate nodes between multiple paths. This process of selecting multiple paths is repeated, ultimately returning multiple secure paths along with their corresponding scores (Algorithm 1, lines 13–20).

The multi-path routing module, through real-time monitoring of network and node states, provides dynamically updated secure multi-path routes for data transmission, enhancing the security of data transfer. Additionally, the module can flexibly choose data transmission schemes based on network conditions. For instance, during low network loads or when low latency is crucial, data can be forwarded through a single path. On the other hand, during high network loads or when increased reliability and bandwidth utilization are required, the module can opt for multi-path forwarding. Moreover, in the event of anomalies in the current forwarding path, the multi-path routing module promptly retrieves alternative paths from the set of multiple paths, ensuring the timely recovery of secure data transmission and effectively enhancing the network's resilience to risks.

3.5. Abnormal handling and location

When the anomaly handling module in TESM receives abnormal information from the security verification module, indicating the presence of an anomaly in the current forwarding path, the anomaly handling module selects a secure alternative path for data forwarding while simultaneously locating the anomaly in the exceptional link.

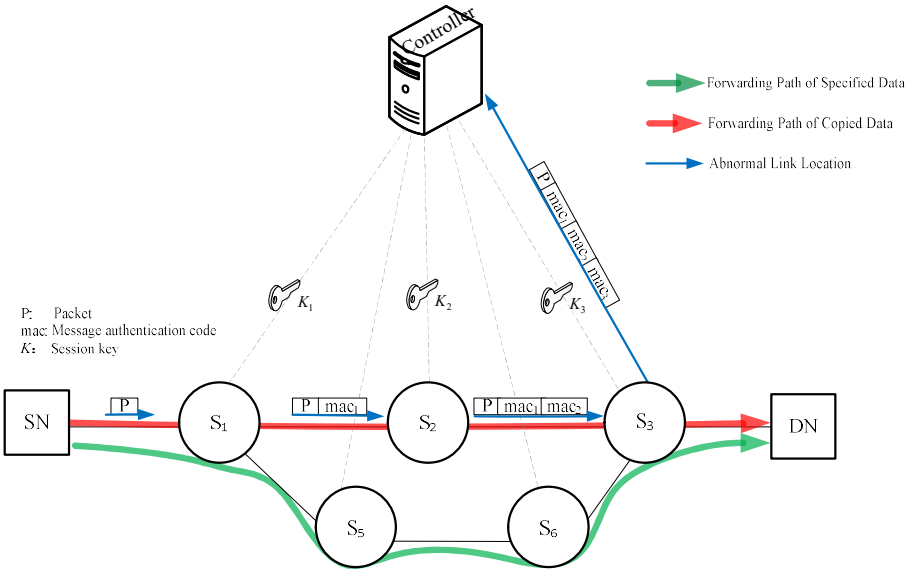


Figure 5. Anomaly handling and location process.

Figure 5 illustrates the process of anomaly handling and localization. In the figure, S_1-S_2-S_3 represents the anomalous path, and S_1-S_5-S_6-S_3 represents the secure alternative path selected from the

secure multi-path routing. The anomaly handling module, through the controller, updates the forwarding rules for the switch nodes in both paths. The source node S_1 duplicates and forwards the data. Nodes in the secure alternative path perform secure verification on the data, as described in Section 3.3. Nodes in the anomalous path calculate the message authentication code (MAC) of the data packet to facilitate the localization process of the anomaly. The controller establishes session keys with nodes on the anomalous path and the secure alternative path. Switch nodes use session keys to generate MACs for the received data. The MAC verifies the integrity of the data packet. The calculation of the MAC for each node is expressed in Eq (10), where $\mathrm{MAC}_K()$ represents a hash-based message authentication code, generating a fixed-length MAC from the message, and H(P) is the hash value of data packet P. Embedding the MAC generated by the upstream node into the calculation of MAC by downstream nodes effectively prevents malicious switch nodes from causing misjudgments of anomaly links by tampering with the MAC generated by upstream switch nodes.

$$\begin{aligned} &\mathrm{mac}_1 = \mathrm{MAC}_{K_1}(\mathrm{H}_3(P)) \\ &\mathrm{mac}_2 = \mathrm{MAC}_{K_2}(\mathrm{H}_3(P)||\mathrm{mac}_1) \\ &... \\ &\mathrm{mac}_i = \mathrm{MAC}_{K_i}(\mathrm{H}_3(P)||\mathrm{mac}_1||\mathrm{mac}_2||...||\mathrm{mac}_i) \end{aligned} \tag{10}$$

The generated mac_i is forwarded with the data packet. After calculating the MAC, the egress switch node sends the data packet and all mac_i to the controller. The controller uses the session key K_i between the nodes to verify mac_i one by one through Eq (10). When all mac_i pass the verification, it indicates that the abnormal path has not tampered with the data packet P at the moment and the detection and positioning process of the abnormal link on this path needs to be continued. When the controller finds that mac_t fails the verification, the link S_{t-1}-S_t between the node S_t corresponding to mac_t and its upstream node S_{t-1} is the abnormal link. Both S_{t-1} and S_t in S_{t-1}-S_t may be malicious nodes, that is, it may be that the malicious node S_{t-1} tampered with P, causing mact to fail the verification, or S_t may be a malicious node. S_t changes the correct mact to deceive the controller into thinking that S_{t-1} is a malicious node. This scheme believes that malicious nodes do not have the ability to modify other nodes' mac_i, but this special situation can be solved by adding digital signatures to mac_i and other methods. This article does not delve into this comparison.

Algorithm 2 provides a concise description of the anomaly localization process conducted by the anomaly handling module. First, a node structure is defined, where each node contains identity information and its corresponding trust score. *nodes* is composed of multiple *node*. The updateScores() function dynamically updates the trust scores of nodes. Thresholds threshold_d and threshold_p are established. If a node's trust score falls below threshold_p, it is considered an anomaly node with disruptive behavior to the data forwarding path. If the trust score is below threshold_d and the node is on a malicious link, it is identified as an anomaly node engaging in data tampering. The function outputs the anomaly nodes and utilizes remove() to eliminate them from the network topology graph G, ensuring that the secure multi-path routing module generates paths that exclude these anomaly nodes. In Section 3.3, malicious nodes tampering with data will cause a decrease in trust scores for other nodes in the same path. Therefore, after locating the malicious node, renew() is used to update the trust scores of nodes affected by the malicious node, restoring their initial trust scores. This ensures the accuracy and real-time nature of the secure multi-path generation process (Algorithm 2, lines 2–11).

When the anomaly handling module detects an abnormal link (S_{t-1},S_t), it first registers S_{t-1} and S_t, and then uses renew() to restore the initial score of score_d of other nodes in the forwarding path where the abnormal link is located. In order to effectively identify malicious nodes from the abnormal link, revice() is used to set the Q value of the action from state S_{t-1} to state S_t in the Q table in Section 3.4 to negative infinity, and the updated Q table is sent to the multipath routing module, so that S_{t-1} and S_t are distributed in different paths, thereby distinguishing the trust scores of the two nodes to effectively locate the malicious nodes. (Lines 12–15 in Algorithm 2).

Algorithm 2. Locating abnormal nodes.

Input: Abnormal link (S_{t-1},S_t), $score\_d$ and $score\_p$ of S_t, Q table
Output : Malicious node
1.Begin
2. *nodes*=[]
3. While true :
4. *nodes* = updateScores()
5. for each *node* in *nodes*:
6. If *node.score_p* < threshold_p :
7. output(*node*)
8. remove(*node*,G)
9. If *node.score_d* < threshold_d && *node* is in an abnormal link :
10. output(*node*)
11. remove(*node*,G), renew(*nodes*,*node*)
12. If abnormal link (S_{t-1},S_t) exists:
13. renew(*nodes*,(S_{t-1},S_t))
14. Q' = revise(Q,S_{t-1},S_t)
15. send Q' to Multipath routing module
16.End

4. Simulation and analysis

In this section, we establish a simulated network environment to simulate the proposed scheme. We discuss the necessity of SZSM and conduct an analysis of its effectiveness and performance.

4.1. Simulation environment

We selected a host with an Intel i7-11370H processor running at 4.266GHz and 32GB of RAM for the simulation process. The operating system used is Ubuntu 14.06. The simulation environment was constructed using Mininet, and the BMv2 (behavioral-model version 2) switch model was employed as the network switch for data transmission. The switches were programmed using the P4 language, and JSON-formatted description files were generated to define the programmable behavior of the data plane [34]. On this foundation, we chose the Ryu controller and implemented the functionalities of various modules in TESM using the Python language. The generation of session keys was achieved through the ECPy (Elliptic Curve Python) library, and HMAC (hash-based message authentication code) was utilized for the generation of message authentication codes [35].

For simulating the IoT environment, we opted for the fat-tree topology as depicted in Figure 6. This topology includes 2 pod units, 10 switches (S_1 to S_{10}), and 8 IoT nodes (N_1 to N_8). Each switch has 4 ports, and multiple transmission paths exist between IoT nodes.

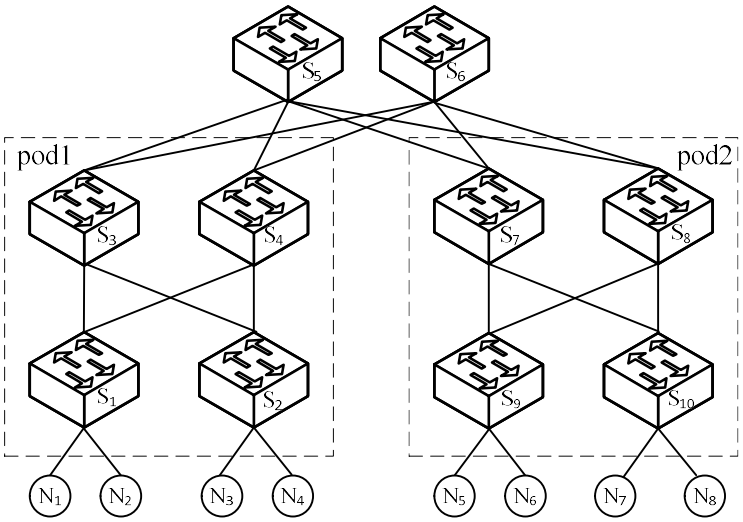


Figure 6. Topology structure.

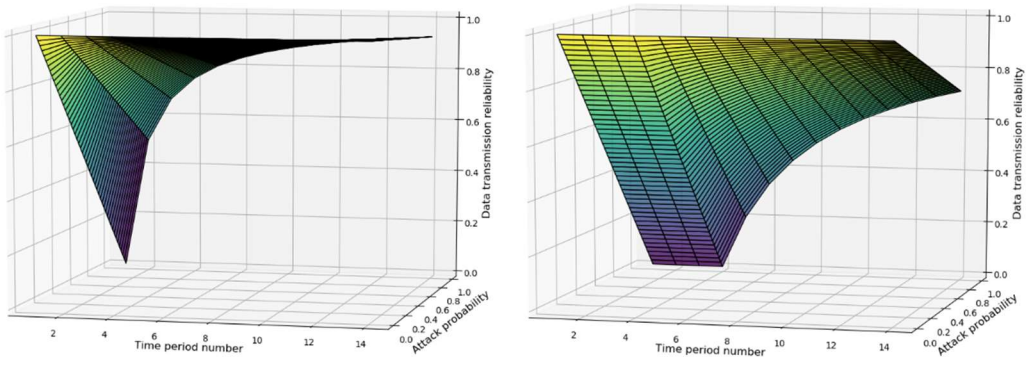
4.2. Necessity analysis

In the process of data transmission, the emergence of malicious switch nodes can lead to the failure of transmission paths, and anomalous paths can result in a decrease in data integrity and transmission reliability. Timely replacement of secure paths for data transmission can minimize the risks posed by malicious nodes. However, if the replacement path still includes malicious nodes, the system will continue to face security threats. Taking the topology structure illustrated in Figure 6 as an example, we conducted a simulation analysis using the NetworkX library in Python to assess the impact of anomalous nodes on the reliability of data transmission. It is stipulated that data is transmitted from N_1 to N_6, with the default forwarding path in the network being S_1-S_3-S_5-S_7-S_9. Assuming S_5 is a malicious node, conducting malicious attacks such as tampering or discarding data with an attack probability P_A, the system periodically checks the forwarding path and replaces the anomalous forwarding path in the next time cycle. We describe the system's recovery capability after being affected by anomalous paths using the dynamic variation in data transmission reliability, as shown in Eq (11), where *Data_correct* and *Data_total* represent the quantities of correctly transmitted data and total data, respectively. The system randomly selects a shortest path as an alternative path, and the resulting variation in data transmission reliability is depicted in Figure 7.

$$\text{Data transmission reliability} = \frac{Data\_correct}{Data\_total} \tag{11}$$

There are 8 different shortest paths between N_1 and N_6, with 4 of them containing the malicious node S_5. In Figure 7(a), after discovering the anomalous path, the system switches to a secure

forwarding path for data transmission in the next time period. The reliability of data transmission is at its lowest when the anomalous path appears and steadily increases after the path replacement. In Figure 7(b), in the first 3 time periods, all the replacement paths still include the malicious node, and it is only in the fourth time period that a secure forwarding path without the malicious node is selected. When the attack probability of the malicious node is 0.5, the data transmission reliabilities for the two path replacement schemes (a) and (b) in the 5th and 10th time periods are 0.9, 0.95 and 0.6, 0.8, respectively. Choosing an unreasonable path replacement scheme can significantly impact the system's reliability. When the attack probability of the malicious node is 1, the data transmission reliabilities for scheme (b) in the 5th and 10th time periods are 0.2 and 0.6, respectively. It is evident that as the attack probability of the malicious node increases, the impact on the secure data transmission becomes more substantial.



(a) Optimal path replacement

(b) Worst path replacement

Figure 7. Changes in data transmission reliability of different path replacement schemes.

This section only considers the selection of the shortest path for replacing anomalous paths. When there are no restrictions on the replacement paths, it may lead to the selection of more anomalous paths during the replacement process, thereby significantly reducing the reliability of data transmission and the system's recovery capability. Therefore, when anomalous paths are detected, the key to improving system stability and risk resistance lies in how to quickly select a secure alternative path.

4.3. Effectiveness analysis

To verify the effectiveness of TESM, we conduct an effectiveness analysis of TESM from several aspects including node trust scoring, multipath scoring, and abnormal node positioning.

Experiment 1: In the topology structure illustrated in Figure 6, N_1 continuously sends data to N6 along the path S_1-S_3-S_5-S_7-S_9. Assuming S_5 is a malicious node with a 20% probability of malicious tampering of data, Figure 8 presents the statistical data on trust scores for various nodes in TESM.

The content depicted in Figure 8 is detailed below. At the initial stage, the initial trust score for each node is set to 1. Data packets that have been tampered with cannot pass through the packet security verification process of TESM. In the second time period, based on Eq (6), the trust scores of nodes in the path are updated with $w_d = 0.2$, causing the trust scores of S_1, S_3, S_5, S_7, and S_9 to decrease

to 0.84. In the third time period, TESM utilizes the alternative path S_1-S_4-S_6-S_8-S_9 to transmit data, resulting in an increase in the trust scores for S_1 and S_9 as the alternative path is secure. In the fourth time period, TESM detects the anomalous link as S_5-S_7 and restores the trust scores of the remaining nodes in the path to their initial values. TESM, through feedback from the anomaly handling module, ensures that S_5 and S_7 do not appear in the same path. In the fifth and sixth time periods, N_2 sends data to N_8 along the path S_2-S_4-S_6-S_7-S_{10}, causing the trust score of S_7 to rise to 0.99. In the seventh time period, N_2 sends data to N_5 along the path S_2-S_4-S_6-S_7-S_{10} and TESM, upon detecting data tampering, updates the trust scores of nodes in the path. At this point, the trust score of S_5 drops to 0.8 and the anomaly handling module identifies S_5 as a malicious node, subsequently restoring the trust scores of nodes affected by S_5 in the next time period.

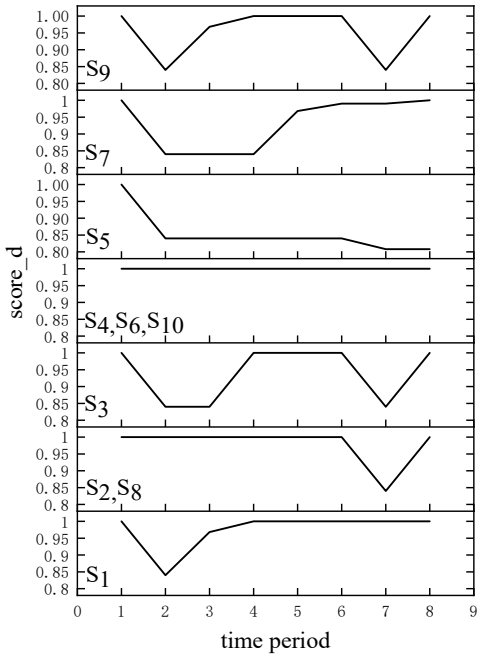


Figure 8. Node trust scoring based on data.

Experiment 2: In the topology structure illustrated in Figure 6, N_1 sends data to N_6 along the path S_1-S_3-S_5-S_7-S_9. Suppose S_3 and S_6 are malicious nodes, where S_3 violates the original forwarding path to transmit data, and S_3 maliciously discards data. Let there be a 2% natural packet loss probability on the forwarding links. Figure 9 presents the statistical data on trust scores for various nodes in TESM.

At the initial stage, the initial trust score for each node is set to 1. In the second time period, the trust scores of nodes S_1, S_5, S_7, and S_9 are affected by the natural packet loss rate, resulting in a decrease to 0.98. Meanwhile, the trust score of S_3 drops to 0.78 because it forwards 20% of the data through the path S_3-S_6-S_7-S_9, revealing S_3 as a malicious node. In the third time period, TESM uses the alternative path S_1-S_4-S_6-S_8-S_9 to transmit data, causing the trust score of S_6 to decrease to 0.78 as it maliciously discards 20% of the data, identifying S_6 as a malicious node. Finally, TESM updates the forwarding path to S_1-S_4-S_5-S_7-S_9, and the trust scores for all nodes become 0.98, indicating that the switch nodes are in a secure forwarding state.

From Experiments 1 and 2, it can be concluded that TESM achieves real-time statistics of trust scores for switch nodes through secure data verification, secure detection of forwarding paths, and

monitoring and statistics of data packet forwarding quantities. Additionally, TESM enables secure path selection and effective localization of malicious nodes based on node trust scores.

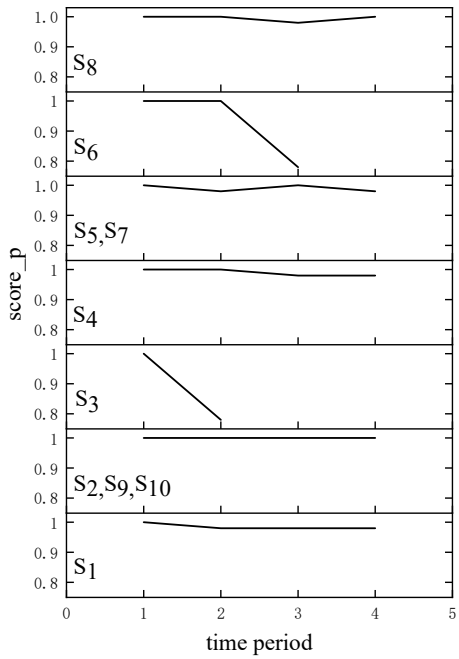


Figure 9. Node trust scoring based on path.

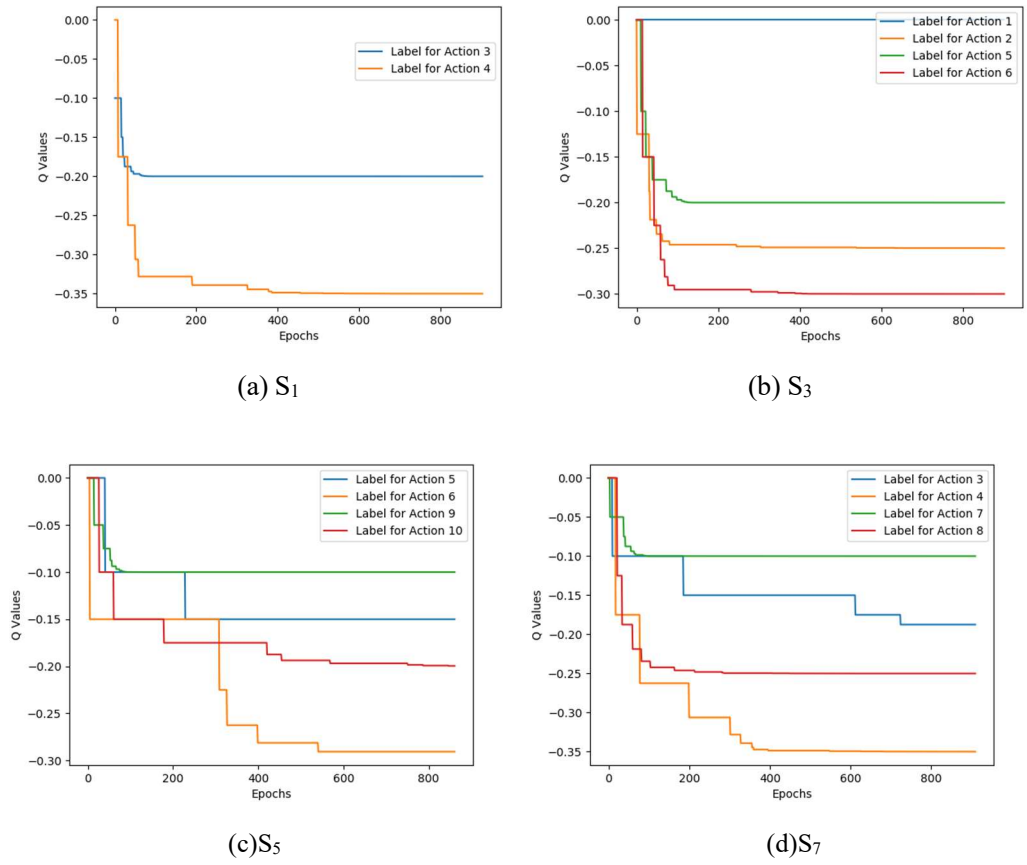
Experiment 3: Table 3 sets trust scores *score_d* and *score_p* for the switch nodes in Figure 6. Without considering malicious nodes, the process of TESM generating multiple secure paths between N_1 and N_6 is analyzed based on the parameters shown in Table 3. TESM sets w_1~ w_3 in Eq (7) to 0.5, 0.5, and 1, respectively. Both α and γ in Eq (8) are set to 0.5, and the number of iterations for reinforcement learning is set to 500.

Table 3. Node parameter settings.

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
score_d	0.9	0.8	0.9	0.7	0.8	0.7	0.9	0.8	0.9	0.8
score_p	0.8	0.7	0.7	0.6	0.8	0.7	0.9	0.7	0.8	0.7

The size of the Q-value reflects the gain of selecting the corresponding action. During each Q-table update, the changes in Q-values for exchange nodes S_1, S_3, S_5, and S_7 are recorded. The statistical results are shown in Figure 10. Each subplot displays different-colored lines representing distinct actions, where the number of actions corresponds to the number of nodes adjacent to the current node. For instance, concerning S_1, which is adjacent to S_3 and S_4, Actions 3 and 4 in Figure 10(a) respectively denote selecting S_3 and S4 as the next-hop node for S1. From the graph, it can be observed that with the operation of the TESM multi-path routing module, the Q-values for different actions continuously decrease. This is because the negative reward based on path length in Eq (7) is higher than the positive incentive based on trust scores. As the number of iterations increases, the Q-value curves for each action ultimately reach a stable state, indicating that the Q-table has been trained. For the same state,

when an action has a higher Q-value, it suggests that selecting the current action will result in a path with higher gains. Therefore, based on Figure 10(a),(c),(d), it can be concluded that the optimal next-hop nodes for S_1, S_5, and S_7 are S_3, S_7, and S_9, respectively. Since the multi-path routing module prohibits creating loops in the path, the action from S_3 to S_1 is not executed during the loop, resulting in the Q-value for Action 1 in Figure 10(b) remaining unchanged at 0. Therefore, for S_3, the optimal next-hop node is selected from Figure 10(b), with S5 having the highest Q-value, excluding S_1.



(a) S_1 (b) S_3

(c)S_5 (d)S_7

Figure 10. Changes in Q-values of nodes S_1, S_3, S_5, S_7.

According to Eq (9), the calculated path scores for some paths are presented in Table 4. From the table, it can be observed that the path S_1-S_3-S_5-S_7-S_9 has the highest score. Under the assumption of the same network state for the switch nodes, this path balances both security and efficiency in data transmission, making it a preferred forwarding path between N_1 and N_6. TESM combines multiple path scores with specific requirements to provide a multi-path routing strategy for the system. For instance, in the case of path failure, although the path S_1-S_3-S_6-S_7-S_9 has a higher path score, it has more overlapping nodes with the original path, potentially including malicious nodes from the original path. Therefore, paths with fewer overlapping nodes, such as S_1-S_3-S_6-S_8-S_9 and S_1-S_4-S_6-S_7-S_9, are more suitable as replacement paths, enhancing the system's resilience. When real-time data forwarding with high latency requirements is needed using multi-path routing, TESM must consider both the security scores and the diversity between paths. This ensures the secure and rapid transmission of data through multiple paths. Furthermore, the process of TESM's multi-path scoring is extensible. The assumption made in the paper is that all switch nodes have the same status in the network. However,

by incorporating parameters such as bandwidth and load as scoring criteria in Eq (7), the multi-path routing can be more finely divided based on the network conditions of individual switch nodes.

Table 4. Multipath scoring.

	path	scores
1	S_1-S_3-S_5-S_7-S_9	−0.749
2	S_1-S_3-S_6-S_7-S_9	−0.849
3	S_1-S_3-S_5-S_8-S_9	−0.899
4	S_1-S_4-S_5-S_7-S_9	−0.899
5	S_1-S_3-S_6-S_8-S_9	−0.999
6	S_1-S_4-S_6-S_7-S_9	−1
7	S_1-S_4-S_5-S_8-S_9	−1.049
8	S_1-S_3-S_5-S_7-S_{10}-S_8-S_9	−1.199
9	S_1-S_3-S_5-S_7-S_{10}-S_7-S_9	−1.199

Experiment 4: To validate TESM's ability to detect malicious data and locate anomalous nodes, we designed Experiment 4. In the network topology shown in Figure 6, assume that S_5 and S_6 are malicious nodes. They will continuously launch content tampering attacks and abnormal forwarding path attacks with probabilities of 0.5, 0.75, 1, 2, and 4%. A malicious activity by a malicious switch node within a random time interval $T_i(100\text{ms} \le T_i \le 200\text{ms})$ is considered one attack event. If TESM can detect abnormal behavior within the detection period T and successfully locate the malicious node, we consider it a successful defense. Define the number of attack events and non-attack events executed by malicious switch nodes within the specified time as N and M, respectively. The number of times TESM did not detect attacks is denoted as FN_1, the number of times TESM detected attacks but failed to accurately locate the malicious node is denoted as FN_2, the number of false positives where TESM detected attacks but misclassified normal nodes as malicious nodes is denoted as FP_1, and the number of false positives where TESM detected and provided anomalous localization when malicious nodes were not attacking is denoted as FP_2. The false negative rate (FNR) is calculated as $FNR = (FN_1 + FN_2) / N$, and the false positive rate (FPR) is calculated as $FPR = (FP_1 + FP_2) / M$. Assuming a 0.5% natural packet loss rate in the links of Figure 6, TESM sets trust thresholds threshold_d and threshold_p to 0.9 and 0.95, respectively. Figures 11 and 12 depict the performance of TESM in terms of false negative rate and false positive rate for detecting and locating S_5 and S_6 in Experiment 4.

From Figure 11, it can be observed that the FPR of TESM for tampered data remains constant with varying attack probabilities. This is because TESM, in conjunction with the location of anomalous links and threshold_d, effectively locates malicious nodes, preventing safe nodes from being misclassified as malicious. The FNR decreases as the attack probability increases, reaching 0 when the attack probability exceeds 2%. When S_5 tampers with data at relatively low attack probabilities, such as 0.5 and 0.75%, the obtained FNR values are 6.6 and 2.5%, respectively. Lower attack probabilities pose challenges for TESM's anomaly handling module to effectively locate malicious nodes within T, resulting in some FNR. This issue can be mitigated by increasing the detection period T. When T is set to 200 ms, the FNR decreases to 4.2 and 0.8% for attack probabilities of 0.5 and 0.75%, respectively. Compared to T = 100 ms, this represents a reduction of 36 and 68%, demonstrating that TESM can

achieve improved anomaly detection and localization by dynamically adjusting T.

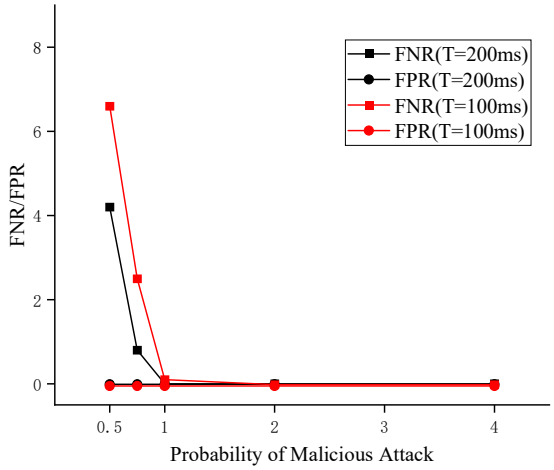


Figure 11. Detection and location effect of tampered data.

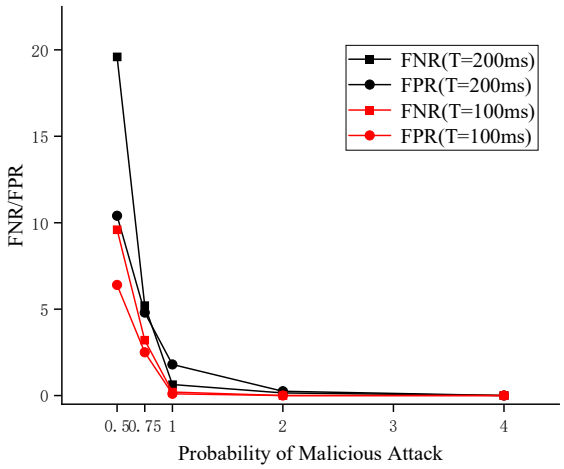


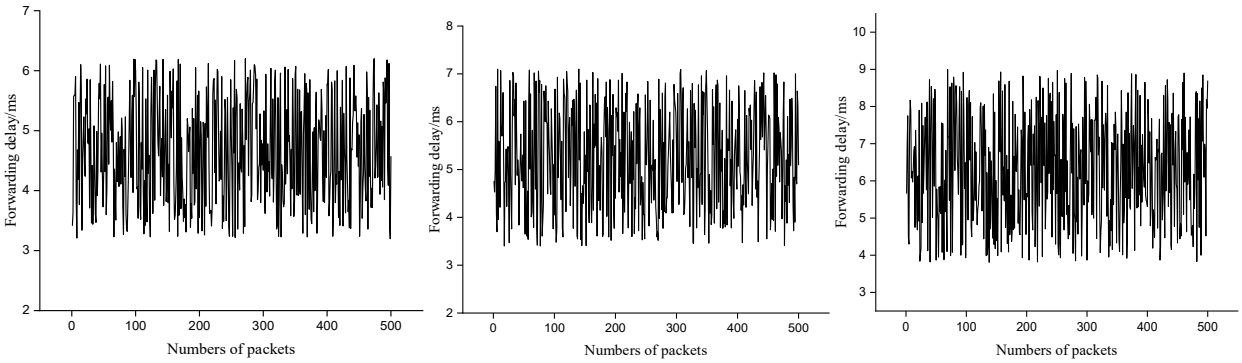
Figure 12. Detection and location effect of path anomalies.

Figure 12 reveals that, with a fixed T, both the FNR and FPR of TESM for path anomaly attacks decrease as the attack probability increases, reaching 0 when the attack probability exceeds 2%. When S6 tampers with data at lower attack probabilities, such as 0.5 and 0.75%, the obtained FNR values are 19.6 and 5.2%, respectively. This is because, in this scenario, TESM's threshold_p is equivalent to the amount of data transmitted over the link, excluding natural packet loss. Therefore, TESM struggles to effectively differentiate between S_6 and cases of natural packet loss. Similarly, the obtained FPR values are 10.4 and 4.8%. Packets subjected to rare path anomaly attacks, such as malicious drops or route deviation, do not pose a threat to the destination host. Therefore, when the attack probability is low, the harm caused by malicious switch nodes before being effectively located is relatively small. Consistent with the results of Figure 11, when T is set to 200 ms, the obtained FNR values are 9.6 and 3.2%, while the FPR values are 6.4% and 2.5% for attack probabilities of 0.5 and 0.75%, respectively. Thus, by increasing T, detection and localization effectiveness for path

anomaly attacks can also be enhanced.

4.4. Performance evaluation

Experiment 5: To test the forwarding latency of TESM at different stages and compare it with the forwarding latency in traditional networks, we conducted Experiment 5. In the network topology shown in Figure 6, N_1 continuously sent 500 data packets to N_6 along the path S_1-S_3-S_5-S_7-S_9. The forwarding latency for three scenarios was then statistically analyzed. In Figure 13 represents the forwarding latency in traditional networks, TESM during the data packet security verification stage, and TESM during the anomaly link localization stage.



(a) Traditional network (b) Packet security verification stage (c) Abnormal link positioning stage

Figure 13. Data forwarding delay.

From Figure 13, it can be observed that the average forwarding latency for data packets in a traditional network is 4.67 ms. In the TESM packet security verification stage, the average forwarding latency for data packets is 5.25 ms, introducing an additional 12.4% latency compared to traditional networks for the calculation and verification of data packet hash values. In the TESM anomaly link localization stage, the average forwarding latency for data packets is 6.29 ms, introducing additional latency for the calculation of data packet message authentication codes at each switch node in the path. Although the anomaly link localization stage introduces higher forwarding latency, TESM uses replicated data for this process, and the destination switch node does not forward the data to the destination node. Therefore, the anomaly link localization process does not impact the forwarding latency of data in the network. Figure 14 presents the content of Figure 13 as a cumulative distribution function (CDF) curve. From the graph, it can be inferred that in a traditional network, over 95% of data packets have a forwarding latency of 5.8 ms or less. In the TESM packet security verification stage, over 95% of data packets have a forwarding latency of 6.55 ms or less, indicating that the introduced additional latency is acceptable.

Table 5 compares the forwarding latency of various schemes with TESM. From the table, it can be observed that LPV [18] incurs higher forwarding latency because it requires sampling data packets at both the ingress and egress switches. P4Label [21] has a base forwarding latency of 3.2 ms, but it requires verification of identity-based digital signatures, a process that takes 16.2 ms. Until the signature verification is complete, the security of the data cannot be verified. In SDNsec [22], each switch needs to perform at least 2 message authentication code calculations for the verification of the forwarding path. This computational load far exceeds the corresponding stages in TESM. SDNsec uses

the data plane development kit (DPDK) for simulation experiments, which may contribute to its lower forwarding latency. MRR [31] shows forwarding latency proportional to the number of paths, requiring source and destination switches to perform multipath computation, forwarding, and multipath comparison. As the number of paths increases, this process introduces higher forwarding latency. In contrast, TESM only needs to perform a lightweight, low-latency hashing-based verification process at the egress switch, making it an efficient and low-latency solution.

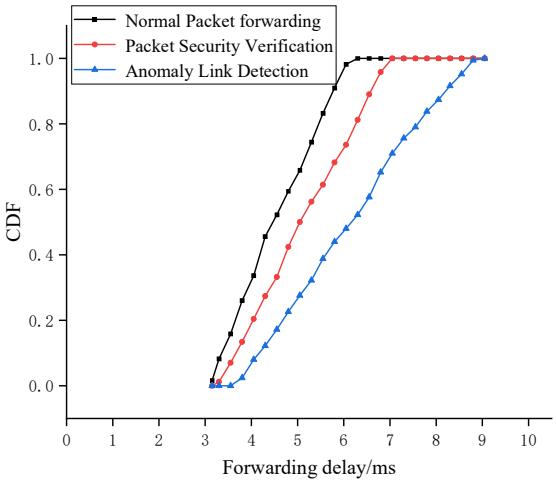


Figure 14. CDF curve.

Table 5. Comparison of various schemes.

Scheme	Main technology	Forwarding delay	Functions
LPV [18]	Sampling verification	33.17 ms (3 to 5 switches)	Detecting and locating forged and tampered packets
P4Label [21]	Identity-based signature	3.2 ms (3 switches)	Detecting forged and tampered packets
SDNsec [22]	Message authentication code verification	0.8 ms/switch	Forwarding path consistency verification
MRR [31]	Multipath comparison verification	80.4 ms (6 switches)	Ensuring correctness of flow rules and data, achieving load balancing
TESM	Hash verification	5.25 ms (5 switches)	Ensuring security of data, paths, and nodes, providing multiple secure paths

Experiment 6: In the network topology illustrated in Figure 6, sender N_1 transmits data to receiver N_6 with payload sizes of 300, 600, 900, and 1200B. Throughput tests were conducted for both the traditional network and TESM. In TESM, evaluations were conducted for both single-path and multi-path transmissions. The forwarding paths for the traditional network and single-path transmission were set as S_1-S_3-S_5-S_7-S_9. In the multi-path transmission mode, the system utilized two paths concurrently: S_1-S_3-S_5-S_7-S_9 and S_1-S_4-S_6-S_8-S_9. The results are depicted in Figure 15.

From Figure 15, it is evident that network throughput increases with the growth of payload size in

all three scenarios. When TESM employs single-path data transmission, the network throughput is slightly reduced by an average of 5.46% compared to the traditional network. In the case of TESM utilizing two paths for data transmission, the throughput is approximately twice that of the traditional network and single-path transmission. Therefore, it can be concluded that TESM incurs relatively modest network throughput losses compared to the traditional network. Additionally, when TESM adopts multi-path data transmission, there is an improvement in throughput, which is advantageous for enhancing network transmission efficiency and reliability.

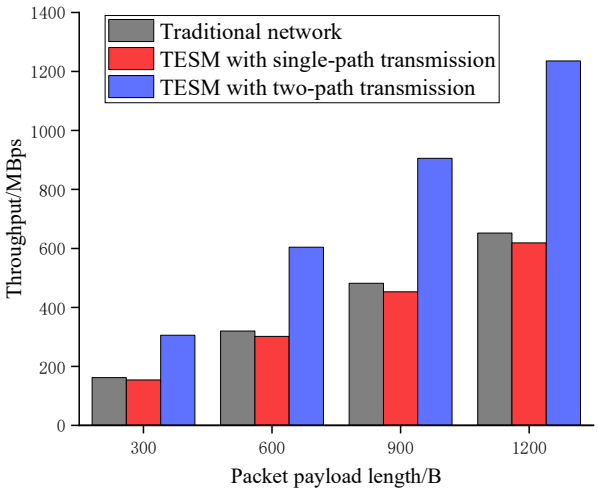


Figure 15. Comparison of throughput.

5. Conclusions

To address security threats in data transmission processes between edge nodes and between edge nodes and cloud data centers, this paper proposes a trust evaluation based IoT secure multi-path routing solution, TESM. TESM incorporates the SDN architecture into the data transmission between edge nodes, achieving dynamic selection of secure transmission paths and effective localization of malicious nodes through continuous security validation of data flows and real-time trust scoring of switch nodes. Implemented with Ryu controller and P4 switches, TESM is tested in a simulated environment using Mininet, demonstrating its ability to enhance the security and stability of data transmission. Compared to related approaches, TESM incurs lower time overhead. In contrast to traditional networks, TESM introduces a 12.46% forwarding delay and a 5.46% throughput overhead. Thus, TESM emerges as a lightweight and effective security defense solution for the IoT.

The next research focus is on addressing the dynamic allocation of secure multi-path routing in the IoT. This involves implementing fine-grained control over multiple paths based on factors such as network status, user requirements, and business types. The goal is to further enhance network reliability and service quality.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. A. A. Laghari, K. Wu, R. A. Laghari, M. Ali, A. A. Khan, A review and state of art of Internet of Things (IoT), *Arch. Comput. Methods Eng.*, **29** (2022), 1395–1413. https://doi.org/10.1007/s11831-021-09622-6
2. F. Guo, F. R. Yu, H. Zhang, X. Li, H. Ji, V. C. Leung, Enabling massive IoT toward 6G: A comprehensive survey, *IEEE Int. Things J.*, **8** (2021), 11891–11915. https://doi.org/10.1109/JIOT.2021.3063686
3. G.Abbas, A. Mehmood, M. Carsten, G. Epiphaniou, J. Lloret, Safety, Security and Privacy in Machine Learning Based Internet of Things, *J. Sensor Actuator Networks*, **11** (2022), 38. https://doi.org/10.3390/jsan11030038
4. B. Costa, J. Bachiega, L. R. de Carvalho, A. P. Araujo, Orchestration in fog computing: A comprehensive survey, *ACM Comput. Surv.*, **55** (2022), 1–34. https://doi.org/10.1145/3486221
5. Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, W. Lv, Edge computing security: State of the art and challenges, *Proc. IEEE*, **107** (2019), 1608–1631. https://doi.org/10.1109/JPROC.2019.2918437
6. A. M. Alwakeel, An overview of fog computing and edge computing security and privacy issues, *Sensors*, **21** (2021), 8226. https://doi.org/10.3390/s21248226
7. P. Ranaweera, A. D. Jurcut, M. Liyanage, Survey on multi-access edge computing security and privacy, *IEEE Commun. Surv. Tutorials*, **23** (2021), 1078–1124. https://doi.org/10.1109/COMST.2021.3062546
8. J. J. Kang, K. Fahd, S. Venkatraman, R. Trujillo-Rasua, P. Haskell-Dowland, Hybrid routing for Man-in-the-Middle (MITM) attack detection in IoT networks, in *2019 29th International Telecommunication Networks and Applications Conference (ITNAC)*, (2019), 1–6. https://doi.org/10.1109/ITNAC46935.2019.9077977
9. N. Mckeown, Software-defined networking, in *IEEE International Conference on Computer Communications*, (2009), 30–32. https://doi.org/10.1145/1530748.1530749
10. S. Javanmardi, M. Shojafar, R. Mohammadi, M. Alazab, A. M. Caruso, An SDN perspective IoT-Fog security: A survey, *Comput. Networks*, **229** (2023), 109732. https://doi.org/10.1016/j.comnet.2023.109732
11. M. Z. Hussain, Z. M. Hanapi, Efficient secure routing mechanisms for the low-powered IoT network: A literature review, *Electronics*, **12** (2023), 482. https://doi.org/10.3390/electronics12030482
12. K. Ramezanpour, J. Jagannath, Intelligent zero trust architecture for 5G/6G networks: Principles, challenges, and the role of machine learning in the context of O-RAN, *Comput. Networks*, **217** (2022), 109358. https://doi.org/10.1016/j.comnet.2022.109358

13. F. Kamoun-Abid, A. Meddeb-Makhlour, F. Zarai, M. Guizani, DVF-fog: distributed virtual firewall in fog computing based on risk analysis, *Int. J. Sensor Networks*, **4** (2019), 30. https://doi.org/10.1504/IJSNET.2019.101242
14. K. A. Sadiq, A. F. Thompson, O. A. Ayeni, Mitigating DDoS attacks in cloud network using fog and SDN: A conceptual security framework, *Int. J. Appl. Inf. Syst.*, **32** (2020), 11–16. https://doi.org/10.5120/ijais2020451877
15. M. Dhawan, R. Poddar, K. Mahajan, V. Mann, Sphinx: detecting security attacks in software-defined networks, in *Ndss*, (2015), 8–11. https://doi.org/10.14722/ndss.2015.23064
16. T. G. Nguyen, T. V. Phan, B. T. Nguyen, C. So-In, Z. A. Baig, S. Sanguanpong, Search: A collaborative and intelligent nids architecture for sdn-based cloud iot networks, *IEEE Access*, **7** (2019), 107678–107694. https://doi.org/10.1109/ACCESS.2019.2932438
17. M. Pourvahab, G. Ekbatanifard, An efficient forensics architecture in Software-Defined Networking-IoT using blockchain technology, *IEEE Access*, **7** (2019), 99573–99588. https://doi.org/10.1109/ACCESS.2019.2930345
18. S. Wang, Q. Li, Y. Zhang, LPV: Lightweight packet forwarding verification in SDN, *J. Comput.*, **42** (2019), 176–189.
19. L. Xie, Y. Ding, H. Yang, X. Wang, Blockchain-based secure and trustwor-thy internet of things in SDN-enabled 5G-VANETs, *IEEE Access*, **7** (2019), 56656–56666. https://doi.org/10.1109/ACCESS.2019.2913682
20. D. Li, E. Zhang, M. Lei, C. Song, Zero trust in edge computing environment: a blockchain based practical scheme, *Math. Biosci. Eng.*, **19** (2022), 4196–4216. https://doi.org/10.3934/mbe.2022194
21. Z. Zuo, C. Chang, Y. Zhang, R. He, X. Qin, K. L. Yung, P4Label: packet forwarding control mechanism based on P4 for software-defined networking, *J. Ambient Intell. Human. Comput.*, **2020** (2020), 1–14. https://doi.org/10.1007/s12652-020-01719-3
22. T. Sasaki, C. Pappas, T. Lee, T. Hoefler, A. Perrig, SDNsec: Forwarding accountability for the SDN data plane, in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, 2016. https://doi.org/10.1109/ICCCN.2016.7568569
23. S. A. Latif, F. B. X. Wen, C. Iwendi, F. W. Li, S. M. Mohsin, Z. Han, et al., AI-empowered, blockchain and SDN integrated security architecture for IoT network of cyber physical systems, *Comput. Commun.*, **181** (2022), 274–283. https://doi.org/10.1016/j.comcom.2021.09.029
24. Z. Zeng, X. Zhang, Z. Xia, Intelligent blockchain-based secure routing for multidomain SDN-enabled IoT networks, *Wireless Commun. Mob. Comput.*, **2022** (2022), 1–10. https://doi.org/10.1155/2022/5693962
25. J. Yan, H. Zhang, Q. Shuai, B. Liu, X. Guo, HiQoS: An SDN-based multipath QoS solution, *China Commun.*, **12** (2015), 123–133. https://doi.org/10.1109/CC.2015.7112035
26. S. Alqahtani, A. Alotaibi, A route stability-based multipath QoS routing protocol in cognitive radio ad hoc networks, *Wireless Networks*, **25** (2019). https://doi.org/10.1007/s11276-019-02014-6
27. Q. De Coninck, O. Bonaventure, C. Multipathtester, Comparing mptcp and mpquic in mobile environments, in *2019 Network Traffic Measurement and Analysis Conference (TMA), IEEE*, (2019), 221–226. https://doi.org/10.23919/TMA.2019.8784653
28. C. Pu, Jamming-resilient multipath routing protocol for flying ad hoc networks, *IEEE Access*, **6** (2018), 68472–68486. https://doi.org/10.1109/ACCESS.2018.2879758

29. D. Jin, Z. Li, C. Hannon, C. Chen, J. Wang, M. Shahidehpour, C. W. Lee, Toward a cyber resilient and secure microgrid using software-defined networking, *IEEE Trans. Smart Grid*, **8** (2017), 2494–2504. https://doi.org/10.1109/TSG.2017.2703911
30. T. Li, C. Hofmann, E. Franz, Secure and reliable data transmission in SDN-based backend networks of industrial IoT, in *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, 2020. https://doi.org/10.1109/LCN48667.2020.9314854
31. Q. Ren, T. Hu, J. Wu, Y. Hu, L. He, J. Lan, Multipath resilient routing for endogenous secure software defined networks, *Comput. Networks*, **194** (2021), 108134. https://doi.org/10.1016/j.comnet.2021.108134
32. X. Guo, H. Lin, Z. Li, M. Peng, Deep-reinforcement-learning-based QoS-aware secure routing for SDN-IoT, *IEEE Int. Things J.*, **7** (2019), 6242–6251. https://doi.org/10.1109/JIOT.2019.2960033
33. J. Clifton, E. Laber, Q-learning: Theory and applications, *Ann. Rev. Stat. Appl.*, **7** (2020), 279–301. https://doi.org/10.1146/annurev-statistics-031219-041220
34. P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, et al., P4: Programming protocol-independent packet processors, *ACM SIGCOMM Comput. Commun. Rev.*, **44** (2014), 87–95. https://doi.org/10.1145/2656877.2656890
35. H. Krawczyk, M. Bellare, R. Canetti, HMAC: Keyed-hashing for message authentication, 1997. https://doi.org/10.17487/rfc2104

AIMS Press

©2024 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0)