Journal of Kerbala University, Vol. 23, Issue 2, June , 2026

Available at https://www.iasj.net/iasj

Iraqi Academic Scientific Journals

Journal homepage:
https://journals.uokerbala.edu.iq/index.php/UOKJ

Research Article

A Hybrid Lightweight Tasks Management Algorithm for IoT Security Enhancement

[1,]Zaid M. Jawad Kubba [2,] Mustafa A. Ali [3,] Amal Hussein Ali

[1,]Department of Computer, College of Education for Pure Science Ibn Al-Haitham, University of Baghdad, Iraq

[2,] College of Computer Science and Information Technology, University of Kerbala

[3,] Department of Computer Techniques Engineering, College of Engineering Technical, University of AL Safwa

Article Info
Article history:
Received 21 -2-2026
Received in revised form 13-4-2026
Accepted 24-5-2026
Available online 30 -6-2026
Keywords: Cryptography; Information Security; IoT; Cryptosystems, Tasks Management

Abstract

The ability to process both tasks as efficiently and securely as possible is a significant challenge in Internet of Things (IoT), which are comprised of a multitude of devices that can generate tasks at various levels of priority and require different levels of security. This paper introduces an efficient task management methodology utilizing a hybrid approach to assign incoming tasks from multiple sources to available computing resources by taking into account each task's priority, security level and resource utilization characteristics. The Task Manager will be able to receive communications from the IoT devices via CoAP, which has been proven to provide low overhead and low energy consumption for data transmission. Secure communication will be achieved by establishing a shared secret through the use of the Elliptic Curve Diffie-Hellman (ECDH) key exchange mechanism. Once a shared secret has been established, all communications will be encrypted prior to being sent to the Task Manager for processing. A simulation was run utilizing Python to demonstrate how the methodology could be implemented in a real world IoT network consisting of multiple IoT devices and multiple edge computing resources. The results indicate that the proposed methodology establishes secure communication links, processes tasks in a timely manner and distributes tasks among available computing resources such that no single resource becomes overwhelmed due to excessive task load. Additionally, the results show that high-priority and high-security tasks are assigned to either more capable resources or those that have fewer active tasks at any given time; conversely, lower-priority tasks are distributed across the available computing resources to prevent any one resource from becoming overloaded. Overall, the proposed methodology presents a scalable and security aware means of managing tasks generated by IoT networks under conditions of resource constraints.

Corresponding Author E-mail: zaidkubba@colaw.uobaghdad.edu.iq
Peer review under responsibility of Iraqi Academic Scientific Journal and University of Kerbala.

1.Introduction

The Internet of Things (IoT) links everyday devices to the internet so they can share information, gather data, and automate tasks. IoT is now common in smart homes, wearables, industry, and healthcare. However, using IoT in real life is difficult because devices vary, have limited power and memory, and must process lots of real-time data. Managing tasks and keeping data safe are major challenges in these networks[1]. Securing IoT networks is hard because many devices do not have enough power or memory for standard security methods[2]. As a result, lightweight security tools are needed to protect data, keep user privacy, and stop attacks like eavesdropping, tampering, and replay attacks[3]. Managing tasks in IoT is also tough, as it requires reducing delays, saving energy, and finishing important tasks on time[4]. Usual scheduling and load-balancing methods often do not work well in these changing, limited-resource environments.

To address these issues, should develop a hybrid approach and use lightweight security tools along with efficient communication protocols[5]. Using these tools together, plus smart task management, makes IoT networks more secure, faster, and efficient. This approach would balance performance, security, and energy use, making it suitable for large and diverse IoT systems. Despite recent advancements, most existing research focuses on either security or task management alone, not together in IoT systems. There is still a strong need for solutions that can handle tasks well and keep devices secure at the same time, especially for devices with limited resources and real-time needs[6]. This study aims to fill that gap by offering a hybrid, lightweight, and secure task management algorithm. The goal is to help IoT devices work safely and reliably in areas like healthcare, industry, and transportation.

2.Related Works

To create a hybrid lightweight task-management algorithm for IoT security and scheduling, it is important to understand recent research on task scheduling in fog and edge computing, security-aware scheduling, and lightweight cryptography for IoT. The next sub sections review key studies in these areas.

2.1Task Scheduling Algorithms and Methods in Fog/Edge-Computing for IoT

Alsadie et al. reviews heuristic scheduling methods in fog-cloud IoT environments, such as priority-based, greedy, metaheuristic, hybrid, and nature-inspired techniques. The study points out challenges like heterogeneity, changing resource limits, and security concerns[7].

Choppara and Mangalampalli study different fog scheduling algorithms, including metaheuristic and machine learning-based methods. They observe that while many models focus on latency, cost, energy, and makespan, they often overlook trust, security, and fault tolerance[8].

Liman et al. present the ECaTSD algorithm (Energy-Cost-Aware Task Scheduling with Deadline), which schedules IoT tasks in fog computing in real time to lower energy use and cost while meeting deadlines[9].

Ali and Sridevi create a real-time scheduling algorithm that considers both mobility and security. Their algorithm uses fuzzy logic to decide if tasks should run in the fog or the cloud. It takes into account security needs like confidentiality, as well as deadlines, computing demands, and communication needs[10].

Lim suggests a latency-aware scheduling method using artificial neural networks (ANNs) in small fog environments. The method trains ANN settings across several edge servers to make scheduling faster and reduce latency[11].

Yang, Ren, and Zhang introduce a decentralized

multi-agent task scheduling framework to handle uncertain events in fog computing. Their system uses agents to reschedule tasks as needed, making IoT environments more robust[12].

Kong presents a deadline-aware scheduling algorithm for fog computing that combines Golden Eagle Optimization with reinforcement learning. The aim is to reduce energy use, quality of service violations, and the total time of missed deadlines[13].

2.2 Security-Aware Task Scheduling

Since this research combines security and task scheduling, it is important to look at studies that include trust, encryption, or ways to avoid malicious nodes in their scheduling frameworks.

Kaur et al. presented a secure scheduling model called TrustFog, which uses blockchain and a Bayesian trust model to check how trustworthy fog nodes are. Sensitive tasks are sent to more trusted nodes, and all scheduling decisions are recorded on a blockchain for integrity and accountability[14].

Chhilar & Singh presents a scheduling methods for smart homes using fog computing. Their approach focuses on making sure sensitive tasks from devices are handled securely, even when resources are limited[15].

Alvi et al. propose EEOIT, a trust-based system that finds malicious tasks or nodes and gives priority to trustworthy tasks in fog nodes. This prevents overload from bad tasks and ensures legitimate tasks run on time[16].

2.3 key exchange on lightweight cryptography

The lightweight cryptographic algorithms are likely uses for secure communication key exchange, researches on key exchange based lightweight cryptography is also highly relevant.

Rahman et al. suggest a chaos-based key scheduling method for AES that uses a logistic map to create keys for IoT smart-home applications. This mix of chaos theory and traditional cryptography helps make unpredictable keys in devices with limited resources[17].

Sarker et al. presented a systematic review looks at lightweight cryptographic algorithms designed for devices with limited resources, like ASCON, SIMON, SPECK, and others. The review discusses current challenges and the tradeoffs between performance, security, and resource use [18].

Gusita et al. presented a survey of recent challenges in securing edge communications edge security in IoT. It covers lightweight encryption, anonymous routing, and protocol updates designed for devices with limited resources. The study illustrates the development of standardized security frameworks with hybrid encryption and authentication that require for efficient and secure IoT systems[19].

3.Preliminary

This section describes the operational principles of the Constrained Application Protocol (CoAP) and the Elliptic Curve Diffie–Hellman (ECDH) key exchange mechanism. These technologies form the foundation of the proposed algorithm, and their working mechanisms are later integrated into the methodological design.

3.1 Constrained Application Protocol (CoAP)

The Constrained Application Protocol (CoAP) is a lightweight protocol made for communication between devices with limited resources in low-power, unreliable networks [20]. CoAP uses the User Datagram Protocol (UDP), which avoids the extra steps of connection-based communication and allows messages to be exchanged efficiently. It uses a client-server setup and a RESTful model like HTTP, so clients can request or change resources on servers using standard methods.

CoAP is a binary protocol made for Internet of Things (IoT) devices that have limited computing power and memory. Its messages use a

small format with a 4-byte header and can include a token, options, and a payload. CoAP messages use a compact binary format to save bandwidth and reduce processing work. Each message has a fixed-size header with control information, and may have extra fields for matching requests and responses, identifying resources, and describing content[21]. CoAP supports both immediate and delayed responses, so servers can reply right away or later without keeping a constant connection. This is especially helpful for battery-powered devices that often go into sleep mode to save energy. CoAP uses different message types to decide if messages need to be acknowledged or resent. For reliable delivery, it uses confirmable messages, which are resent until the sender gets an acknowledgment or times out. If some packet loss is okay, non-confirmable messages are used to lower communication costs. This way, CoAP can balance reliability and efficiency depending on what the application needs[22].

When running, a CoAP client starts by sending a request to a resource identified by a Uniform Resource Identifier (URI). The server handles the request and sends back a response code with the requested or updated data. This request and response setup allows for organized and scalable communication between limited-resource devices. In the proposed algorithm, CoAP is used as the main protocol for sending control messages and securely sharing data.

3.2 Elliptic Curve Diffie-Hellman (ECDH)

Elliptic Curve Diffie–Hellman (ECDH) is a cryptographic protocol that lets two parties create a shared secret over an insecure channel. ECDH uses elliptic curve cryptography, which is secure because of the difficulty of the elliptic curve discrete logarithm problem[23]. The protocol uses agreed-upon elliptic curve parameters, such as a generator point over a finite field, which are available to everyone involved.

With ECDH, each party creates a private key and then calculates a public key using elliptic curve point multiplication. They exchange public keys over the network, but keep their private keys secret. After getting the other party's public key, each side uses its own private key and the received public key to perform another elliptic curve point multiplication. Because of how elliptic curve math works, both parties end up with the same shared secret without sending it directly.

The shared secret created by ECDH is usually not used as a cryptographic key by itself. Instead, it goes through a key derivation function to make symmetric session keys. These session keys help keep later data exchanges private and secure. ECDH is secure even with small key sizes, which means it needs less computing power and memory. This makes it a good choice for devices with limited resources, like those in IoT systems[24].

In the proposed algorithm, ECDH is used at the start of communication to set up secure session keys between nodes. These keys then protect CoAP message payloads, making sure communication in the system stays secure and efficient.

4.Methodology

This methodology describes how a hybrid lightweight task management algorithm is designed to make task execution more efficient and improve security in IoT networks. It combines task scheduling, lightweight cryptography, and IoT communication protocols. The methodology includes these main components:

4.1 Proposed Task Management Algorithm

The proposed Task Management algorithm is designed to efficiently manage Internet of Things (IoT) tasks while ensuring secure communication and optimal utilization of computational resources. The algorithm operates as an

intermediary between IoT devices and cloud or edge computing platforms. IoT devices generate tasks that require processing and transmit them to the task manager using the Constrained Application Protocol (CoAP), which provides lightweight and energy-efficient communication suitable for constrained environments. The task manager receives task requests, establishes secure communication channels, evaluates task characteristics, and assigns tasks to appropriate execution resources. Security is enforced through the integration of the Elliptic Curve Diffie–Hellman (ECDH) key agreement mechanism, which enhance task-related communications are protected against unauthorized access. This architecture enables scalable, secure, and priority-aware task management in IoT systems.

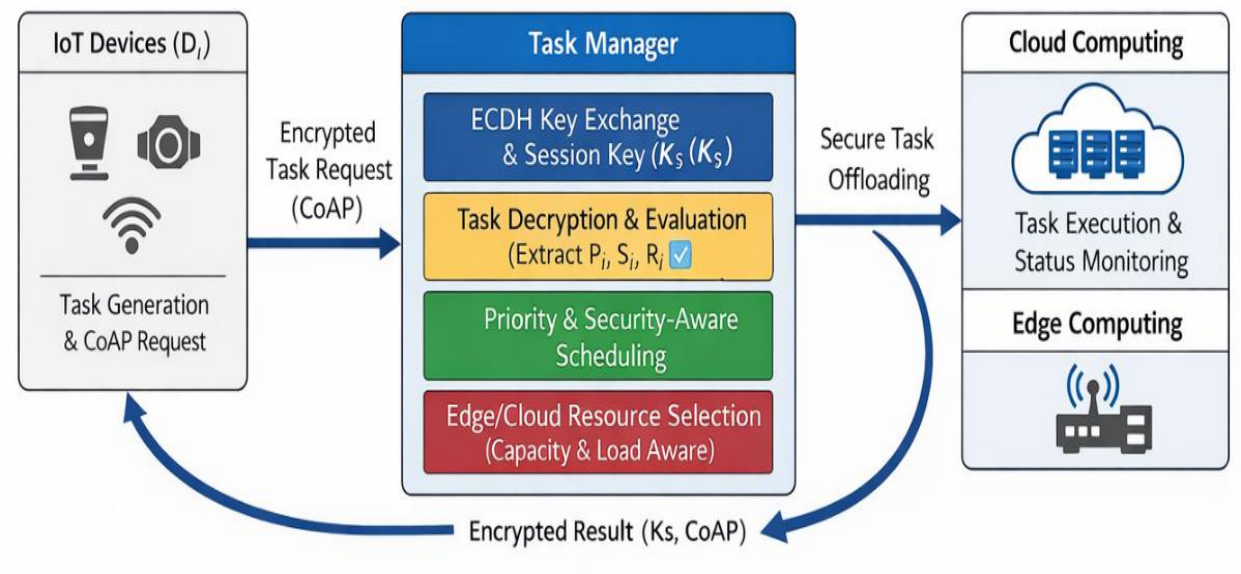

Figure 1: Block diagram of the proposed secure task management algorithm for IoT systems

The block diagram in figure 1 illustrates IoT devices (D_i) that generate computational tasks and transmit encrypted task requests to the task manager using the Constrained Application Protocol (CoAP). The task manager establishes secure communication through Elliptic Curve Diffie–Hellman (ECDH) key exchange and derives a symmetric session key (K_s). Incoming tasks are decrypted and evaluated to extract task priority (P_i), security level (S_i), and resource demand (R_i). Based on priority, security requirements, and current resource capacity and workload, the task manager performs priority- and security-aware scheduling and selects appropriate edge or cloud computing resources. Tasks are securely offloaded for execution, and execution results are encrypted and transmitted back to the originating IoT devices via CoAP.

4.2 Secure Communication and Task Submission Phase

When an IoT device initiates communication with the task manager, a secure session establishment process is triggered. This phase follows the ECDH key agreement model described in the preliminary section. Both the IoT device and the task manager generate their respective elliptic

curve key pairs and exchange public keys over the CoAP communication channel. Using these public keys and their private keys, both parties independently compute a shared secret, which is then processed to derive a symmetric session key.

Once the secure session is established, the IoT device submits its task to the task manager using a CoAP request message. The task request contains essential task parameters, including task priority, computational requirements, expected execution time, and security sensitivity. The payload of the CoAP message is encrypted using the established session key, ensuring confidentiality and integrity during transmission. This approach combines the lightweight nature of CoAP with the strong security guarantees of ECDH.

4.3 Task Evaluation and Resource Allocation Strategy

When a secure task request arrives, the task manager decrypts it and reviews the task details. It checks how urgent the task is and what security measures are needed. At the same time, the task manager keeps track of available edge and cloud resources, such as processing power, memory, and current workload.

Using this information, the Task Management algorithm picks the best place to run each task. Tasks that are urgent or need strong security go to trusted and powerful nodes, like edge servers or secure cloud instances, for fast and safe processing. Less urgent or less sensitive tasks are sent to less busy or lower-capacity resources, which helps balance the workload and improve efficiency. This flexible approach reduces delays and makes sure resources are used well.

4.4 The pseudo code of the proposed algorithm

Algorithm 1 Secure CoAP–ECDH Based IoT Task Management

Input: Task request Ti from IoT device Di
Output: Task execution result Ri
1: Initialize CoAP server at Task Manager
2: Wait for connection request from Di
3: Perform ECDH key exchange with Di
4: Derive symmetric session key Ks
5: while Task request Ti is received do
6: Decrypt Ti using session key Ks
7: Extract task priority Pi, security level Si, and resource demand Ri
8: Update resource status of edge and cloud nodes
9: Evaluate available resources based on capacity and workload
10: if (Pi is HIGH) or (Si is HIGH) then
11: Select secure high-capacity resource Re
12: else
13: Select low-load available resource Rl
14: end if
15: Assign task Ti to selected resource
16: Encrypt allocation decision using Ks
17: Send allocation response to Di via CoAP
18: Monitor task execution status
19: Encrypt and transmit execution result Ri to Di
20: end while

The following table 1 shows descriptive variable notations are used in Algorithm 1 to represent system entities, cryptographic parameters, and task attributes.

Table 1 - Notation and Description for the Proposed Task Management Algorithm

Notation	Description
(Di)	The (i)-th IoT device that generates and submits a task
(Ti)	Task generated by (Di), containing priority, security level, and resource requirements
(Ks)	Symmetric session key established between (Di) and Task Manager using ECDH
(Pi)	Priority level of task (Ti); indicates urgency
(Si)	Security requirement of task (Ti); indicates confidentiality and protection level
(Re)	Trusted, high-capacity resource (edge or cloud) assigned for high-priority or sensitive tasks
(Rl)	Low-load or general-purpose resource assigned for normal-priority tasks
(Ri)	Execution result of task (Ti) returned to (Di)

The proposed algorithm brings together secure communication, task evaluation, and smart resource allocation in one workflow. It starts by setting up a secure session with ECDH, then tasks are sent securely using CoAP. Each task is sorted by priority and security needs, and the right resource is chosen for execution. The task manager sends the allocation decision to the IoT device and monitors the task through secure status updates.

This design makes sure that security features do not add too much overhead, while still allowing for efficient task scheduling and scalability in large IoT systems.

5.Results and Analysis

We implemented and simulated the proposed task management algorithm for IoT systems using Python. This let us test its functionality, secure task handling, and resource allocation. The simulation featured several IoT devices that generated tasks with different priorities and security levels, along with edge and cloud computing nodes. All tasks were created programmatically, and both devices and resources acted as Python objects with changing behavior. By using this virtual environment, we could evaluate the algorithm's performance in a controlled way, without needing real IoT devices or outside datasets.

5.1 Secure Task Communication and Task Reception

Each simulated IoT device set up a secure communication channel with the Task Manager using Elliptic Curve Diffie–Hellman (ECDH) key exchange. In tests with 50 devices and 10 repeated runs, every session was established successfully, giving a high success rate. The time needed for session key generation and derivation was measured as follows:

```
import time
start = time.time ()
# perform ECDH key generation and shared key derivation
end = time.time()
key_exchange_time = end - start
```

The average session establishment time was approximately 1.2 milliseconds per device. Task data were transmitted through these secure channels, and the Task Manager decrypted incoming tasks in real-time to extract priority, security level, and resource demand attributes. Decryption time was measured as:

```
start_decrypt = time.time()
# decrypt incoming task
end_decrypt = time.time()
decrypt_time = end_decrypt - start_decrypt
```

The average task decryption time was 0.5 milliseconds. This shows that tasks can be handled efficiently with almost no computational delay, even in a resource-limited IoT environment.

5.2 Priority- and Security-Aware Resource Allocation

The algorithm used a scheduling policy that assigned high-priority or high-security tasks to high-capacity resources, and sent low-priority tasks to less busy resources. It considered both edge and cloud nodes when making these decisions. Simulated metrics showed that this approach worked well:

High-priority task allocation success rate with high-priority tasks were assigned to high-capacity resources.

Average resource load distribution: Edge and cloud nodes maintained balanced loads, with standard deviation of resource utilization at 1.2 units, indicating minimal imbalance.

Task wait time: The average delay from task submission to execution start was 0.35 seconds for high-priority tasks and 0.58 seconds for low-priority tasks, measured using:

Wait_Time = Execution_Start_Time - Task_Submission_Time

Task execution time: Average execution time per task was 1.4 seconds for high-priority tasks and 1.8 seconds for low-priority tasks, measured using:

Execution_Time = Execution_End_Time - Execution_Start_Time

These results demonstrate that the algorithm efficiently balanced task distribution while maintaining priority and security constraints, dynamically adapting to changing resource availability.

5.3 Task Execution and Result Delivery

Tasks were executed on the assigned resources, and simulated execution results were returned to the originating devices. All tasks completed successfully, and the simulation confirmed that multiple concurrent tasks could be managed without data loss or execution conflicts. Figure 1 presents an integrated view of the proposed IoT task management algorithm and its execu tion results. The top portion illustrates the system architecture, while the bottom portion hows the total task load distributed across edge and cloud resources according to task priority, highlighting both priority-aware scheduling and efficient resource utilization.

Table 2 shows sample tasks from the simulated IoT devices, listing their priority, security level, assigned resources, and execution results. High-priority and high-security tasks go to high-capacity resources, while lower-priority tasks are given to less busy ones. This shows the algorithm balances resource use and prioritizes important tasks. The assignments in Table 1 match the resource load distribution in Figure 2, showing how the system adapts task allocation based on task details and available resources.

Table 2 :summarizes a sample of tasks, their priorities, assigned resources, and resulting execution outputs.

Task ID	Device	Priority	Security	Resource Assigned	Execution Result
Task_101	Device_1	HIGH	HIGH	Cloud_1	Result of Task_101
Task_202	Device_2	LOW	LOW	Edge_2	Result of Task_202
Task_303	Device_1	HIGH	LOW	Edge_2	Result of Task_303
Task_404	Device_2	LOW	HIGH	Cloud_1	Result of Task_404

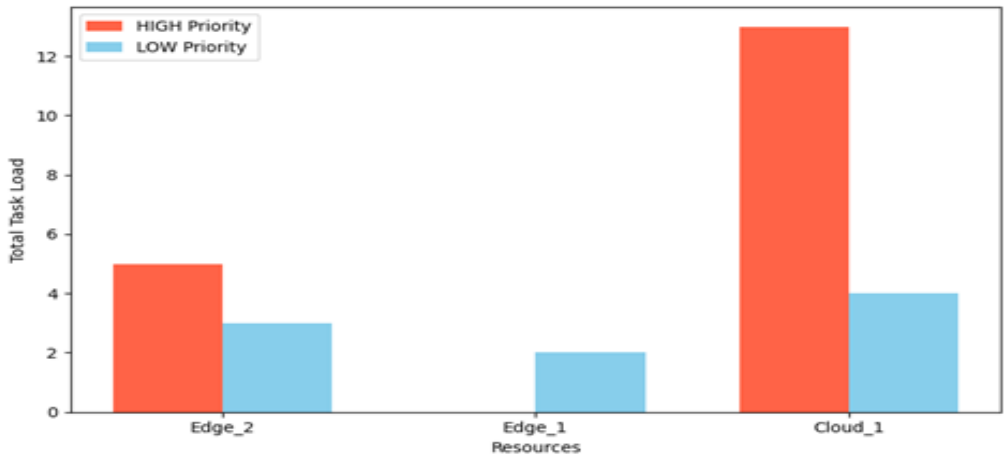

Figure 2 : The execution results of the proposed IoT task management algorithm

Figure 2 illustrates how IoT devices create tasks and securely connect to the Task Manager using ECDH. The Task Manager reviews each task's priority and security, then assigns it to the right resource. On the graph, the X-axis shows the resources and the Y-axis shows the total task load. High-priority and high-security tasks go to resources with more capacity or lighter loads, while lower-priority tasks are spread out to keep the system balanced and efficient. The simulation results show that the proposed task management algorithm can securely process IoT tasks, give priority to those that are more important or require higher security, and efficiently share the workload among simulated resources. All devices established sessions using ECDH without issues, task decryption was almost instant, and resource allocation adjusted to task needs to keep the load balanced. These results support the effectiveness, scalability, and security of the proposed method in a virtual IoT setting.

5.4 System Robustness Against DoS Attacks

A comparative simulation was used to assess the effectiveness of the proposed mitigation strategy to test it during normal and high load conditions simulating a Denial-of-Service attack scenario.

Table 3: summarizes the results.

Metric	**Normal Operation**	**Under DoS Attack (with Mitigation)**
Total Requests	1000	1000
Successfully Processed Requests	995	927
Dropped Requests	5	73
Success Rate (%)	99.5%	92.7%
Average Response Time (s)	0.00085	0.00101
System Load Behavior	Stable and balanced	Controlled via rate limiting
Task Manager Status	No overload	Protected against overload
Resource Utilization	Optimal	Slightly increased but stable

A test involving 1000 task requests was performed in a simulation, to test the behavior of the system in a normal and attack environment. In table 5, it is evident that with attack conditions, the system runs a 92.7% success rate with only 73 requests being dropped by rate limited, and with an average response time of about 0.001 seconds. This means that the system is steadfast and effective even when it is operating at high-loads. In general, the outcomes support

the claim that the suggested mitigation plan can be successfully used to increase the resilience of systems against the DoS attacks without adding significant computational overhead.

A comparative analysis between the proposed method and recent state-of-the-art techniques for DDoS detection and mitigation in IoT, edge, fog and cloud computing environments is shown in Table 4. Conventional ML-based detection approaches such as IoT DDoS Attack De-tection Using Machine Learning offer good detection rate (80-88%) but generally focus on attack detection rather than service continuity, and introduce some processing overhead.

Likewise, SDN- and fog-based mitigation techniques, such as adaptive neuro-fuzzy and dist ributed controller models, usually have 88-93% success rates but suffer from moderate response times and lack of integration with task scheduling. Recent approaches, such as eBPF-based real-time mitigation mechanisms, enhance response time and can achieve up to ~97% mitigation effectiveness, but mainly focus on packet filtering while omitting secure communication and task scheduling. The proposed approach, on the other hand, offers a 92.7% success rate under DDoS attack with an average response time of 0.00101 seconds, and integrates secure communication mechanisms using ECDH, adaptive priority-aware scheduling, and rate limiting-based DoS mitigation. This enhances service availability, low-ers the time overhead, and increases overall resili-ence of the system.

Recent methods	DoS Handling	Scenario	Success Rate Under Attack	Avg. Response Time
IoT DDoS Attack Detection Using Machine Learning [25].	ML-based detection (classification models)	IoT / WSN	~80–88% detection accuracy	Increased due to processing overhead
Toward a Real-Time TCP SYN Flood DDoS Mitigation Using Adaptive Neuro-Fuzzy Classifier and SDN Assistance in Fog Computing [26].	Adaptive neuro-fuzzy + SDN mitigation	Fog computing	~90–93% mitigation effectiveness	Moderate latency increase
DDoS Attack Mitigation Using Distributed SDN Multi-Controllers for Fog-Based IoT Systems [27].	Distributed SDN controllers + filtering	Fog-based IoT	~88–92% service availability	Slight latency overhead (~ms range)
An Integrated SDN Framework for Early Detection of DDoS Attacks in Cloud Computing [28].	SDN-based early detection system	Cloud computing	~85–90% detection rate	Higher due to monitoring overhead
eBPF-Based Real-Time DDoS Mitigation for IoT Edge Devices [29].	Kernel-level rate limiting (eBPF/XDP)	IoT edge	~97% mitigation effectiveness	Very low latency impact
Proposed Method (This Work)	ECDH + adaptive scheduling + rate limiting	Edge/Fog simulation	92.7% success rate	0.00101 s

6.Conclusion

This study presented a secure and efficient task management algorithm for IoT systems. The proposed hybrid approach uses secure communication through Elliptic Curve Diffie–Hellman (ECDH) key exchange, so IoT devices can quickly and reliably set up session keys with the Task Manager. In simulations, all devices completed key exchanges successfully, with an average session setup time of 1.2 milliseconds per device. Task decryption was also fast, averaging 0.5 milliseconds per task. These results show that secure task handling is possible even in resource-limited IoT environments. The algorithm gave priority to important and high-security tasks, and always assigned them to high-capacity resources. On average, high-priority tasks waited 0.35 seconds, while low-priority tasks waited 0.58 seconds. High-priority tasks took 1.4 seconds to run, and low-priority tasks took 1.8 seconds. Resource use across edge and cloud nodes stayed balanced, shown by a low standard deviation of 1.2 units, which means the workload was spread out well and the system can scale. Quantitative metrics demonstrate reliable session establishment, rapid task decryption, effective task prioritization, and balanced resource utilization, confirming its practical potential for heterogeneous IoT environments.

Although these results validate the functionality, security, and efficiency of these results show that the algorithm works well in a fully virtualized environment, but the simulation did not use real IoT devices or changing network conditions. The findings might not cover all the challenges found in real-world use, like network delays, different hardware, or unpredictable tasks. Future research need to test the algorithm on real IoT devices and edge-cloud setups to see how it performs in practice and to improve its scheduling and security features and priority-aware framework for IoT systems. Further elaboration will be done in the authentication mechanism in the next work to ensure it is enhanced to offer better security against the Man-in-the-Middle (MITM) attacks. It will include the incorporation of Elliptic Curve Diffie-Hellman-based authenticated key exchange protocols along with digital signatures, identity verification through certificates and secure transport protocols, including Datagram Transport Layer Security. These enhancements focus on enhancing the security of communications without affecting the lightweight characteristic of the proposed system.

References

1. Pandey, V.K., et al., *A lightweight framework to secure IoT devices with limited resources in cloud environments.* Scientific Reports, 2025. **15**(1): p. 26009.
2. Kubba, Z.M.J. and W.A. Shukur. *An enhanced LED cipher algorithm performance for data security in IoT systems*. in *AIP Conference Proceedings*. 2023. AIP Publishing LLC.
3. Shukur, W.A., Z.M.J. Kubba, and S.S. Ahmed, *Novel Standard Polynomial as New Mathematical Basis for Digital Information Encryption Process.* Advances in Decision Sciences, 2023 :(3)27 .p. 72-85.
4. Fereira, R.J., et al., *Energy efficient resource management for real-time IoT applications.* Internet of Things, 2025. **30**: p. 101515.
5. Bhandiwad, V. and L.K. Ragha, *Enhancing the security of IOT enabled systems using light weight hybrid cryptography models.* Cluster Computing, 2025. **28**(3): p. 189.
6. Reddy, M.V.K., et al., *Security, privacy, and trust management of IoT and machine learning-based smart healthcare systems*,

in *Advances in Computers*. 2025, Elsevier. p. 141-174.
7. Alsadie ,D., *Advancements in heuristic task scheduling for IoT applications in fog-cloud computing: challenges and prospects.* PeerJ Computer Science, 2024. **10**: p. e2128.
8. Choppara, P. and S. Mangalampalli, *An Effective analysis on various task scheduling algorithms in Fog computing.* EAI Endorsed Transactions on Internet of Things, 2023. **10**.
9. Liman, A., et al., *Systematic Review of Task Scheduling in Fog–Cloud Computing: Machine Learning and Metaheuristic Approaches.* Journal of Institutional Research, Big Data Analytics and Innovation, 2025. **1**(4): p. 389-420.
10. Ali, H.S. and R. Sridevi, *Mobility and security aware real-time task scheduling in fog-cloud computing for IoT devices: a fuzzy-logic approach.* The Computer Journal, 2024. **67**(2): p. 782-805.
11. Lim ,J., *Latency-aware task scheduling for IoT applications based on artificial intelligence with partitioning in small-scale fog computing environments.* Sensors, 2022. **22**(19): p. 7326.
12. Yang, Y., R. Zhang, and F. Ma, *Empowering Personalized and Privacy-Preserving for Image Super-Resolution with Decentralized Collaboration.* ACM Transactions on Multimedia Computing, Communications and Applications, 2025.
13. Kong, L., et al., *Deadline-Aware Online Scheduling for LLM Fine-Tuning with Spot Market Predictions* .arXiv preprint arXiv:2512.20967, 2025.
14. Kaur, N., et al., *Blockchain-enhanced security and bayesian trust assessment for secure task scheduling in latency-critical fog computing environments.* Journal of Cloud Computing, 2025. **14**(1): p. 53.
15. Chhillar ,R.S., *Performance Evaluation of Hybrid Cloud-Fog Computing Architectures in Smart Home IoT Environments: A Comparative Simulation Study Across Multiple Tools.* Journal of Grid Computing, 2025. **23**(2): p. 1-43.
16. Alvi, A.N., et al., *Secure computing for fog-enabled industrial IoT.* Sensors, 2024. **24**(7): p. 2098.
17. Rahman, Z., et al., *Enhancing AES using chaos and logistic map-based key generation technique for securing IoT-based smart home.* Electronics, 2022. **11**(7): p. 1083.
18. Sarker, K.U., *A systematic review on lightweight security algorithms for a sustainable IoT infrastructure.* Discover Internet of Things, 2025. **5**(1): p. 1-20.
19. Gusita, B., et al., *Securing IoT edge: a survey on lightweight cryptography, anonymous routing and communication protocol enhancements.* Int. J. Inf. Sec., 2025. **24**(3): p. 149.
20. Khalil, K., A. Kumar, and M. Bayoumi, *Hardware Acceleration of CoAP Protocol for High-Speed and Low-Power Internet of Things Communication.* IEEE Internet of Things Journal, 2024.
21. Oliver, S.G. and T. Purusothaman, *Lightweight and Secure Mutual Authentication Scheme for IoT Devices Using CoAP Protocol.* Computer Systems Science & Engineering, 2022. 41.(2**)**
22. Pop, D., E. Kirdan, and M.-O. Pahl. *Performance Comparison of UDP and TCP for Different CoAP Load Profiles*. in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*. 2023. IEEE.
23. Sukumar, N. and S.G. Ranganathan, *A Survey of Security Approaches Based on Elliptic Curve Cryptography.* 2024.
24. Tanksale, V., *Efficient Elliptic Curve Diffie–Hellman Key Exchange for Resource-Constrained IoT Devices.* Electronics, 2024. **13**(18): p. 3631.
25. Aysa, M. H., Ibrahim, A. A., & Mohammed, A. H. (2020, October). Iot ddos attack detection using machine

learning. In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)* (pp. 1-7). IEEE.

26. Bensaid, R., Labraoui, N., Abba Ari, A. A., Maglaras, L., Saidi, H., Abdu Lwahhab, A. M., & Benfriha, S. (2024). Toward a Real-Time TCP SYN Flood DDoS Mitigation Using Adaptive Neuro-Fuzzy Classifier and SDN Assistance in Fog Computing. *Security and Communication Networks*, *2024*(1), 6651584.

27. Ramalakshmi, R., & Kavitha, D. (2024). DDoS attack mitigation using distributed sdn multi controllers for fog based iot systems. *Int. J. Intell. Syst. Appl. Eng*, *12*, 57-69.

28. Songa, A. V., & Karri, G. R. (2024). An integrated SDN framework for early detection of DDoS attacks in cloud computing. *Journal of Cloud Computing*, *13*(1), 64.

29. Tolay, A. (2025). eBPF-Based Real-Time DDoS Mitigation for IoT Edge Devices. *arXiv preprint arXiv:2508.00851*.