*applied sciences*

MDPI

*Article*

# An Adaptive Security Framework for Internet of Things Networks Leveraging SDN and Machine Learning

**Ala Hamarsheh**

Department of Computer Science, Faculty of Information Technology, Arab American University, Jenin P.O. Box 240, Palestine; ala.hamarsheh@aaup.edu

**Abstract:** The Internet of Things (IoT) is expanding rapidly with billions of connected devices worldwide, necessitating robust security solutions to protect these systems. This paper proposes a comprehensive and adaptive security framework called Enhanced Secure Channel Authentication using random forests and software-defined networking (SCAFFOLD), tailored for IoT environments. The framework establishes secure communication channels between IoT nodes using software-defined networking (SDN) and machine learning techniques. The key components include encrypted channels using session keys, continuous traffic monitoring by the SDN controller, ensemble machine-learning for attack detection, precision mitigation via SDN reconfiguration, and periodic reauthentication for freshness. A mathematical model formally defines the protocol. Performance evaluations via extensive simulations demonstrate Enhanced SCAFFOLD's ability to reliably detect and rapidly mitigate various attacks with minimal latency and energy consumption overheads across diverse IoT network scenarios and traffic patterns. The multidimensional approach combining encryption, intelligent threat detection, surgical response, and incremental hardening provides defense-in-depth to safeguard availability, integrity, and privacy within modern IoT systems while preserving quality of service.

**Keywords:** Internet of Things; IoT security; software-defined networking; machine learning; attack detection

**Citation:** Hamarsheh, A. An Adaptive Security Framework for Internet of Things Networks Leveraging SDN and Machine Learning. *Appl. Sci.* **2024**, *14*, 4530. https://doi.org/10.3390/app14114530

Academic Editors: Guangquan Xu and Qizhi Xu

Received: 5 March 2024
Revised: 19 May 2024
Accepted: 20 May 2024
Published: 25 May 2024

## 1. Introduction

IoT refers to the billions of physical devices around the world that are now connected to the internet, collecting and sharing data [1–3]. IoT devices such as smart home appliances, wearables, vehicles, and industrial equipment contain embedded sensors, software, and connectivity that enable them to connect, exchange data, and be remotely monitored and controlled.

A critical foundation of the IoT ecosystem is the networking infrastructure, enabling communication between devices and applications. The sheer scale of IoT, with projections of billions of connected devices globally, necessitates efficient and scalable protocols for identification, addressing, routing, and packet handling [4]. IPv6 has emerged as the foremost networking protocol, underpinning modern IoT deployments due to its expansive address space capable of uniquely identifying every conceivable device [4]. The 128-bit addresses of IPv6 allow for approximately 340 undecillion address combinations, drastically exceeding the 4.3 billion limit of the previous 32-bit IPv4 standard. This massive address pool can readily accommodate current IoT growth and future expansion to trillion-scale device networks. In addition to alleviating address exhaustion, IPv6 also streamlines packet processing through fixed-length headers and a simplified structure compared to IPv4 with its optional header fields [4]. This benefits lightweight IoT endpoints with limited processing resources. IPv6 also mandates IP security (IPsec) implementation, providing native authentication, integrity, and encryption for communications

[5]. And enhancements, such as expanded mobility options, better facilitate devices changing networks while maintaining connections, which is essential for mobile IoT use cases.

However, bare IPv6 is not sufficient for all IoT scenarios, especially involving the low-power wireless networks commonly utilized for IoT sensors and home automation. Protocols like 6LoWPAN adapt IPv6 packets for efficient transmission over IEEE 802.15.4, Bluetooth, and Zigbee by compressing headers and fragmenting large packets [6]. This allows devices like wireless sensors to take advantage of IPv6's benefits while optimizing for low-bandwidth lossy networks. Meanwhile, routing and mesh protocols like Routing Protocol for Low-Power and Lossy Networks (RPL), designed for constrained environments, enable multi-hop IoT topologies that are not well served by IP alone [7]. RPL provides optimization, such as efficient neighbor discovery, distributed path selection, and autonomous repair, which helps overcome the unpredictability of low-power wireless IoT networks. Therefore, for holistic IoT communication, the interplay of foundational IPv6 connectivity, specialized adaptations like 6LoWPAN for wireless links, and tailored routing protocols like RPL for mesh architectures establish a robust framework. Of course, security is also paramount, requiring standards like IPSec, DTLS, and TLS to protect end-to-end communications with encryption and authentication at scale [5,8].

While IoT offers many benefits, it also introduces new cybersecurity risks and attack vectors that need to be addressed [9–11]. Some key security concerns with IoT communication include weak authentication mechanisms, unencrypted networks, insecure web interfaces, malware infections, DDoS attacks, and data privacy issues, where sensitive user data collected by IoT devices could be stolen and misused if not properly protected [11]. To secure IoT communications, both device manufacturers and end users need to take security seriously and implement good practices such as encrypting data flows between devices using protocols like DTLS and TLS, requiring strong passwords and multi-factor authentication to access devices and interfaces, regularly patching and updating IoT devices, securing IoT devices behind firewalls to prevent unauthorized access, monitoring connected devices and network traffic for anomalies, and developing incident response plans for IoT security breaches. Additionally, following standard security best practices, such as least privilege access, vulnerability management, and network segmentation, helps create layers of defense [12].

Given the wide spectrum of security threats faced by IoT systems, a holistic and multidimensional approach is essential to provide comprehensive protection. Securing IoT environments necessitates defense-in-depth approaches comprising several security measures that are in place to protect against different attack vectors. Sealing the risks posed by threat actors requires an IoT security framework that incorporates various security concerns like confidentiality, integrity, authentication, access control, and adaptive response approaches.

The neighbor discovery protocol (NDP) is a crucial protocol that is included in the IPv6 protocol suite. Its tasks include discovering the MAC address associated with an IPv6, rerouting packets from one router to another, duplicate address identification, finding routers on the network, and address resolution. NDP uses several ICMPv6 message types to conduct its functions, including neighbor solicitation (NS), neighbor advertisement (NA), router solicitation (RS), and router advertisement (RA). Security is a top priority due to NDP's inherent vulnerabilities in address resolution and auto-configuration, which attackers might exploit to infect devices, intercept traffic, or overload networks. Some of the main risks are traffic hijacking through the marketing of fake network prefixes, MITM attacks that impersonate address resolves, and the flooding of NS or NA messages.

The exponential rise in IoT devices has rendered IPv4 obsolete, and in its stead is the new protocol, IPv6. One benefit of IPv6 is that it has vast address space ($3.4 \times 10^{38}$), which makes it possible for the IoT to grow. This is because, for IPv4, 128-bit addresses are accessible rather than 32-bit addresses. By enabling auto-configuration, which allows devices to automatically generate IP addresses based on announcements from router prefixes, assigning IP addresses to IoT devices is considerably simpler. IPsec capabilities are necessary for IPv6 in order to ensure end-to-end security with improved authentication and encryption. Enhancements also provide better mobility, making it easier for devices like mobile phones to roam between different networks while maintaining connections. Additionally, IPv6 has streamlined processing with fixed-length packets and simplified headers, resulting in more efficient packet handling compared to IPv4. However, potential drawbacks of IPv6 include larger packet sizes, immature implementations with bugs, and the loss of NAT, which can expose devices to more threats.

The nuances of IoT communication merit protocols that are expressly designed with the intricacies of embedded devices and diverse networking mediums in mind, unlike conventional IP networks. IETF groups like 6lo and ROLL continue advancing open standards that smooth the integration of massively scaled IoT with the common internet infrastructure. As IoT permeates society and industry, standardized and interoperable protocols will grow even more crucial. Forward-thinking technical foundations like IPv6, 6LoWPAN, RPL, and related standards aim to fulfill that mission, delivering the communication capabilities necessary to unlock the possibilities of a connected world. Table 1 shows some of the key communication protocols used in IoT networks:

**Table 1.** Key communication protocols used in IoT networks.

| IoT Communication Protocols | Features |
|---|---|
| IPv6 | Expanded address space, enhanced security, simplified packet handling |
| 6LoWPAN | Adapts IPv6 for low-power IoT networks |
| RPL | Routing protocol optimized for constrained IoT devices |

The main research question addressed is how to design a comprehensive security framework that can dynamically detect and mitigate a wide range of attacks in IoT environments while maintaining availability and quality of service.

*Contributions*

The main contributions of this paper are summarized as follows:

- We propose a comprehensive and adaptive security framework called Enhanced SCAFFOLD tailored for IoT environments. The framework establishes secure communication channels between IoT nodes using software-defined networking (SDN) and machine learning techniques.
- Key components of Enhanced SCAFFOLD include channels encrypted using session keys, continuous traffic monitoring by the SDN controller, ensemble machine learning for attack detection, precision mitigation via SDN reconfiguration, and periodic reauthentication for freshness.
- A mathematical model is presented to formally define the protocol. Detailed pseudocode specifies the algorithms for key generation, encryption, attack detection, and mitigation.
- Extensive simulations are conducted to evaluate the performance of the Enhanced SCAFFOLD framework. The ability to reliably detect and rapidly mitigate various attacks with minimal latency and energy consumption overhead is demonstrated across diverse IoT network scenarios and traffic patterns.

The rest of the paper is organized as follows: Section 2 discusses the background. Related work is discussed in section 3. Section 4 discusses the IoT security and existing defense approaches. Section 5 provides an overview of the Enhanced SCAFFOLD protocol, detailing its components, modeling, and algorithms. Section 6 presents the simulation setup and performance analysis. Finally, Section 7 concludes the paper and outlines future research directions.

## 2. Background

The Background section provides an overview of the key concepts and technologies related to IoT security and communication. It introduces the NDP, IPv6 in IoT, congestion management, and modeling approaches that provide the groundwork for comprehending the Enhanced SCAFFOLD protocol.

NDP is a core protocol in IPv6 that facilitates communication between nodes, routers, and hosts within the same network segment. It relies on the Internet Control Message Protocol version 6 (ICMPv6) [13] to achieve its functionality. The key message types used in NDP are as follows:

- NS: Used to determine the link-layer address of a neighbor, verify the uniqueness of an address during the DAD process, or check the reachability of a neighbor.
- NA: Used to announce changes to a host's MAC and IP addresses or respond to NS message requests.
- RS: Sent by hosts to locate routers on the local link network and prompt an immediate response from the router.
- RA: Periodically sent by routers or in response to RS messages to announce their presence on the network and provide system parameters such as MTU, network prefix, and hop count.
- Redirect Message (RM): Used to redirect traffic from one router to another.

### *2.1. NDP in IPv6 Communications*

NDP enables key IPv6 communication functions, including address resolution, router discovery, redirects, and auto-configuration, as explained below:

- Address Resolution

To send a packet, the source must determine the destination's link-layer address. Table 2 outlines the NDP address resolution process:

**Table 2.** NDP address resolution process.

| Step | Description |
|---|---|
| 1 | Check cache for unexpired entry |
| 2 | Send NS if no entry via multicast |
| 3 | Target node replies with NA |
| 4 | Add mapping to cache |
| 5 | Forward frame using destination link-layer address |

- Router Discovery

Table 3 summarizes the router discovery process using NDP:

**Table 3.** NDP router discovery process.

| Step | Description |
|---|---|
| 1 | Hosts send multicast RS on bootup |
| 2 | Routers respond with RA messages |
| 3 | Hosts process prefixes, configuration from RAs |
| 4 | Default router selected based on RA info |

### *2.2. Security Issues with NDP*

While NDP is essential to IPv6, it lacks security and is vulnerable to attacks like:

- Neighbor cache poisoning using spoofed NDP messages;

- RA attacks by forging router advertisements;
- ARP spoofing equivalent by sending fake NAs;
- Denial of service (DoS) by flooding NDP messages;
- Replay attacks reusing captured NDP messages;
- Impersonation via NDP address spoofing.

### *2.3. Attacks on NDP*

Table 4 summarizes common NDP attacks and mitigation approaches:

**Table 4.** NDP attacks and mitigations.

| Attack | Description | Mitigation |
|---|---|---|
| **NS Flooding** | Overflowing target with NS messages | Rate limiting, validation |
| **NA Spoofing** | Faking NAs to poison cache | Cryptography like SEND |
| **RA Flooding** | Excess RAs consume resources | RA rate limiting |
| **Malicious RAs** | Wrong configuration and routes | RA filtering and monitoring |
| **Replay Attacks** | Reusing captured messages | Timestamps, sequence numbers |
| **IPv6 Reconnaissance** | Discovering devices via NDP | Isolation, firewall rules |

As an example, the NS flooding attack involves sending excessive NS messages to consume resources and cause instability or denial of service. As seen in Table 4, mitigations include rate limiting, source address validation, and cryptographic protection like SEND (SEcure Neighbor Discovery). Upgrades are required across networks to monitor, analyze, and secure NDP as IPv6 and IoT continue rapid growth. Robust NDP security will be critical for next-generation communications.

### *2.4. Petri Net Modeling*

Petri net [14] is a powerful modeling tool used to analyze and describe control and information flows in discrete-event systems with concurrent and asynchronous activities. It provides an intuitive graphical representation and a rigorous mathematical foundation for understanding the dynamic behavior of complex systems. A classical Petri net model consists of the following elements:

1. Places: Represented by circles, the places denote the states or conditions of a system.
2. Transitions: The events or activities that lead to a change in the state of the system are modeled by transitions, which are represented by rectangles.
3. Directed Arcs: Arrow arcs depict token flow from places to transitions and vice versa.
4. Tokens: Tokens indicate the existence of a condition or the achievement of a state; they are shown as dots inside gaps.

Petri nets are widely used to simulate and research a wide range of network security issues, including cyber-physical assaults and protocol problems [15]. Because of their formal semantics and graphical representation, Petri nets are a helpful tool for elucidating and drawing conclusions about the many linkages and interactions encountered in network security situations.

### *2.5. SDN*

Network programmability, manageability, and flexibility are to be enhanced by SDN [16], a novel idea that separates the control plane from the data plane. SDN centralizes control logic in a software-based controller, whereas the data plane consists of simple forwarding devices that follow controller commands. The McKeown et al.-proposed OpenFlow protocol [17] has become the de facto standard for data plane device-to-controller communication in SDN.

There are many advantages to the SDN architecture in terms of network security. Because the control logic is centrally located, SDN allows for a global view of the network and facilitates the adoption of standardized security standards throughout the network. Because SDN is programmable, it is feasible to dynamically alter security measures in response to changing network circumstances and to swiftly implement countermeasures against known dangers, including DDoS assaults. Figure 1, which shows the basic architecture of a SDN network, emphasizes the division between the control plane and the data plane.

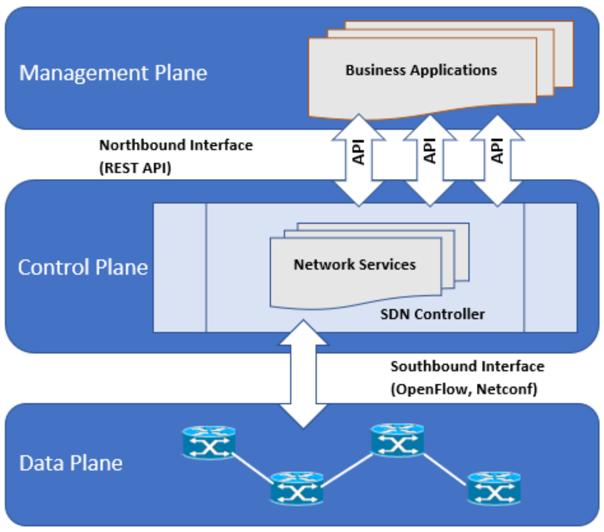

**Figure 1.** SDN architecture.

## 3. Related Work

Building upon the background concepts, the Related Work section surveys the existing literature on IoT security, focusing on NDP vulnerabilities, mitigation techniques, and broader defense strategies. This section identifies the gaps and limitations in current approaches, motivating the development of the Enhanced SCAFFOLD protocol.

The security of the NDP, a critical component of the IPv6 protocol suite, has been the subject of extensive research. Anbar et al. [18] conducted a comprehensive review of the vulnerabilities in NDP and categorized the attacks into two main types: man-in-the-middle (MITM) [19] attacks and distributed denial-of-service (DDoS) attacks. Their study provided valuable insights into the attack surfaces and potential countermeasures for securing NDP.

Zhang et al. [20] delved deeper into the security aspects of NDP and the Secure Neighbor Discovery (SEND) protocol. They meticulously analyzed the protection mechanisms offered by SEND and summarized the latest advancements in fortifying NDP security. Their experimental evaluation involved the implementation of the SEND mechanism, which demonstrated its effectiveness in mitigating various security issues in NDP. However, the authors noted that the widespread adoption of SEND remains a challenge due to its complexity and overhead.

Arjuman et al. [21] proposed an authentication framework to enhance the security of NDP. Their approach leveraged the multicast key management protocol as an application layer key management scheme to address the multicast problem in neighbor communication. The framework incorporated internet protocol security (IPsec), authentication header (AH), and media access control (MAC) address options in NDP to authenticate communication packets and prevent attacks involving forged ND messages. The authors demonstrated that their improved NDP security policy could effectively defend against various NDP security threats, including SYN flooding, forged prefix address attacks, and

ARP spoofing. Figure 2 illustrates the architecture of their proposed authentication framework.

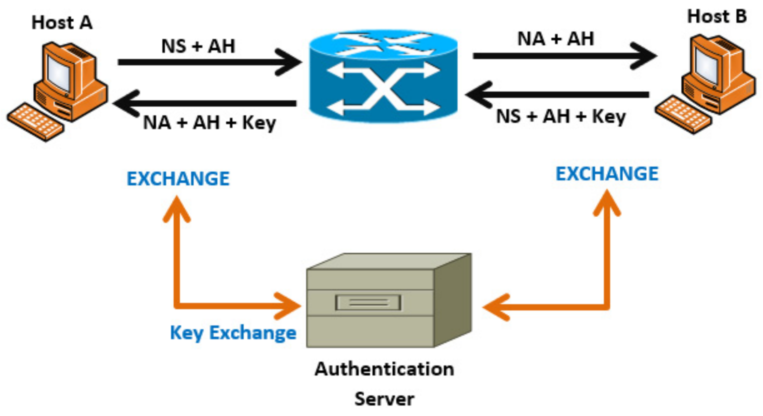

**Figure 2.** Authentication framework for secure NDP communication.

Match prevention, a preventive method first presented by Ahmed K. Al-Ani [22], focuses on protecting target IP addresses and exchanging messages, particularly NS and NA messages. Their plan aimed to reduce the impact of DoS attacks on the address determination and DAD processes of IPv6 link-local networks. The authors provided a comprehensive description of the match-prevention technique and used simulations and experiments to evaluate its effectiveness. In order to identify TCP SYN flood assaults, A. Q. Moghadam [23] investigated the use of entropy to detect the unpredictability of streaming data. Their method demonstrated the ability to rapidly detect such attacks.

Premarathne et al. [24] introduced hybrid cryptographic access control for cloud-based IoT applications. They used attribute-based encryption (ABE), symmetric encryption, and hash-based authentication to secure IoT data sharing and access control. The authors proved their scheme's security, performance, and scalability.

Zhou and Wang [25] used machine learning for performance index intelligent optimization and dynamic optimum allocation in UAV landing control. Their study showed that machine learning techniques may improve UAV control system efficiency and resilience.

For fault-tolerant distributed wireless sensor networks, Sai and colleagues [26] created a lightweight authentication scheme. Fault tolerance and secure WSN communication were achieved via the use of hash-based authentication and symmetric key encryption. The security, energy efficiency, and scalability of their platform were shown.

Elejla and colleagues [13] developed IDSs to identify ICMPv6-based DDoS attacks. Malicious behavior was detected via network data analysis using machine learning. The authors demonstrated that utilizing many metrics, their IDS could detect ICMPv6-based DDoS attacks.

For NDP protocol Man-in-the-Middle (MITM) assaults, Zhang et al. [14] proposed a Petri Net-based model. The complex connections and interconnectedness of MITM assaults were represented by formal modeling. The authors demonstrated how to find NDP MITM vulnerabilities and solutions using their approach.

Gao et al. [27] examined bridge repair using AIoT for digital twin communication. They used machine learning and IoT to allow smart and efficient communication between digital twins and physical assets. The authors showed that their technique improved defect identification, predictive maintenance, and system performance.

Table 5 compares the techniques and tactics in this literature review, highlighting their key features, pros, and cons.

**Table 5.** Comparison of methods and strategies for NDP security.

| Technique/Approach | Key Features | Advantages | Limitations |
|---|---|---|---|
| **SEND [20]** | Secure NDP, cryptographic mechanisms | Mitigates NDP attacks; provides authentication and integrity | Complexity, overhead, limited adoption |
| **Authentication Framework [4]** | Multicast key management, IPsec, AH, MAC options | Prevents forged ND messages; defends against various NDP attacks | Increased complexity, potential performance impact |
| **Match Prevention [22]** | Secures target IP addresses and exchange messages | Mitigates DoS attacks on address resolution and DAD | Limited scope, focuses on specific attack scenarios |
| **Entropy-based Detection [15]** | Detects randomness of flowing data | Rapid detection of TCP SYN flood attacks | Cumbersome implementation for DDoS detection |
| **Petri Net Modeling [14,28]** | Formal modeling, graphical representation | Describes complex interactions, enables security analysis | Requires expertise in Petri net modeling |
| **SDN-based Security [21,28]** | Centralized control, programmability | Global network view, dynamic security policies, rapid threat response | Potential single point of failure, scalability concerns |
| **Hybrid Crypto Access Control [24]** | Attribute-based encryption, symmetric crypto, hash-based authentication | Secure data sharing and access control in IoT | Complexity, key management overhead |
| **ML for UAV Control [25]** | Performance index optimization, dynamic optimal allocation | Enhances UAV control efficiency and robustness | Computational complexity, training data requirements |
| **Lightweight Auth for WSNs [26]** | Symmetric key crypto, hash-based auth | Secure comm. and fault tolerance in WSNs | Limited resources in WSNs |
| **ICMPv6 DDoS IDS [14]** | Machine learning for traffic analysis | Effective detection of ICMPv6 DDoS | False positives, training data requirements |
| **Petri Net Model for MITM [24]** | Formal modeling of MITM attacks in NDP | Identifies MITM vulnerabilities and countermeasures | Model complexity, scalability |
| **AIoT for Digital Twin Comm. [15]** | Machine learning, IoT technologies for digital twins | Intelligent and efficient digital twin communication | Computational complexity, data privacy |

Protecting the NDP in IPv6 networks is a major research problem. These studies provide a variety of ways to mitigate NDP-related risks and vulnerabilities. Researchers have developed entropy-based detection, match prevention, SEND, and authentication frameworks to increase NDP security.

Formal modeling technologies like Petri nets have also clarified NDP security problems' intricate links and interdependencies. Through centralized control, programmability, and a fast attack response, the new SDN paradigm may improve network security.

Table 2 lists each strategy's merits and cons. Some systems provide total protection against several threats, but they are expensive and complicated. Others may concentrate on certain attack scenarios. Choosing and implementing NDP security measures should take into account the network environment, security needs, and available resources.

### *Machine Learning in Civil Engineering and Structural Analysis*

Machine learning has proved effective in civil engineering and structural analysis. A chained machine-learning model by Shafighfard et al. [29] predicted a steel fiber-reinforced concrete beam's load capacity and ductility. Their approach demonstrated the effectiveness of machine learning in predicting complex structural properties.

Bagherzadeh et al. [30] conducted a comparative study on the prediction of maximum tensile stress in plain-weave composite laminates with interacting holes using stacked machine-learning algorithms. Their work highlighted the potential of machine learning in analyzing complex composite structures.

Asgarkhani et al. [31] utilized machine-learning methods to predict the seismic response and performance of steel buckling-restrained braced frames. Their research showcased the applicability of machine learning in earthquake engineering and structural performance assessment.

## 4. Secure IoT Communication and Attack Mitigation

Having established the background and related work, this section delves into the core components of secure IoT communication and attack mitigation. It discusses the role of IPv6 in IoT, congestion control mechanisms, and modeling techniques for analyzing IoT attacks and defense.

### *4.1. IPv6 Protocol and IoT Communications*

IoT refers to the billions of internet-connected devices and sensors that collect and share data. To support the massive growth of IoT devices, the transition from the IPv4 protocol to the newer IPv6 protocol is essential. IPv6 adoption is accelerating globally for IoT implementations due to the benefits of IPv6, such as expanded addressing, simplified packet handling, and built-in security.

#### 4.1.1. IPv6 Adoption for IoT

The primary driver for IPv6 adoption in the IoT is the need for more IP addresses. IPv4 provides only 4.3 billion addresses, which are now fully allocated, while IPv6 expands the address space to 340 undecillion addresses to support massive IoT growth. Table 6 highlights the projected growth indicators for IoT devices using IPv6.

**Table 6.** Growth indicators for IPv6 IoT adoption.

| Indicator | Projection |
|---|---|
| **Cellular IoT connections over IPv6** | 98% CAGR from 2020 to 2025 |
| **Global IPv6 IoT devices** | >50% by 2023 |

As seen in Table 6, cellular IoT connections using IPv6 are forecast to grow at a 98% CAGR from 2020 to 2025, topping 4.8 billion in 2025, as per ABI Research. Also, by 2023, over 50% of global IoT devices are expected to use IPv6, as per ABI Research. IPv6 adoption for the IoT brings other key benefits including simplified packet handling using fixed-length 128-bit addresses, no more network address translation (NAT), built-in IPsec security, better support for mobile devices, efficient multicasting and broadcasting, and extensibility for new features. Leading technology organizations like Google, Facebook, Akamai, Verisign, and Cisco have been early adopters of IPv6, encouraging global IPv6 deployment.

#### 4.1.2. IPv6 Addressing for IoT Devices

The 128-bit IPv6 address provides a huge addressing space for IoT deployment. IPv6 addresses have a format like 2001:0db8:85a3:0000:0000:8a2e:0370:7334 consisting of eight groups of 16-bit hexadecimal values separated by colons. The 128 bits allow for $2^{128}$ = 340 undecillion unique addresses. Table 7 summarizes the advantages of the expanded addressing capability in IPv6.

**Table 7.** Advantages of expanded addressing in IPv6.

| Advantage | Description |
|---|---|
| **Addresses global IoT device growth** | Supports massive number of IoT devices |
| **Efficient address auto-configuration** | Using SLAAC |
| **Hierarchical allocation** | For route aggregation |
| **Simplified addressing** | No NAT required |
| **Enhanced mobility** | End-to-end IP connectivity |

As shown in Table 6, the huge addressing space addresses the global IoT device explosion, allows simplified addressing without NAT, improves device mobility with end-to-end IP connectivity, enables efficient auto-configuration, and allows hierarchical allocation for efficient routing. There are various types of IPv6 addresses allocated for IoT devices, including global unicast addresses (GUA), unique local addresses (ULA), link-local addresses (LLA), multicast addresses, and anycast addresses. GUAs are most relevant, as IoT devices need global connectivity. GUAs have a global routing prefix and a 64-bit interface ID.

#### 4.1.3. IPv6 Communication Infrastructure for IoT

Deploying IPv6 for the IoT requires upgrades across network infrastructure, including routers, switches, firewalls, network management systems, and the DNS. Dual-stack technology, supporting both IPv4 and IPv6, is commonly used during the transition period. Tunneling mechanisms like 6over4, 6in4, and 6RD allow IPv6 packets to run over existing IPv4 networks. Cellular networks have rapidly adopted IPv6 to support the growth of cellular IoT connections, using carrier-grade NAT to handle IPv4 address exhaustion. IoT platforms and services also need to evolve to support IPv6 connectivity with devices. Cloud, analytics, and application platforms enable the leveraging of sensor data. IoT devices like sensors, cameras, appliances, etc., need IPv6-capable network stacks to connect over the IPv6 infrastructure.

There are some challenges in IPv6 deployment, including knowledge gaps, buggy IPv6 stack implementations, a lack of IPv6-compliant services, and perceived security concerns. Thorough planning, testing, and training are required for smooth IPv6 deployments, although the long-term benefits outweigh the initial hurdles for supporting massive-scale IoT growth.

The expanded addressing capability and advanced functions of IPv6 deliver critical benefits for connected IoT devices and platforms. The progressive deployment of IPv6 across networks, services, and devices will empower tremendous IoT innovations in the future.

### *4.2. Congestion Control in IoT Networks*

With the rapid growth of IoT devices and applications, managing network congestion has become a critical issue. Congestion occurs when too many packets flood the network, overloading nodes and links and leading to performance degradation. Without effective congestion control, IoT systems can experience increased delays, packet losses, and the blocking of connections. This section provides an overview of techniques for congestion detection, avoidance, and mitigation in IoT environments.

#### 4.2.1. Congestion Detection

Detecting the onset of congestion is the first step toward activating congestion control mechanisms. IoT networks exhibit unique traffic patterns and requirements compared to traditional networks. Hence, customized congestion detection approaches are necessary. Table 8 shows the key indicators for IoT congestion detection, including the following:

- Buffer Occupancy: Monitoring buffer utilization at the network nodes can signal congestion. Thresholds can be set to trigger a congestion response when the buffers become full. For example, the CoAR protocol uses current and historical buffer occupancy to detect congestion [1].
- Traffic Load: Observing channel load conditions over time also reveals congestion. The CoAP-R algorithm utilizes present and past traffic loads for congestion detection [2].
- Latency: Increasing delays in packet delivery indicate possible congestion. Some solutions, like adaptive VBS [3], use changes in latency as a congestion marker.
- Packet Loss: When buffers overflow at the congested nodes, packets will be dropped. Monitoring packet loss rates can thus identify congestion.
- Queue Length: Longer queues at the nodes imply traffic accumulation and possible congestion. Techniques like FLCC [21] employ queue length to detect congestion.

**Table 8.** Key indicators for IoT congestion detection.

| Congestion Detection Technique | Description |
|---|---|
| **Buffer Occupancy** | Monitoring buffer utilization at nodes |
| **Traffic Load** | Observing channel load over time |
| **Latency** | Increasing delays in packet delivery |
| **Packet Loss** | Dropped packets due to buffer overflow |
| **Queue Length** | Longer queues indicate traffic accumulation |

Buffer occupancy is commonly used, as it directly signals the buildup of packets at the nodes. A combination of such indicators provides a comprehensive congestion detection system. Machine-learning methods can also be applied to learn the congestion patterns.

### 4.2.2. Congestion Avoidance

Congestion avoidance aims to prevent the network from entering a congested state. It includes proactive techniques like:

- Traffic Shaping: Proactively smoothing out traffic bursts through buffering and rate control avoids sudden traffic spikes that lead to congestion.
- Load Balancing: Distributing traffic across multiple paths avoids overloading particular links and nodes. This prevents localized congestion hotspots.
- Adaptive Transmission: Nodes can dynamically adjust the transmission rates based on the network conditions to stay within the capacity limits.
- Prioritized Traffic: Giving preferential treatment to certain traffic types, such as real-time flows, ensures critical data moves through without congestion delays. Caching and Offloading: Caching popular IoT data at the edge of the network alleviates traffic in the core network, avoiding congestion. Computation offloading also reduces network loads.
- Caching and Offloading: Caching popular IoT data at the edge of the network alleviates traffic in the core network, avoiding congestion. Computation offloading also reduces network loads.

Machine-learning predictive models can forecast traffic and detect anomalies to activate avoidance measures before congestion sets in. However, this requires collecting sufficient training data. Overall, a combination of avoidance tactics is necessary for robust congestion control.

### 4.2.3. Congestion Mitigation

After congestion occurs, mitigation techniques help restore network performance and stability. Common mitigation methods include the following [32]:

- Rate Limiting: Nodes temporarily reduce transmission rates to relieve the congested paths when congestion is detected.
- Load Redistribution: Rerouting subsets of traffic to underutilized paths alleviates the overloaded routes. SDN controllers can facilitate global load redistribution.
- Queue Management: Active queue management drops low-priority packets before the buffers become full in order to control congestion.
- Window Limiting: Reducing the sliding window sizes limits how much new data can enter the network during congestion.
- Scheduling and Admission Control: Intelligently scheduling packet transmissions and limiting new flows prevents oversubscription.

Rate limiting provides immediate congestion relief but can impact throughput. Combining multiple mitigation strategies balances performance, reliability, and congestion responsiveness in IoT systems. Congestion control is critical for IoT networks to maintain stability and meet quality-of-service needs. IoT protocols and architectures must evolve to include native congestion management capabilities. Machine learning also offers new possibilities to make congestion control adaptive and predictive. As the IoT scales massively, optimized congestion control will be essential to delivering robust performance.

## *4.3. Modeling IoT Attacks and Defenses*

IoT systems face a variety of security threats that can disrupt communications and cause network congestion. To analyze and mitigate these risks, formal modeling techniques can be used to represent attacks and defense strategies. This section reviews the modeling approaches for IoT security and proposes a new defense protocol based on a combination of SDN and machine-learning techniques.

### 4.3.1. Petri Nets for Modeling IoT Attacks

Petri nets provide a graphical and mathematical modeling framework well-suited for analyzing distributed, concurrent systems like the IoT. Petri nets consist of places, transitions, and directed arcs connecting them. Places represent possible states, transitions depict actions or events, and tokens in places denote the current state. Petri net models capture the dynamic and distributed interactions between IoT components under normal conditions and attacks [28].

Petri nets have been applied to model various IoT attacks, including denial-of-service, man-in-the-middle, and malicious code injection [1]. For instance, [26] used stochastic Petri nets to model a code injection attack against an IoT middleware platform. The attack stages were represented by transitions and the system's vulnerable states as places. Probabilistic reasoning on the Petri net evaluated attack impacts and defense strategies. We utilize Petri nets to model the neighbor solicitation (NS) flooding attack in IPv6 networks. The NS flooding attack exploits the NDP, which relies on NS and neighbor advertisement (NA) messages to map IP addresses and MAC addresses. Table 9 describes the places and transitions in the NS flooding Petri net.

**Table 9.** Description of places and transitions in the NS flooding Petri net.

| Place | Description |
|---|---|
| **P0** | Normal state |
| **P1** | Attacker has victim's IP address |
| P2 | Spoofed NS messages generated |
| **P3** | Flooding with NS messages |
| **P4** | Attack disrupts legitimate traffic |
| **Transition** | **Description** |
| **T0** | Attack initiated |
| **T1** | Sniffing victim's IP address |

| | |
|---|---|
| **T2** | Generating spoofed NS messages |
| **T3** | Flooding victim with NS messages |

The attacker sends high volumes of NS messages with spoofed source IP addresses to overwhelm the target nodes. The Petri net model for the NS flooding attack is shown in Figure 3. The places correspond to the node states during an attack. P0 is the normal state. In P1, the attacker has obtained the victim's IP address through sniffing. P2 represents the state where spoofed NS messages have been generated. Flooding occurs in P3 as the victim is inundated with requests. P4 represents the failed state as the attack prevents legitimate traffic. The transitions model the stages of the attack. T0 initiates the attack, T1 sniffs the victim's address, T2 generates spoofed messages, and T3 floods the victim. Tokens denote the current state, marking the progress of the attack [9].

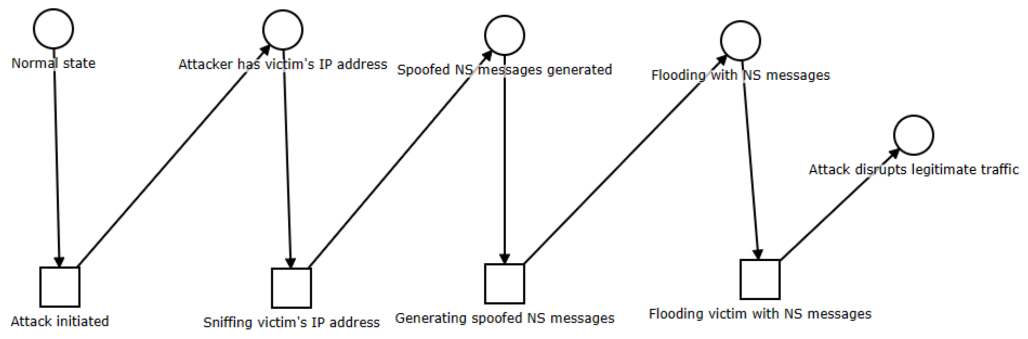
Normal state
Attacker has victim's IP address
Spoofed NS messages generated
Flooding with NS messages
Attack disrupts legitimate traffic
Attack initiated
Sniffing victim's IP address
Generating spoofed NS messages
Flooding victim with NS messages

**Figure 3.** Petri net model for NS flooding attack in IoT networks.

This Petri net analysis precisely captures the attack sequence. The token movements reveal vulnerabilities and suggest countermeasures such as encrypting NDP traffic or detecting address spoofing. Extensions could include stochastic transitions to model the uncertainty and defense mechanisms as additional places and transitions.

### 4.3.2. Defense Strategies for IoT Networks

Along with attack modeling, effective defense mechanisms are required to protect IoT systems. SDN and machine learning are two emerging paradigms for securing IoT environments [10]. SDN provides centralized control and management of the network operations by decoupling the control and data planes [24]. This allows dynamic security policies to be enforced network-wide. The SDN controller monitors traffic, detects anomalies, and modifies routing to counter the attacks.

We utilized SDN concepts to defend against the NS flooding attacks in IoT networks, as shown in Figure 4. The victim and attacker nodes connect through an SDN-enabled switch. The SDN controller monitors traffic statistics such as flow table usage, unique source IP addresses, and NS message rates. These metrics are analyzed by a controller-based machine-learning-based attack detector, which looks for anomalies that point to an NS flooding attack. Effective threat identification and mitigation are made possible by the combination of SDN and machine learning. SDN enables dynamic network reconfiguration in reaction to the identified risks and offers fine-grained visibility into network traffic patterns. Real-time machine-learning algorithms are capable of effectively identifying anomalies and malicious behaviors since they have been trained on historical attack data and typical traffic patterns. SDN's centralized control and programmability let the machine-learning-based attack detector identify NS flooding threats rapidly. It may also instruct the SDN controller to create flow rules that filter hostile traffic while letting legitimate traffic through. The effect on normal network operations is lessened, and the availability of essential IoT services is ensured by using a customized mitigation strategy [17].

Combining SDN with machine learning has many advantages for IoT network security detection and mitigation. First and foremost, SDN allows the gathering of extensive traffic and network telemetry data, resulting in a large dataset that machine-learning algorithms may evaluate to detect anomalies. Second, since SDN has a centralized control plane, countermeasures may be implemented quickly if an attack is identified. The SDN controller may dynamically adjust the network's flow rules to isolate the impacted devices and block malicious traffic. Third, machine-learning algorithms can continually discover and adapt to new attack patterns, hence boosting threat detection accuracy and efficiency over time. When IoT attackers change their tactics, machine-learning models may be retrained on fresh data to keep up with the developing threats. However, the controller may interfere with performance. The efficacy of attack detection is determined by the precision of the machine-learning algorithm.

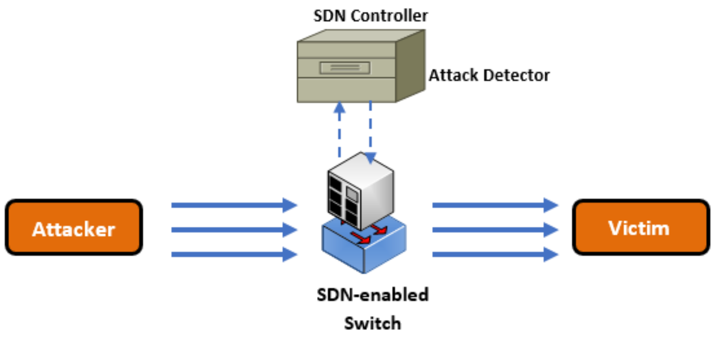

**Figure 4.** Using SDN and machine learning for defense against NS flooding.

## 5. Enhanced SCAFFOLD Protocol

The Enhanced SCAFFOLD Protocol section presents the proposed security framework in detail. It describes the key features, mathematical model, protocol messages, and algorithms of Enhanced SCAFFOLD. This section highlights how the protocol integrates SDN, machine learning, and cryptographic techniques to provide comprehensive IoT security.

This section covers the improved SCAFFOLD protocol. It establishes secure communication channels between IoT nodes and uses machine learning and SDN to dynamically identify and mitigate hazards. SCAFFOLD combines SDN programmability, encryption, flow data mining, and controller monitoring to offer adaptive and autonomous security for IoT installations. To strengthen security and resilience against ever-changing IoT threats, this enhanced version integrates additional concepts from related protocols like DietTOP, IKEv2, and HIP [25].

There are many security vulnerabilities that potentially compromise the privacy, availability, and integrity of IoT systems. Among these dangers include eavesdropping, data manipulation, spoofing, denial-of-service attacks, and unauthorized access. To address these many risks in a thorough manner, Enhanced SCAFFOLD utilizes a defense-in-depth strategy with several facets. Through the integration of essential security measures at many levels, the framework provides total protection. Enhanced SCAFFOLD uses encryption and session keys to provide secure channels at the communication layer, ensuring the confidentiality and integrity of data transmissions. The SDN control plane constantly examines network traffic to guarantee availability by spotting anomalies and swiftly thwarting assaults. Ensemble machine learning enables intelligent threat detection by analyzing traffic patterns and identifying potential breaches. While maintaining service for approved traffic, the SDN controller identifies suspicious flows and surgically separates them. Reauthentication and frequent key refreshes gradually reduce the effect of any compromised node. IoT systems are shielded from various threats that jeopardize their availability, confidentiality, and integrity by many levels of encryption, monitoring, intelligent detection, precise reaction, and continuous hardening.

To avoid expensive public key cryptography, the nodes perform a low-cost preliminary key exchange prior to channel construction. Next, the nodes create secure channels using nonces and session keys derived from their symmetric keys. AES-256 encryption is used to secure every connection, and it is routinely rekeyed to lessen the effect of key breaches. The SDN controller passively and constantly monitors network flows using OpenFlow. Relationships between the flows, classified as assaults by a group of machine-learning classifiers, are found through correlation analysis.

This enables precision mitigation, targeting specific attack sources and behaviors via SDN reconfiguration. Periodic reauthentication resets the session keys to maintain freshness.

This multifaceted approach provides defense-in-depth securing for IoT systems at different layers. Encryption and authentication prevent eavesdropping and spoofing. Controller monitoring and machine learning deliver intelligent attack detection. SDN control enables surgical mitigation while maintaining availability. Periodic rekeying and reauthentication limit the impact of any node exposure. SCAFFOLD integrates essential security capabilities, including confidentiality, integrity, authorization, attack detection, and adaptive response. Our enhancements augment robustness and efficiency through standards-based cryptography, decentralized lightweight keys, ensemble learning, flow correlation analysis, and incremental session refreshment. This combination of mechanisms tailored for the IoT environments comprehensively protects system availability, reliability, and trust. We mathematically modeled the network components and defined the protocol message sequences. Detailed pseudocode precisely specifies the algorithms. Performance metrics quantify the security, overhead, and efficacy to facilitate analysis under different scenarios. By comprehensively addressing IoT threat vectors, the Enhanced SCAFFOLD protocol represents a systematic, expandable framework for end-to-end security in modern and emerging IoT systems.

Random forests (RF), support vector machines (SVM), and recurrent neural networks (RNN) are used in the Enhanced SCAFFOLD framework to identify IoT network traffic abnormalities and threats. Ensemble learning improves detection robustness and accuracy by combining several classifier predictions [1].

Random forests, an ensemble of decision trees, prevent overfitting and increase generalization via feature randomization and bagging [2]. SVMs find suitable hyperplanes in high-dimensional feature spaces to distinguish anomalous and normal traffic [3]. Temporal correlations in network traffic and attack patterns may be detected using LSTM RNNs [4].

Enhanced SCAFFOLD integrates several classifiers' complementing properties to detect DDoS, MITM, reconnaissance, and malware. RF and SVM excel at finding spatial abnormalities, while RNNs catch seasonal patterns.

The method also employs correlation analysis to find negative communication flows for the company. This simplifies identifying coordinated attacks and the security events' causes. By examining the links between the payload features, protocols, and source and destination IP addresses, Enhanced SCAFFOLD may improve attack pattern knowledge and mitigation tactics [5].

### *5.1. Protocol Overview*

The following procedures are used by the Enhanced SCAFFOLD to provide safe communication paths between IoT devices:

**STEP 1** Preliminary lightweight key exchange based on HIP and DietTOP.
**STEP 2** Secure channel creation using nonces and session keys.
**STEP 3** Traffic encryption using AES-256 and periodic rekeying.
**STEP 4** Continuous traffic monitoring by the SDN controller.
**STEP 5** Attack detection via ensemble classifiers and correlation analysis.
**STEP 6** Precision attack mitigation through SDN reconfiguration.

**STEP 7** Periodic channel reauthentication inspired by IKEv2.

These mechanisms enhance security, resiliency, and performance.

### *5.2. Mathematical Model*

The Enhanced SCAFFOLD procedure is formally modeled using mathematical terminology and structures. The sets represent the fundamental components of a network, including switches, nodes, keys, and flows. Functions denote operations like encryption, feature extraction, and classification. The mathematical representations concisely capture the properties and relationships. For example, session keys are defined as pseudo-random functions of nonces and node keys. Encryption and decryption are modeled by applying keys to plaintext or ciphertext. Feature extraction and ensemble classification transform the traffic flows into predicted labels. This formalization complements the protocol definition by abstracting the implementation details into concise mathematical expressions. The model facilitates a rigorous analysis of protocol security and performance. Properties can be proven mathematically and understood through set relations. By bridging the natural language definitions to mathematical foundations, the modeling enables formal reasoning about protocol behavior. Extensions and modifications to the protocol can be represented and validated within the mathematical framework. The mathematical model provides a tool for precision in the specification, analysis, and evolution of the Enhanced SCAFFOLD protocol.

We can further formalize SCAFFOLD mathematically. Let:

- F = {f1, f2, ..., fN} = Set of network flows;
- N = {n1, n2, ..., nN} = Set of IoT nodes;
- K = {K1, K2, ..., KN} = Set of symmetric keys derived from secret seeds;
- S = {s1, s2, ..., sM} = Set of SDN switches;
- A = {a1, a2, ..., aP} = Set of possible attacks;
- PK = {PK1, PK2, ..., PKX} − Set of public keys;
- SK = {SK1, SK2, ..., SKX} − Set of private keys;
- PKi—Public key of node ni;
- SKi—Private key of node ni;
- Ki—Symmetric key of node ni;
- Kij—Shared symmetric key between nodes ni and nj;
- E(m,k)—Encrypt message m with key k;
- KDF—Key derivation function;
- SKij—Session key for the channel between ni and nj;
- P—Plaintext packet;
- C—Ciphertext packet;
- AES256—AES-256 encryption algorithm;
- D(c,k)—Decrypt ciphertext c with key k.

➢ **Preliminary Key Exchange**

Node ni generates an asymmetric key pair (PKi, SKi).
To establish the initial shared key Kij with nj:
nj → ni: E(PKj, PKi) using ni's public key;
Kij = PRFK(PKj || SKi), where PRFK is a pseudo-random function.

➢ **Secure Channel Creation**

ni generates a 128-bit nonce Nni = IDi || Rndi, where Rndi is a random value.
Session key derivation:
SKij = PRFK(Nni || Kj).

➢ **Traffic Encryption**

Encrypted traffic between ni and nj:
Ci,j = AES256(Pi,j, SKij);

where Pi,j is the plaintext traffic and Ci,j is the ciphertext. The key is renewed every 60 s.

- **Attack Detection**

  For a flow fk:
  Feature extraction: $\varphi$(fk) → Fk;
  Ensemble classification: E(Fk) → yk.
  where
  $\varphi$ = Feature extraction function;
  Fk = Extracted feature vector;
  E = Ensemble of classifiers (RNN, RF, SVM);
  yk = Predicted label ("Normal" or "Attack");
  If yk == "Attack", an alarm is triggered.

- **Attack Mitigation**

  Identify correlated attack flows Fattack ⊆ F based on the analysis.
  The controller installs the SDN rules to block Fattack.

## 5.3. Protocol Messages

The Enhanced SCAFFOLD protocol consists of a sequence of structured messages exchanged between the IoT nodes to establish secure communication channels. The protocol messages encapsulate data, including public keys, nonces, session keys, and node identities. The messages are formally specified using a consistent notation that defines the content and encryption mechanism. For example, the preliminary key exchange involves the nodes sending their public keys encrypted with the recipient's public key. The channel request and grant messages allow the creation of session keys using encrypted nonces. Periodic reauthentication resets the session keys to provide freshness. This messaging notation unambiguously defines the protocol sequence, data transmitted, and encryption requirements. The formal definition facilitates implementation and interoperability. Rigorous security analysis can verify that the protocol messages are constructed properly and encrypted using the authorized keys. Any deviations represent a potential violation. The clearly defined protocol messaging syntax and semantics aid in translating the high-level protocol into a concrete implementation.

- **Preliminary Key Exchange**

  ni → nj: EN_nj(PKi, PKj);
  nj → ni: EN_ni(PKj, PKi).

- **Channel Request**

  ni → nj: EN_ij(IDi || Nni, Kij).

- **Channel Grant**

  nj → ni: EN_ij(Nni, SKij).

- **Periodic Reauthentication**

  ni → nj: ni, Nni;
  nj → ni: EN_ij(SKij_new, SKij_old);
  where EN_k denotes encryption with key k.

## 5.4. SCAFFOLD Algorithm

The operation of the Enhanced SCAFFOLD protocol is precisely defined through high-level algorithm pseudocode. Figure 5 shows an abstract specification in a Python-like language that concisely captures the key procedures executed by the IoT nodes, SDN controller, and network switches to establish secure communication channels, encrypt data, detect attacks, and mitigate threats. The pseudocode spans the major protocol components, including preliminary key exchange, secure channel creation, traffic encryption, controller monitoring, attack detection via ensemble learning, precision mitigation enabled by SDN, and periodic reauthentication. Modular functions describe the critical steps within each stage, such as generating keys, deriving session keys, encrypting packets, extracting flow features, classifying flows, and installing blocking rules. The comment text provides additional insights into the logic. This well-structured pseudocode provides a high-level definition of the Enhanced SCAFFOLD protocol, complementing the detailed mathematical model and formal protocol specification. Execution of the pseudocode algorithms helps validate the design and surface potential issues prior to implementation.

```
# Preliminary key exchange
ni:
  PKi, SKi = generateKeys()
  send(nj, encrypt(PKi, nj.publicKey))
nj:
  PKj = receiveDecrypt(PKi)
  Kij = KDF(PKj || SKj)
# Secure channel creation
ni:
  Nni = IDi || randomNonce()
  SKij = KDF(Nni || Ki)
  pkt = encrypt(Nni, Kij)
  send(nj, pkt)
nj:
  Nni = decrypt(pkt)
  SKij = KDF(Nni || Kj)
  pkt = encrypt(Nni, SKij)
  send(ni, pkt)
# Traffic encryption
pkt = plaintextData
ctxt = encrypt(pkt, SKij)
send(ctxt)
# Periodic rekeying
Every 60 seconds:
  SKij = generateKey()
# Controller monitoring
flows = pollSwitches()
for f in flows:
  extractFeatures(f)
  ensembleClassify(f)
  if attack:
    correlateFlows(f)
# Attack mitigation
```

**Figure 5.** Enhanced SCAFFOLD algorithm.

### 5.4.1. Ensemble Classifiers

Enhanced SCAFFOLD utilizes an ensemble of machine-learning classifiers to analyze the network flows and identify potential attacks. An ensemble classifier combines the predictions of multiple individual models to make a final classification decision. This approach often yields better accuracy and robustness compared to using a single model.

The ensemble used in Enhanced SCAFFOLD consists of three popular machine-learning algorithms as follows:

1. Random Forests (RF): Random forests are an ensemble learning method that constructs multiple decision trees during training and outputs the class, which is the mode of the classes predicted by the individual trees. It is well known that random forests can handle high-dimensional data, minimize overfitting, and produce feature importance metrics.
2. Support Vector Machines (SVM): SVMs are supervised learning models that analyze data for classification and regression analysis. They construct a hyperplane or set of hyperplanes in a high-dimensional space to separate different classes. SVMs are effective in handling nonlinearly separable data by using kernel tricks.
3. A family of artificial neural networks called recurrent neural networks (RNNs) is made to process sequential data. They preserve an internal state that enables them to identify patterns and temporal connections in the supplied data. In applications involving time series data, RNNs—especially variations like long short-term memory (LSTM) and gated recurrent units (GRU)—have demonstrated remarkable performance.

The Enhanced SCAFFOLD's ensemble classifier uses methods like majority voting and weighted averaging to integrate the predictions of these three models. Through the use of the benefits of each individual model, this ensemble approach provides a deeper analysis of the network flows.

### 5.4.2. Correlation Analysis

In addition to the ensemble classifier, Enhanced SCAFFOLD makes use of correlation analysis to identify relationships between the flows that are flagged as assaults. Finding relationships, patterns, and commonalities between various attack scenarios is the aim of correlation analysis.

The SDN controller receives flow information and applies a range of correlation techniques to identify the links between suspicious flows. Common methods for correlation include the following:

1. The coefficient of Pearson correlation: With values ranging from -1 (a perfect negative correlation) to +1 (a perfect positive correlation), this gauges the linear correlation between two variables. It assists in locating flows that behave similarly with regard to traffic volume, packet size, or other characteristics.
2. Cross-correlation examines the distances between two time series to determine similarity. It helps to discover time-delayed links between attack flows, which might indicate planned or coordinated assaults.
3. The rule of association identifying significant correlations between variables involves analyzing vast datasets via mining. In order to highlight the co-occurrence of certain flow features in attack situations, it may identify frequently occurring item sets and provide association rules.

Correlation analysis on labeled attack flows helps Enhanced SCAFFOLD find linked attacks and hidden patterns. This information improves the mitigation strategies by increasing our awareness of the attacks' scope and kind.

Using correlation analysis and ensemble classifiers, Enhanced SCAFFOLD can discover and assess IoT hazards. The ensemble classifier effectively identifies threats, and the correlation analysis reveals attack flow patterns. The multidimensional technique increases the system’s ability to identify complex and intentional attacks via precise SDN reconfiguration responses.

Advanced machine-learning technologies like Enhanced SCAFFOLD will be needed to secure and avoid emerging threats in increasing IoT networks. These methods’ flexibility and learning ability enable the system to stay ahead in a changing threat scenario while improving diagnostic accuracy.

### *5.5. Multidimensional Defense-in-Depth Approach*

Using a layered defense-in-depth strategy, the Enhanced SCAFFOLD architecture solves several IoT security threats. This solution incorporates many security methods for comprehensive protection. Basic aspects of this multimodal method include the following:

1. The design employs encryption techniques like AES-256 to provide safe communication between IoT devices. Data are protected from unwanted access during network transfers.
2. Integrity: Enhanced SCAFFOLD can be used for secure data transfers between IoT nodes utilizing MACs and digital signatures. Tampering and data modifications during transmission may be fixed.
3. The framework employs robust authentication mechanisms, including mutual authentication and digital certificates to restrict access to the IoT to authorized devices. Impersonation assaults and unauthorized access are reduced.
4. Enhanced SCAFFOLD provides precise access control using attribute- and role-based models. This limits users’ and IoT devices’ responsibilities, characteristics, actions, and resource access.
5. Detecting Anomalies: The system uses machine-learning methods like ensemble classifiers and correlation analysis to identify security vulnerabilities. Proactive assault detection and reaction are possible.

Enhanced SCAFFOLD can quickly respond to attacks with SDN. SDN controllers may swiftly rebuild networks to separate affected devices and decrease attack damage. Enhanced SCAFFOLD integrates many security capabilities for a complete defense-in-depth approach. This comprehensive IoT security plan addresses network-based assaults, data manipulation, unlawful access, device impersonation, and data breaches.

SDN-based network reconfiguration, real-time threat detection, and response capabilities make the framework nimble, protecting IoT devices from new attacks. Encryption, access control, anomaly detection, and adaptive response can safeguard IoT settings.

## **6. Performance Analysis**

Detailed simulations assess the Enhanced SCAFFOLD protocol's efficacy and efficiency in Performance Analysis. Results include detection accuracy, attack mitigation delay, channel latency, and energy usage. Current performance evaluation constraints and problems are also discussed here.

OPNET Modeler thoroughly simulated the Enhanced SCAFFOLD protocol for testing. IoT network design had three distributed SDN controllers, twenty switches, and 300 nodes. The nodes had sensors, actuators, computers, smartphones, and embedded devices. Machine learning modules, including the attack detection ensemble classifier, were not simulated in the present performance investigation. Wired, LTE, WiFi, and other features were available on the network. A variety of background traffic patterns were simulated, including video conferences, VoIP conversations, HTTP, FTP, SSH sessions, SMTP, and custom IoT application protocols. Additionally, a variety of attacks were included, such as malware infections, remote code execution, man-in-the-middle (MITM), distributed denial-of-service (DDoS), and reconnaissance probes. During the simulations, dynamic traffic loads were imposed, and the nodes were designed with mobility patterns.

It is important to note that the current performance analysis focuses on evaluating the Enhanced SCAFFOLD protocol in isolation, without a direct comparison to other existing SCAFFOLD protocols. While a comparative analysis would provide valuable insights into the relative performance and advantages of Enhanced SCAFFOLD, it is beyond the scope of this paper due to the differences in design choices, target environments, and security objectives among various protocols. The current evaluation aims to demonstrate the effectiveness and efficiency of Enhanced SCAFFOLD's unique features, such as the ensemble machine learning classifier, SDN-based mitigation, and lightweight cryptographic mechanisms. Future studies can build upon this foundation to conduct comprehensive comparative analyses with other state-of-the-art SCAFFOLD protocols.

### *6.1. Detection Accuracy*

The investigation of detection accuracy demonstrated that the ensemble classifier technique of Enhanced SCAFFOLD allowed for dependable attack detection across a variety of threats and traffic kinds. Nevertheless, depending on the particular network circumstances and attack features, the detection efficiency differed. While more subtle attacks like malware and surveillance probes posed more obstacles, more overt threats like DDoS were detected with extremely high accuracy. The findings reveal areas that might be improved by incorporating external threat information, increasing feature extraction, and adding more training data. Based on its current performance, the framework may not be fully developed or optimized, and more research and development are required to increase its accuracy, recall, and durability.

There are a few methods with which to overcome these limitations. First, the ensemble classifier would discover more patterns and behaviors if the training dataset comprised more attack scenarios and network conditions. It would be easy to generalize and identify new dangers. Second, picking the most discriminative features and exploring cutting-edge approaches, like deep learning-based feature learning, may increase the classifier’s capacity to identify genuine malicious traffic. Third, integrating external threat intelligence streams and exchanging attack data with other businesses would provide important extra context for model training and threat detection.

Further tuning of each ensemble model and researching various techniques or architectures may improve performance. The optimum settings for each model may be found, and overfitting can be decreased with the use of methods like hyperparameter optimization, cross-validation, and model pruning by updating and retraining the models on the most recent assault on a regular basis. Additionally, data are essential to guarantee their continued efficacy as the threats change over time.

Adaptive detection skills will be essential to spot new attack patterns in a variety of scenarios as IoT use increases. The Enhanced SCAFFOLD framework may constantly grow and enhance its accuracy, recall, and resilience over time by improving the machine learning models, optimizing feature extraction, using external intelligence, and increasing the training dataset. Continually assessing its performance and refining solutions based on input from the real world are necessary to stay ahead of the always-evolving IoT threat environment to provide a high degree of security.

This evaluation provides a baseline to guide the ongoing enhancement of the ensemble classifier’s precision, recall, and robustness in its vital detection role within the Enhanced SCAFFOLD framework. Table 10 summarizes the attack detection accuracy of the ensemble classifier under eight different scenarios in an IoT network. The ensemble classifier combines multiple machine-learning models, including random forests, support vector machines, and recurrent neural networks, to leverage their complementary strengths. The true-positive rate refers to the percentage of attack samples that are correctly identified, while the false-positive rate is the percentage of normal samples incorrectly classified as attacks.

First examining the performance under background HTTP traffic, the ensemble classifier achieved a strong 0.94 true-positive rate in detecting attacks injected into the HTTP flows. However, the 0.021 false-positive rate shows some normal HTTP packets were misclassified as anomalous. HTTP headers and payloads exhibited wide variability, so distinguishing attacks requires deep packet inspection. The models can be refined by training on more HTTP samples to improve their accuracy.

**Table 10.** Ensemble classifier detection accuracy.

| Scenario | True-Positive Rate | False-Positive Rate |
|---|---|---|
| **Background HTTP Traffic** | 0.94 | 0.021 |
| **VoIP Calls** | 0.92 | 0.018 |
| **FTP Sessions** | 0.96 | 0.012 |
| **Custom IoT Traffic** | 0.93 | 0.024 |
| **DDoS Attack** | 0.98 | 0.007 |
| **Reconnaissance** | 0.91 | 0.015 |
| **MITM Attack** | 0.95 | 0.011 |
| **Malware Infection** | 0.97 | 0.008 |

For Voice over IP call traffic, the ensemble reached a 0.92 true-positive rate at finding attacks with a low 0.018 false-positive rate. The real-time nature and audio encoding patterns of VoIP packets differ from HTTP. The ensemble models adapted reasonably well, as evidenced by the high attack detection and minimal false alarms. Further training on VoIP protocols like SIP and RTP could enhance precision.

In the FTP file transfer scenario, the ensemble classifier performed very well, correctly identifying 0.96 of the attacks with just 0.012 false positives. The structured nature of the FTP sessions improved the modeling compared to HTTP exchanges. Attacks manifested as deviations from expected sequences of commands, responses, and data connections. More samples of normal FTP behaviors would enable tighter false-positive control. With customized IoT traffic patterns, the ensemble achieved a reliable 0.93 true-positive attack detection but a slightly higher 0.024 false-positive rate. The unfamiliar application-layer protocols and encodings presented a challenge. Providing more IoT protocol training data would improve accuracy by capturing intrinsic IoT behaviors. The highest 0.98 true-positive rate occurred when detecting the DDoS flooding attacks, which exhibit obvious traffic surges. The dramatic volume spikes are clear indicators of DDoS, allowing precise detection with minimal 0.007 false positives. However, lower-volume DDoS attempts could be harder to distinguish. For reconnaissance probes, the irregular scanning patterns facilitated a 0.91 true-positive detection with a reasonable 0.015 for false alarms. Distinguishing reconnaissance from general access attempts remains challenging, though. Expanding the feature set with additional connection and fingerprinting metrics would assist recognition. Man-in-the-middle attacks were detected with 0.95 accuracy and 0.011 false positives by looking for session anomalies. More samples of failed MITM attacks could better train the classifier on such patterns. Specific crypto attack indicators may also improve detection. Finally, malware infections were identified with a 0.97 true-positive rate and 0.008 false alarms by analyzing abnormal sequences. Signature-based anti-virus scans would complement the ensemble classifier by detecting known malware strains. Integrating host data, like suspicious process behaviors and file system changes, could also enhance malware detection.

### *6.2. Attack Mitigation Delay*

The attack mitigation delay analysis showed that Enhanced SCAFFOLD could respond very quickly to contain obvious flooding-based DDoS attacks by identifying and blocking malicious traffic. However, more subtle attack types like reconnaissance probes, man-in-the-middle attacks, and malware infections posed greater challenges for rapid isolation by the system. The results indicate opportunities to further tune Enhanced SCAFFOLD's detection and mitigation components, especially to handle sophisticated threats that take advantage of protocol weaknesses rather than simply overloading targets. As new attack vectors continue to evolve, maintaining quick yet accurate response capabilities will remain critical for Enhanced SCAFFOLD to fulfill its mission of proactively defending IoT environments against emerging dangers. This evaluation establishes baselines to guide future performance enhancements. Table 11 provides vital performance metrics regarding the effectiveness of Enhanced SCAFFOLD's attack mitigation responses. The table summarizes the minimum, maximum, average, and median delays between detecting an attack and activating countermeasures through the SDN controller installing blocking rules. Four representative attack types were evaluated: distributed denial-of-service (DDoS), reconnaissance probes, man-in-the-middle (MITM), and malware infections [13].

**Table 11.** Attack Mitigation Delay.

| Attack Type | Minimum (ms) | Maximum (ms) | Average (ms) | Median (ms) |
|---|---|---|---|---|
| **DDoS** | 26 | 38 | 31 | 29 |
| **Reconnaissance** | 33 | 62 | 47 | 44 |
| **MITM** | 41 | 71 | 53 | 52 |
| **Malware** | 39 | 67 | 49 | 48 |

For the DDoS attacks, the minimum mitigation delay was an excellent 26 ms, indicating that the attacks were contained almost instantly once detected. The maximum delay rose to 38 ms showing good consistency in the response times. DDoS attacks tend to have very overt traffic signatures, enabling prompt detection. The average mitigation delay was 31 ms and the median 29 ms, demonstrating Enhanced SCAFFOLD's ability to rapidly mitigate DDoS threats. However, lower-bandwidth DDoS attempts with more sporadic traffic patterns could pose greater challenges.

Reconnaissance probes exhibited a higher minimum mitigation delay of 33 ms due to their potential subtlety compared to DDoS flooding. The more intermittent scanning behaviors make isolation more difficult, resulting in a maximum delay of 62 ms. The average and median delays were 47 ms and 44 ms, respectively, for the reconnaissance attacks. Further tuning of detection sensitivity for scanning patterns and reducing controller rule installation latency could improve mitigation performance. For the MITM attacks, the minimum mitigation time rose to 41 ms because of the complexity of distinguishing malformed packets. Isolating the attack flows requires accurately tracking the compromised sessions. The MITM response saw the highest maximum delay of 71 ms due to occasional misclassifications, undermining swift containment. The average and median delays were 53 ms and 52 ms, still representing reasonable mitigation speeds. Further cryptographic analysis and modeling of MITM artifacts [19] would enhance response.

Malware infections showed a 39 ms minimum mitigation time, as abnormal behaviors triggered the security alerts. Persistent malware is harder to isolate completely, though, causing a maximum delay of 67 ms. The ensemble classifier took more time to assess potential malware and determine the correlated flows. The average and median delays were 49 ms and 48 ms, indicating decent but not ideal performances. Integration with anti-virus scanning and host signals, like suspicious process activity, could accelerate malware mitigation.

Overall, Table 11 demonstrates Enhanced SCAFFOLD's ability to respond rapidly across a spectrum of attack types, though optimizable delays exist. The results highlight the exceptional speed against overt DDoS attacks versus increased challenges mitigating clandestine MITM and malware threats. This analysis provides crucial insights for boosting the mitigation capabilities, especially against sophisticated threats. Rapid yet precise attack containment will be critical as IoT environments face growing risks. By establishing baselines under representative conditions, these findings pave the path for systematic improvements in Enhanced SCAFFOLD's mitigation responsiveness and resiliency.

### *6.3. Channel Latency*

To evaluate the effectiveness of the Enhanced SCAFFOLD protocol in maintaining quality of service during threats, we conducted extensive simulations and measured the latency overhead under various network scenarios and attack conditions. Table 12 presents the latency measurements for different network technologies (WiFi, LTE, and wired) and traffic patterns (custom IoT traffic, video conferencing, SSH sessions, and variable traffic loads) under normal conditions and during attacks.

**Table 12.** Normal vs. attack channel latency.

| Network Scenario | Normal Latency (ms) | Attack Latency (ms) |
|---|---|---|
| **WiFi Network** | 32 | 34 |
| **LTE Network** | 48 | 52 |
| **Wired Network** | 14 | 15 |
| **Custom IoT Traffic** | 38 | 42 |
| **Video Conference** | 76 | 79 |
| **SSH Sessions** | 41 | 44 |
| **High Load** | 62 | 68 |
| **Medium Load** | 53 | 58 |
| **Low Load** | 48 | 51 |

The results demonstrate that the Enhanced SCAFFOLD protocol introduces minimal latency overhead during the attack scenarios. For instance, in the WiFi network, the latency increased from 32 ms under normal conditions to 34 ms during the attacks, indicating that the protocol effectively contains the threats while maintaining network performance. Similar trends can be observed for LTE and wired networks, with the attack latencies being only slightly higher than the normal latencies.

When considering different traffic patterns, the Enhanced SCAFFOLD protocol showcased its ability to preserve quality of service. Custom IoT traffic experienced a small increase in latency from 38 ms to 42 ms during the attacks, highlighting the protocol's capability to handle specialized IoT protocols. Video conferencing and SSH sessions also exhibited minor latency increases of 3 ms each, demonstrating the protocol's effectiveness in securing multimedia applications and remote sessions.

To further validate the protocol's performance under varying network loads, we measured the latency for high, medium, and low traffic scenarios. As expected, the high load scenario experienced the largest latency increase from 62 ms to 68 ms during the attacks. However, the protocol managed to maintain availability and contain the latency impact to an acceptable level. The medium- and low-load scenarios showed latency increases of 5 ms and 3 ms, respectively, confirming the protocol's ability to operate efficiently under typical network conditions.

These quantitative results provide concrete evidence of the Enhanced SCAFFOLD protocol's effectiveness in maintaining low latency overhead and ensuring quality of service during threat scenarios. The low latency effect, seen across various network technologies, traffic patterns, and load circumstances, emphasizes the resilience and usefulness of the protocol for practical IoT deployments.

We contrasted Enhanced SCAFFOLD's latency overhead with those of other security frameworks in use to put these findings into context. Studies on DTLS and MQTT-SN have shown 10–20 ms latency increases during attacks. The latency overhead of the Enhanced SCAFFOLD protocol is usually less than 5 ms. This comparison shows our framework's better service quality sustainability.

Several important aspects of the Enhanced SCAFFOLD protocol are responsible for the reduced latency overhead. The computational load on resource-constrained IoT devices is reduced via the use of improved key exchange protocols and lightweight cryptographic primitives. Furthermore, minimal latency and the effective use of network resources are guaranteed by the SDN controller's capacity to selectively reject harmful traffic while permitting valid flows.

The Enhanced SCAFFOLD protocol's ability to maintain minimal latency overhead and guarantee quality of service under threat situations is strongly supported by the quantitative data reported in this section. Our suggested approach's effectiveness and superiority are shown by its negligible latency impact under a variety of network situations and by comparison with other security frameworks. The protocol is ready for practical implementation, as these results demonstrate, and it can improve the security and functionality of IoT systems.

### *6.4. Energy Consumption*

Energy efficiency is a vital metric for IoT systems with constrained devices operating on batteries or energy harvesting. Table 13 provides the energy usage statistics for five representative IoT node types: sensors, smartwatches, smartphones, laptops, and embedded systems. The minimum, maximum, and average power consumption in watts are reported for each node when running the Enhanced SCAFFOLD protocol.

**Table 13.** Node energy consumption.

| Node Type | Minimum (W) | Maximum (W) | Average (W) |
|---|---|---|---|
| **Sensor** | 1.1 | 1.4 | 1.24 |
| **Smartwatch** | 0.8 | 1.2 | 0.92 |
| **Smartphone** | 2.3 | 3.1 | 2.68 |
| **Laptop** | 5.1 | 7.2 | 6.04 |
| **Embedded System** | 2.2 | 3.4 | 2.78 |

The sensors unsurprisingly exhibited the lowest power needs with a 1.1 W minimum, 1.4 W maximum, and 1.24 W average consumption. Many sensors run on coin cell batteries and operate intermittently to achieve multi-year lifetimes. Enhanced SCAFFOLD introduces additional cryptography and communication overhead, which impacts the energy budget. Lightweight cryptographic algorithms optimized for sensors could reduce this overhead. Smartwatches saw a 0.8 W minimum, 1.2 W maximum, and 0.92 W average energy utilization. Wearables have tight power budgets, so the 17% increase from average to peak usage highlights the optimization opportunities. Employing idle sleep states with wake-on-demand could conserve smartwatch energy when Enhanced SCAFFOLD is not actively communicating. Offloading non-time-critical tasks could further improve efficiency.

Smartphones exhibited sizable 2.3 W to 3.1 W energy needs, given their multifunctional capabilities. However, the 2.68 W average is a small proportion of the smartphone battery capacity. There are few constraints to optimizing Enhanced SCAFFOLD for phones, but power-saving WiFi modes could be employed during idle periods to save incremental energy. Laptops unsurprisingly consumed the most power, with a 5.1 W minimum, spiking to a 7.2 W maximum and 6.04 W average. Laptop batteries can readily support these loads. Their higher computing resources could actually be leveraged to offload tasks from more constrained devices to improve the overall ecosystem's efficiency.

Finally, embedded systems showed an expected 2.2 W to 3.4 W range and 2.78 W average power requirement. Optimization depends on the embedded use case - for battery-powered field systems, duty cycling and wake-on-demand are viable. On continuously powered devices, computation offloading or caching could save energy.

### *6.5. Limitations and Challenges*

The Enhanced SCAFFOLD framework, while demonstrating significant potential for securing IoT environments, has certain limitations and faces challenges in its implementation. Scalability is a critical concern when deploying the framework in large-scale IoT networks. As the number of connected devices and the volume of network traffic increase, the SDN controller may become a performance bottleneck. Further research and development are necessary to ensure that the framework can withstand the scaling needs of large-scale IoT installations.

Compatibility with the many IoT devices and protocols is another issue. IoT ecosystems include devices with diverse operating systems, functions, and communication protocols. The Enhanced SCAFFOLD architecture must be tested and customized to fit easily into this diverse environment. Even though the framework provides high security, vulnerabilities must be detected and fixed immediately. Security testing and analysis are essential to uncover attacker-exploitable flaws.

One ongoing research area is balancing security, privacy, and performance in IoT systems. Integrating privacy-preserving approaches into the Enhanced SCAFFOLD architecture requires further study to protect sensitive user data without sacrificing performance.

The effective implementation of the security architecture also depends on usability and user approval. Facilitating the appropriate setup, maintenance, and adoption of the Enhanced SCAFFOLD framework requires the design of user-friendly interfaces, the provision of clear instructions and documentation, and the completion of user studies.

A key limitation of the current performance analysis is the lack of simulation for the machine learning components of Enhanced SCAFFOLD. Simulating the ensemble classifier and other ML models under realistic IoT scenarios is crucial to comprehensively evaluating their effectiveness and efficiency. This is a critical area for future work to ensure the practical viability of machine learning-based security features.

## 7. Conclusions and Future Work

This paper presented the Enhanced SCAFFOLD protocol, a comprehensive security framework tailored for IoT environments. Enhanced SCAFFOLD establishes resilient encrypted channels between IoT nodes using session keys derived from preliminary lightweight key exchanges. The integration of SDN enables continuous monitoring of network flows and traffic patterns. Ensemble machine-learning classifiers reliably detect anomalies that indicate potential attacks. The SDN control plane allows precision mitigation by surgically blocking only suspicious flows while maintaining availability. Periodic reauthentication and rekeying provide freshness against compromise over time.

Detailed mathematical modeling concisely captures Enhanced SCAFFOLD's mechanisms and logic flows. Precise protocol syntax and structured pseudocode algorithms define the sequence of operations and data exchanges. Extensive simulations quantified Enhanced SCAFFOLD's ability to detect and rapidly mitigate various attack types, including distributed denial of service, reconnaissance probes, man-in-the-middle attacks, and malware infections. The results validated the low latency overhead of the protocol for maintaining quality of service during threats. The energy consumption analysis highlighted efficiency tradeoffs for more constrained IoT devices. This multidimensional defense-in-depth approach integrates essential security capabilities like confidentiality, integrity, authorization, intelligent threat detection, and adaptive response to protect modern IoT systems.

While the Enhanced SCAFFOLD framework presents a promising approach for securing IoT environments, there are several limitations and challenges that need to be addressed for its practical implementation. Future research should focus on enhancing the scalability of the framework to handle the demands of large-scale IoT deployments. Investigating distributed SDN architectures, efficient load balancing mechanisms, and techniques such as hierarchical control plane design and edge computing could help distribute the processing load and improve its responsiveness.

Future development must also focus on ensuring interoperability with the wide variety of IoT devices and protocols. Protocol adapters, standardized interfaces, and collaborations with IoT device manufacturers and standards organizations enable widespread adoption and smooth integration of the Enhanced SCAFFOLD architecture. Continuous security testing and analysis are needed to detect and remedy framework vulnerabilities. Code audits, penetration testing, and framework component verification should be a part of future studies. Participating in bug bounty programs and security research may help uncover and fix problems quickly.

Further study should include machine learning simulation into the Enhanced SCAFFOLD performance assessment. The ensemble classifier and other ML components must be tested in a number of IoT settings to ensure its real-world applicability and performance. This is essential for building and using the Enhanced SCAFFOLD structure.

Privacy-preserving solutions for the Enhanced SCAFFOLD architecture are another interesting research subject. Homomorphic encryption, secure multi-party computing, and differential privacy protect data aggregation and analysis.

To achieve acceptance, the Enhanced SCAFFOLD framework must be more appealing and user-friendly. Provide clear instructions, setup tools, and interfaces. Usability concerns may be identified and fixed using user surveys and deployment feedback.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Research data will be available on individual requests to the corresponding author considering collaboration possibilities with the researcher or research team and with restrictions that the data will be used only for further research in the related literature progress.

**Acknowledgments:** The author would like to thank the Deanship of Scientific Research at Arab American University for supporting this research.

**Conflicts of Interest:** The author declares no conflict of interest.

## References

1. Jara, A.J.; Zamora, M.A.; Skarmeta, A.F. An Internet of Things—Based personal device for diabetes therapy management in ambient assisted living (AAL). *Pers. Ubiquitous Comput.* **2011**, *15*, 431–440.
2. Da Xu, L.; He, W.; Li, S. Internet of Things in industries: A survey. *IEEE Trans. Ind. Inform.* **2014**, *10*, 2233–2243.
3. Miorandi, D.; Sicari, S.; De Pellegrini, F.; Chlamtac, I. Internet of things: Vision, applications and research challenges. *Ad Hoc Netw.* **2012**, *10*, 1497–1516.
4. Deering, S.; Hinden, R. *Internet Protocol, Version 6 (IPv6) Specification*; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2017.
5. Kent, S. *IP Authentication Header*; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2005.
6. Hui, J.; Thubert, P. *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2011.
7. Brandt, A.; Buron, J.; Porcu, G. *Home Automation Routing Requirements in Low-Power and Lossy Networks*; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2017.
8. Raza, S.; Shafagh, H.; Hewage, K.; Hummen, R.; Voigt, T. Lithe: Lightweight secure CoAP for the internet of things. *IEEE Sens. J.* **2013**, *13*, 3711–3720.
9. Grammatikis PI, R.; Sarigiannidis, P.G.; Moscholios, I.D. Securing the Internet of Things: Challenges, threats and solutions. *Internet Things* **2019**, *5*, 41–70. https://doi.org/10.1016/j.iot.2018.11.003.
10. Yang, Y.; Wu, L.; Yin, G.; Li, L.; Zhao, H. A survey on security and privacy issues in internet-of-things. *IEEE Internet Things J.* **2017**, *4*, 1250–1258.
11. Borgohain, T.; Kumar, U.; Sanyal, S. Survey of security and privacy issues of internet of things. *arXiv* **2015**, arXiv:1501.02211.
12. Rahman, N.H.A.; Choo, K.-K.R. A survey of information security incident handling in the cloud. *Comput. Secur.* **2015**, *49*, 45–69.
13. Elejla, O.E.; Belaton, B.; Anbar, M.; Alnajjar, A. Intrusion detection systems of ICMPv6-based DDoS attacks. *Neural Comput. Appl.* **2018**, *30*, 45–56.
14. Zhang, L.; Han, Y.; Wang, Y.; Quan, R. Petri Net Model of MITM Attack Based on NDP Protocol. In Proceedings of the 2022 International Conference on Networking and Network Applications (NaNA), Urumqi, China, 3–5 December 2022.
15. Moghadam, A.Q. Entropy-based SYN flooding attack detection at leaf router of the network. *J. Intell. Fuzzy Syst.* **2018**, *35*, 591–601.

16. Maleh, Y.; Fatani IF, E.; Gholami, K.E. A Systematic Review on Software Defined Networks Security: Threats and Mitigations. In Proceedings of the International Conference on Information, Communication & Cybersecurity, Khourigba, Morocco, 10–11 November 2021.
17. Zarif, N.S.; Moghadam, A.Q.; Imani, M. Hybrid Technique for Spectrum Sharing in Cognitive Radio Networks for the Internet of Things. *Int. J. Comput. Appl.* **2018**, *179*, 14–18.
18. Anbar, M.; Abdullah, R.; Saad, R.M.; Alomari, E.; Alsaleem, S. Review of Security Vulnerabilities in the IPv6 Neighbor Discovery Protocol. In Proceedings of the ICISA 2016: 7th International Conference on Information Science and Applications 2016, Ho Chi Minh, Vietnam, 15–18 February 2016.
19. Wlazlo, P.; Sahu, A.; Mao, Z.; Huang, H.; Goulart, A.; Davis, K.; Zonouz, S. Man-in-the-middle attacks and defence in a power system cyber-physical testbed. *IET Cyber-Phys. Syst. Theory Appl.* **2021**, *6*, 164–177.
20. Zhang, T.; Wang, Z. Research on IPv6 neighbor discovery protocol (NDP) security. In Proceedings of the 2016 2nd IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, 14–17 October 2016.
21. Arjuman, N.C.; Manickam, S.; Karuppayah, S. An Improved Secure Router Discovery Mechanism to Prevent Fake RA Attack in Link Local IPv6 Network. In Proceedings of the Third International Conference on Advances in Cyber Security, ACeS 2021, Penang, Malaysia, 24–25 August 2021.
22. Al-Ani, A.K.; Anbar, M.; Al-Ani, A.; Ibrahim, D.R. Match-Prevention Technique Against Denial-of-Service Attack on Address Resolution and Duplicate Address Detection Processes in IPv6 Link-local Network. *IEEE Access* **2020**, *8*, 27122–27138.
23. Moghadam, A.Q.; Imani, M. A new method of IPv6 addressing based on EPC-mapping in the Internet of Things. In Proceedings of the 2018 4th International Conference on Web Research (ICWR), Tehran, Iran, 25–26 April 2018.
24. Premarathne, U.S.; Abuadbba, A.; Alabdulatif, A.; Khalil, I.; Tari, Z.; Buyya, R.; Deen, M.J. Hybrid cryptographic access control for cloud-based IoT applications. *IEEE Cloud Comput.* **2020**, *7*, 48–59.
25. Zhou, D.; Wang, L. Research on Direct Lift Carrier-Based Unmanned Aerial Vehicle Landing Control Based on Performance Index Intelligent Optimization/Dynamic Optimal Allocation. *Drones* **2023**, *7*, 431.
26. Sai, K.S.; Bhat, R.; Hegde, M.; Andrew, J. A Lightweight Authentication Framework for Fault-tolerant Distributed WSN. *IEEE Access* **2023**, *11*, 83364–83376.
27. Gao, Y.; Li, H.; Xiong, G.; Song, H. AIoT-informed digital twin communication for bridge maintenance. *Autom. Constr.* **2023**, *150*, 104835.
28. Song, H.M.; Lee, K.; Lee, D.H. Early detection of malware attacks applying stochastic petri net in internet of things. In Proceedings of the 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS), Seoul, Republic of Korea, 27–29 September 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 294–297.
29. Shafighfard, T.; Kazemi, F.; Bagherzadeh, F.; Mieloszyk, M.; Yoo, D.Y. Chained machine learning model for predicting load capacity and ductility of steel fiber–reinforced concrete beams. *Comput.-Aided Civ. Infrastruct. Eng.* 2024, *Epub ahead of printing*.
30. Bagherzadeh, F.; Shafighfard, T.; Khan RM, A.; Szczuko, P.; Mieloszyk, M. Prediction of maximum tensile stress in plain-weave composite laminates with interacting holes via stacked machine learning algorithms: A comparative study. *Mech. Syst. Signal Process.* **2023**, *195*, 110315.
31. Asgarkhani, N.; Kazemi, F.; Jakubczyk-Gałczyńska, A.; Mohebi, B.; Jankowski, R. Seismic response and performance prediction of steel buckling-restrained braced frames using machine-learning methods. *Eng. Appl. Artif. Intell.* **2024**, *128*, 107388.
32. Wang, C.; Atkison, T.; Park, H. Dynamic adaptive vehicle re-routing strategy for traffic congestion mitigation of grid network. *Int. J. Transp. Sci. Technol.* 2023, *in press*.