International Journal on Applications in Engineering and Technology
Volume 10: Issue 2 : May 2024, pp 16 –22 www.aetsjournal.com ISSN (Online) : 2455 - 0523

DYNAMIC KEY MANAGEMENT FOR IOT NETWORKS USING LIGHTWEIGHT PROTOCOLS AND SPECK SYMMETRIC ALGORITHM WITH CLUSTER-BASED OPTIMIZ

THIRUPPATHY KESAVAN . V , NIRANJANI . K , RAMYA PRIYATHARSINI . T.G, KANIMOZHI . P

Abstract— This article proposes a novel dynamic key management approach for IoT networks, aiming to enhance security and scalability. The key novelty lies in the integration of lightweight protocols, the adoption of the Speck symmetric algorithm, and the application of clustering techniques to optimize network efficiency.The key aspects of this research encompass a dynamic key management process designed to adapt to evolving network conditions. The methodology involves the utilization of the Contiki-NG simulator, offering a realistic representation of IoT network dynamics and resource constraints.Simulation results demonstrate the effectiveness of the proposed method, showcasing improvements in various performance parameters such as energy consumption, memory utilization, latency, communication overhead, and computational overhead. Comparative analysis with an existing dynamic keying techniques called CSDKT for assessing the performance of the proposed dynamic key management approach in the context of IoT networks.

Keywords : Internet of Things; Dynamic Key management; Speak Symmetric algorithm; Clustering; Scalability

I. INTRODUCTION

The Internet of Things (IoT) [1], [2]represents a transformative paradigm that seamlessly integrates physical devices, sensors, and actuators into a networked ecosystem, allowing them to communicate and exchange data. This interconnected web of "things" encompasses a wide array of objects, from everyday household items and industrial machinery to wearable devices and smart city infrastructure. The fundamental goal of IoT is to enhance efficiency, automation, and user experiences by facilitating real-time data exchange and intelligent decision-making.

1) Security Challenges in IoT

While the advantages of IoT are substantial, the proliferation of interconnected devices raises significant security concerns. The very nature of IoT, characterized by a vast and heterogeneous network of devices, amplifies the attack surface for malicious actors. Security challenges in IoT can be categorized into several key areas such as Device Heterogeneity, Data Privacy, Scalability and Network Connectivity[2]. The disadvantages of IoT in terms of security are Inadequate Authentication, Limited Resources, Firmware and Software Vulnerabilities, Key management and Interoperability Challenges. Among these, one of the foremost concerns being the robust management of cryptographic keys in IoT networks.

2) Challenges in Key Management

Managing cryptographic keys in IoT networks is a challenging endeavor due to the dynamic and resource-constrained nature of these devices[1]. The sheer volume and heterogeneity of IoT devices pose a significant challenge in deploying and maintaining secure key management schemes. This complexity is further exacerbated by the mobility of devices and their susceptibility to various security threats[2].

For instance, consider a smart home ecosystem where interconnected devices ranging from door locks to thermostats rely on cryptographic keys for secure communication. In such a scenario, a compromise in key management can lead to unauthorized access, potentially jeopardizing the safety and privacy of the inhabitants.

Thiruppathy Kesavan. V, Faculty of Information Technology, Dhanalakshmi Srinivasan Engineering College, Tamil Nadu, India
Niranjani . K, Faculty of Information Technology, Dhanalakshmi Srinivasan Engineering College, Tamil Nadu, India
Ramya Priyatharsini . T.G, Faculty of Information Technology, Dhanalakshmi Srinivasan Engineering College, Tamil Nadu, India
Kanimozhi . P , Faculty of Information Technology, Dhanalakshmi Srinivasan Engineering College, Tamil Nadu, India

3) Static Key Management Vulnerabilities

Traditionally, static key management approaches have been employed in IoT networks, where a fixed set of cryptographic keys is used for an extended period. However, this static nature proves to be a vulnerability, as intruders can exploit compromised keys over time[3]–[5]. In a dynamic and evolving IoT environment, the inflexibility of static key management renders networks susceptible to malicious activities, leading to unauthorized data access and potential device manipulation.

To illustrate, imagine a smart healthcare system relying on static key management. If a malicious actor gains access to the cryptographic keys, they could manipulate patient data, compromise medical device functionality, and breach confidentiality.

4)Dynamic Key Management Advantages

Recognizing the limitations of static key management, dynamic key management emerges as a compelling solution. Dynamic key management involves the continuous generation and modification of cryptographic keys in response to changes in the network environment[3]–[5]. This adaptability not only enhances the security posture of IoT networks but also mitigates the risks associated with compromised keys.Consider the benefits of dynamic key management in a smart transportation system[6], [7]. As vehicles move within the network, dynamic key updates thwart potential intruders, ensuring secure communication and preventing unauthorized access to critical vehicular systems.

In this context, we propose "SpeckDKM," a novel Dynamic Key Management approach tailored for IoT networks. SpeckDKM leverages the Speck symmetric algorithm for cryptographic operations and integrates clustering techniques to optimize network efficiency. By dynamically managing cryptographic keys, SpeckDKM aims to strengthen IoT networks against evolving security threats while addressing the inherent challenges posed by resource constraints.This paper delves into the details of SpeckDKM, detailing its methodology, cluster-based optimization, and the advantages it offers over traditional static key management approaches. Through comprehensive simulations, we assess SpeckDKM's performance in terms of energy consumption, memory utilization, latency, communication overhead, and computational overhead.

5)Related works

Static key management has traditionally been employed in IoT networks and sensor networks, where fixed cryptographic keys are utilized for an extended duration. Static key management schemes face inherent vulnerabilities that can compromise the security of IoT networks. Commonly observed disadvantages includeLong-Term Key ExposurePerrig, A.[8], Inflexibility to Dynamic EnvironmentsZhu, S.[9]several other challenges in distributing and updating static keys across a large number of devices.While static key management has been a prevalent approach, the identified vulnerabilities underscore the necessity for more adaptive and secure mechanisms, especially in the context of the dynamic and resource-constrained nature of IoT and sensor networks.

The article by Rana M et al. [10] highlights the necessity for tailored lightweight key management schemes in IoT, addressing resource constraints like low processing power. It stresses the significance of scalability and efficiency for secure communication in extensive IoT deployments, managing a growing number of devices. The importance of adaptability to the dynamic nature of IoT networks, including key renewal and addition, is highlighted. The paper calls for future research focusing on application-specific key management schemes for smart homes, healthcare, and industrial IoT, accounting for unique characteristics and security needs.

Thiruppathy Kesavan and Radhakrishnan [11]proposed a Cluster-Based Secure Dynamic Keying Technique for Heterogeneous Mobile Wireless Sensor Networks, emphasizing authentication during node mobility. Cluster heads are strategically selected based on weighted parameters, and dynamic key generation enhances security. The bidirectional malicious node detection technique eliminates potential threats. Simulations validate efficient security and reduced energy consumption during mobility. Comparative analysis

demonstrates the proposed technique's effectiveness in preventing both insider and outsider attacks.

Vipin Kumar[3] has extensively explored cryptographic techniques, emphasizing the pivotal role of key management in ensuring the integrity, authentication, and confidentiality of WSNs. While existing studies have proposed various key management schemes, the abstract and conclusion of the present research introduce a novel approach, SSEKMS, designed to address challenges in dynamic key distribution and management in WSNs. Notably, the literature gap addressed by SSEKMS lies in its focus on storage efficiency, resiliency against node capture, and energy efficiency, although a more comprehensive evaluation and quantitative analysis of these aspects would contribute to a more nuanced understanding of its comparative advantages in the existing landscape.

Hua Yi Lin & Meng-Yen Hsieh[5] addresses the critical challenge of information security in the context of the Internet of Vehicles (IoV), where personal details are exposed within the open communication environment. Focusing on the advancements in broadband wireless networks and 5G, the study proposes a multi-level security infrastructure employing an M-tree based elliptic curve digital signature algorithm (ECDSA). Notably, the research contributes to the field by integrating M-tree key management with secure data transmission, providing adaptability to the dynamic IoV topology and reducing the phases required to resynchronize the system key. They have highlighted the effectiveness of the proposed key management system in adaptable and expandable IoV environments, emphasizing its operational and communication cost reductions compared to conventional methods. Furthermore, the study underscores the computational efficiency achieved through the use of M-tree and simplified cryptographic operations, ensuring information security and secure IoV communication.

Yuxiang Zhou[7] proposed a novel authentication and key agreement scheme based on challenge authentication handshake protocols. The scheme prioritizes mutual authentication, session key security, and resistance against common attacks to ensure secure communication between vehicles and roadside units (RSUs). Notably, the research emphasizes the flexible implementation of time keys for dynamic vehicle management, providing a unique advantage over existing schemes. The proposed scheme is further validated through a formal security proof under the random oracle model, showcasing its reliability. The conclusion reinforces the significance of the developed scheme, emphasizing its achievement of secure authentication, forward security, and resistance against common network attacks.

Various optimization techniques have been suggested to enhance the efficiency of IoT network clusters. Notably, Azimi [12] and Al-Janabi[13] concentrate on load balancing and latency reduction. Azimi employs a particle swarm optimization algorithm, while Al-Janabi introduces a load-balanced PSO clustering algorithm. Addressing energy efficiency, Iwendi[14] adopts a hybrid metaheuristic algorithm for Cluster Head selection, while Alazab[15] presents a multi-objective approach for CH selection, incorporating a modified Rider Optimization Algorithm. Collectively, these studies underscore the significance of considering diverse factors—such as load balancing, latency, and energy efficiency—in the optimization of IoT network clusters.

Advances in lightweight cryptographic algorithms and optimization techniques will play a crucial role in shaping the future landscape of key management in these networks.

6)SpeckDKM - Dynamic Key Management with Clustering

The Dynamic Key Management process is designed to ensure the continuous generation and modification of cryptographic keys in response to changes in the network environment. The process leverages the Speck symmetric algorithm [16]for secure key operations and incorporates clustering optimization to enhance scalability and efficiency. The entire process is explained below:

1) Key Generation

Speck Symmetric Algorithm

- The Speck symmetric algorithm is employed for key generation, denoted as K_i at each time instance i.
- The key generation process can be represented as:

K_i=Speck_Key_Generation(K_{i-1})

Here, K_{i-1} is the key from the previous time instance.

The pseudocode the key generation phase is given below:

Algorithm 1: Speck Key Generation

```
Input:
  - Previous Key: K_{i-1}
Output:
  - New Key: K_i
Parameters:
  - Word Size: w
  - Number of Rounds: r
  - Key Size: k
Constants:
  - Alpha: 8
  - Beta: 3
Function: Speck Key Generation(K_{i-1})
1. Initialize Key Schedule:
L = K_{i-1}[0 : w-1]
R = K_{i-1}[w : 2w-1]
2. Perform Key Expansion:
  for round = 1 to r:
    L = (L + R) <<< Beta
    L = L XOR round_constant
    R = (R XOR L) <<< Alpha
3. Generate New Key:
K_i = Concatenate(L, R)
4. Return K_i
```

2) Clustering Optimization

Weighted Parameters

- Determine the weight value (W_i) for each node based on parameters such as node degree (*ND*), average distance (D_{av}), node speed (S_{av}), and virtual battery power (*VBP*).

$$W_i = w_1 \cdot ND + w_2 \cdot D_{av} + w_3 \cdot S_{av} + w_4 \cdot VBP$$

- Normalize the weights to ensure a unified scale.

Cluster Head Selection

- Select high-configured nodes as cluster heads based on the calculated weight values

$$CH_i = \text{Select\_Cluster\_Head}(W_i)$$

The pseudocode for clustering optimization is given as Algorithm 2 in which the adjustments may be required based on specific considerations and network characteristics:

Algorithm 2: Clustering Optimization

```
Input:
- Weighted Parameters: ND, Dav, Sav, VBP
- Network Nodes Information
- Number of Nodes: N
- Cluster Formation Threshold: Threshold
Output:
- Cluster Heads
Parameters:
- Weight Factors: w1, w2, w3, w4
Function: Clustering Optimization(ND, Dav, Sav, VBP, N, Threshold)
1. Initialize Empty Cluster Head Set: CH_set = {}
2. Calculate Weighted Values for Each Node:
  for each node in Network:
W_node = w1 * ND[node] + w2 * Dav[node] +_
      w3 * Sav[node] + w4 * VBP[node]
3. Normalize Weighted Values:
W_normalized = Normalize(W_node)
4. Identify High-Configured Nodes as Cluster Heads:
  for each normalized weight in W_normalized:
    if normalized weight > Threshold:
      Add corresponding node to CH_set
5. Return CH_set
Function: Normalize(W_node)
1. Calculate Min and Max of W_node:
W_min = min(W_node)
W_max = max(W_node)
2. Normalize W_node:
W_normalized = (W_node - W_min) / (W_max - W_min)
3. Return W_normalized
```

Algorithm 3: Cluster Head Selection

```
Input:
- Weighted Parameters: ND, Dav, Sav, VBP
- Normalized Weight Threshold: Threshold
- Node Information
- Network Topology
Output:
- Cluster Heads
Parameters:
- Weight Factors: w1, w2, w3, w4
Function: Select Cluster Head (ND, Dav, Sav, VBP, Threshold)
1. Initialize Empty Cluster Head Set: CH_set = {}
2. Calculate Weighted Values for Each Node:
  for each node in Network:
W_node = w1 * ND [node] + w2 * Dav[node] +_
      w3 * Sav[node] + w4 * VBP [node]
3. Normalize Weighted Values:
W_normalized = Normalize(W_node)
4. Identify High-Configured Nodes as Cluster Heads:
  for each normalized weight in W_normalized:
    if normalized weight > Threshold:
      Add corresponding node to CH_set
5. Return CH_set
```

The *Threshold* is a predefined threshold for selecting high-configured nodes as cluster heads.The *Normalize* function scales the weighted

values between 0 and 1.TheAlgorithm 2 builds upon the previous step and focuses specifically on selecting high-configured nodes as cluster heads based on the normalized weighted values

3)Key Modification and Adaptation

Dynamic Key Modification

- Dynamically modify the cryptographic keys to adapt to changes in the network environment.

$K_i = \text{Speck\_Key\_Modification}(K_i)$

- *Adaptive Key Renewal*
  - Implement adaptive key renewal processes to enhance the security of the network.

$Key\_Renewal\_Policy = Adaptive\_Renewal(K_i)$

The pseudocode for step 3 is given in Algorithm 4.This pseudocode provides a structure for the dynamic key modification process and introduces adaptive key renewal based on network dynamics.

Algorithm 4: Key Modification and Adaptive Renewal

Input:
- Current Key: K_i
- Previous Key: K_{i-1}
- Network Dynamics Information
- Security Parameters
Output:
- Updated Key: K_i
Parameters:
- Key Renewal Threshold: Renewal_Threshold
- Adaptive Renewal Function: Renewal_Function()
Function: *Modify and Renew Key(K_{i-1}, K_i, Network Dynamics)*
1. Perform Dynamic Key Modification:
  - Implement a secure dynamic modification function based on cryptographic principles.
  - Example: K_i = Cryptographic_Modification(K_{i-1})
2. Check for Adaptive Key Renewal:
  IfRenewal_Function(Network Dynamics) >Renewal_Threshold:
    - Implement Adaptive Key Renewal Mechanism:
K_i = Renewal_Mechanism(K_{i-1}, Network Dynamics)
3. Return K_i

4)Security Enhancement Mechanisms

Key Compromise Mitigation

- Introduce mechanisms to mitigate the impact of key compromises, preventing attackers from exploiting compromised keys.

Mitigated_Key=Mitigation_Mechanism(*Compromised_Key*)

Bidirectional Malicious Node Detection

- Employ bidirectional detection to identify and eliminate malicious nodes from the network.

Detected_Malicious_Nodes=Bidirectional_Detection(*Network_State*)

This pseudocode given in Algorithm 5 provides a foundation for implementing security enhancement mechanisms, focusing on mitigating the impact of key compromises and detecting potentially malicious nodes bidirectionally in the network.

Algorithm 5: Security Enhancement Mechanisms

Input:
- Compromised Key: Compromised_Key
- Network State: Current Network State
Output:
- Mitigated Key: Mitigated_Key
- Detected Malicious Nodes: List of Detected Malicious Nodes
Parameters:
- Threshold for Malicious Node Detection: Detection_Threshold
Function: *Key Compromise Mitigation(Compromised_Key)*
1. Implement Key Compromise Mitigation Mechanism:
  - Use cryptographic techniques to mitigate the impact of a compromised key.
  - Example: Mitigated_Key = Cryptographic_Mitigation(Compromised_Key)
2. Return Mitigated_Key
Function: *Bidirectional Malicious Node Detection(Current Network State)*
1. Initialize Empty List for Detected Malicious Nodes: Detected_Malicious_Nodes = []
2. Perform Bidirectional Detection:
  for each node in Current Network State:
    if Is_Malicious(Node):
      Add Node to Detected_Malicious_Nodes
3. Return Detected_Malicious_Nodes
Function: *Is_Malicious(Node)*
1. Implement Malicious Node Detection Logic:
  - Use criteria such as abnormal behavior, communication patterns, or known attack signatures.
  - Example: if Node's Behavior indicates malicious activity:
      return True
    else:
return False

II. RESULTS AND DISCUSSIONS

In this section, we present and analyze the results of our proposed Dynamic Key Management in IoT Networks using the SpeckDKM. Comparative evaluations with the CSDKT reveal notable insights into various key performance parameters. In our simulation environment, we precisely replicated a realistic IoT network scenario using the Contiki-NG simulator. Leveraging this powerful tool, we

emulated various network dynamics and conditions to assess the performance of our proposed Dynamic Key Management system—SpeckDKM. The simulation provided a controlled yet representative platform, enabling a thorough exploration of key parameters such as energy consumption, memory utilization, latency, communication overhead, and computational overhead. This controlled environment facilitated a comprehensive understanding of SpeckDKM'sbehavior in comparison to the established CSDKT, establishing a robust foundation for our results and discussions.

These results illuminate the efficacy of SpeckDKM in balancing enhanced security with minimal impact on resource-constrained IoT environments, setting the stage for a comprehensive exploration of our findings. The parameters which are discussed in this section are:

1. **Energy Consumption:** Analyze how SpeckDKM's lightweight cryptographic operations impact energy usage.
2. **Memory Utilization:** Assess how the lightweight nature of the Speck algorithm influences memory requirements.
3. **Latency:** Analyze how the dynamic key management process affects communication delays.
4. **Communication Overhead:** Compare the impact on the overall network traffic between SpeckDKM and CSDKT.
5. **Computational Overhead:** Evaluate the processing requirements for dynamic key modification and adaptive renewal.

The Tables 1 to 4 provides the simulation results for the above said parameters respectively.

Table 1: Energy Consumption

Simulation Scenario	SpeckDKM	CSDKT
Base Energy Consumption	500 mJ	700 mJ
Energy consumed	520 mJ	730 mJ
Energy Savings	3.8%	-3.7%

The Base Energy Consumption represents the energy consumption in a baseline scenario without dynamic key management.SpeckDKM shows a 3.8% reduction in energy consumption compared to the baseline, suggesting improved energy efficiency.CSDKT results in a 3.7% increase in energy consumption compared to the baseline, potentially due to higher computational demands or increased communication overhead.

Table 2: Memory Utilization

Simulation Scenario	SpeckDKM	CSDKT
Base Memory Utilization	120 KB	100 KB
Memory usage	132 KB	133KB
Additional Memory Used	15 KB	30 KB

The Base Memory Utilization represents the memory usage in a baseline scenario without dynamic key management.SpeckDKM introduces an estimated additional 15 KB of memory compared to the baseline. While this represents a moderate increase, it is crucial to note that the Speck algorithm's lightweight nature contributes to relatively efficient memory usage.CSDKT, on the other hand, demonstrates a higher increase of 30 KB in memory compared to the baseline. This potentially indicates higher memory overhead, which may impact scalability in resource-constrained environments.

Table 3: Latency

Simulation Scenario	SpeckDKM	CSDKT
Base Latency	5 ms	4 ms
Average Latency	6 ms	6.5
Latency Increase	1ms	1.5ms

The Base Latency represents the latency in a baseline scenario without dynamic key management.SpeckDKM introduces an estimated additional latency of 1 ms compared to the baseline. This slight increase can be attributed to the dynamic key modification and adaptive renewal processes.CSDKT exhibits a higher latency increase of 1.5 ms compared to the baseline, indicating potentially higher communication delay. This may be due to the complexity of the clustering optimization process.

Table 4: Communication Overhead

Simulation Scenario	SpeckDKM	CSDKT
Base Communication Overhead	884 bits	800 bits
Communication Overhead	950 bits	1050 bits

Additional Communication Overhead	50 bits	100 bits
Computation Overhead	2 ms	3 ms

The Base Communication Overhead represents the communication overhead in a baseline scenario without dynamic key management, measured in bits. Additional Communication Overhead indicates the increase in communication overhead compared to the baseline or between the two methods. SpeckDKM introduces an additional communication overhead of 50 bits, suggesting a moderate increase in the amount of data transmitted. This overhead is attributed to the dynamic key modification and adaptive renewal mechanisms.CSDKT exhibits a higher communication overhead increase of 100 bits compared to the baseline. This higher overhead may be attributed to the clustering and key management processes, potentially impacting network scalability.CSDKT incurs a higher computational overhead of 3 ms, potentially due to the complex clustering optimization and key management processes.

Overall, SpeckDKM demonstrates a balanced trade-off between enhanced security through dynamic key management and acceptable increases in energy consumption, memory utilization, latency, communication overhead, and computational overhead.CSDKT, while providing secure dynamic keying, exhibits higher increases in various performance parameters, which may impact its suitability for resource-constrained IoT environments.

III. CONCLUSION

In conclusion, our proposed Dynamic Key Management in IoT Networks, leveraging the SpeckDKM, demonstrates promising outcomes in the realm of lightweight and secure key management. Comparative analyses against the CSDKT showcase SpeckDKM'ssubtle balance between security and resource efficiency in energy consumption, memory utilization, latency, communication overhead, and computational overhead. These findings underscore SpeckDKM'spotential as a robust solution for securing IoT networks with constrained resources, opening avenues for further exploration and implementation in real-world scenarios.

REFERENCES

[1] F. Samiullah, M. L. Gan, S. Akleylek, and Y. Aun, “Group Key Management in Internet of Things: A Systematic Literature Review,” *IEEE Access*, vol. 11, 2023, doi: 10.1109/ACCESS.2023.3298024.

[2] N. A. Khan, A. Awang, and S. A. A. Karim, “Security in Internet of Things: A Review,” *IEEE Access*, vol. 10. 2022. doi: 10.1109/ACCESS.2022.3209355.

[3] V. Kumar, N. Malik, G. Dhiman, and T. K. Lohani, “Scalable and Storage Efficient Dynamic Key Management Scheme for Wireless Sensor Network,” *WirelCommun Mob Comput*, vol. 2021, 2021, doi: 10.1155/2021/5512879.

[4] V. T. Kesavan, “Scalable and Secure Dynamic Key Management Framework for Static and Mobile Wireless Sensor Networks,” KALASALINGAM UNIVERSITY, 2015.

[5] H. Y. Lin and M. Y. Hsieh, “A dynamic key management and secure data transfer based on m-tree structure with multi-level security framework for Internet of vehicles,” *Conn Sci*, vol. 34, no. 1, 2022, doi: 10.1080/09540091.2022.2045254.

[6] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah, and Z. Sun, “Blockchain-Based Dynamic Key Management for Heterogeneous Intelligent Transportation Systems,” *IEEE Internet Things J*, vol. 4, no. 6, 2017, doi: 10.1109/JIOT.2017.2740569.

[7] Y. Zhou, H. Tan, and K. C. A. A. Iroshan, “A secure authentication and key agreement scheme with dynamic management for vehicular networks,” *Conn Sci*, vol. 35, no. 1, Dec. 2023, doi: 10.1080/09540091.2023.2176825.

[8] A. Perrig*et al.*, “SPINS: Security Protocols for Sensor Networks SPINS : Security Protocols for Sensor Networks,” *Wireless Networks*, vol. 8, no. September, 2009.

[9] S. Zhu, S. Setia, and S. Jajodia, “LEAP+: Efficient security mechanisms for large-scale distributed sensor networks,” *ACM Trans Sens Netw*, vol. 2, no. 4, 2006, doi: 10.1145/1218556.1218559.

[10] M. Rana, Q. Mamun, and R. Islam, “Enhancing IoT Security: An Innovative Key Management System for Lightweight Block Ciphers,” *Sensors*, vol. 23, no. 18, 2023, doi: 10.3390/s23187678.

[11] V. T. Kesavan and S. Radhakrishnan, “Cluster based secure dynamic keying technique for heterogeneous mobile Wireless Sensor Networks,” *China Communications*, vol. 13, no. 6, pp. 178–194, 2016, doi: 10.1109/CC.2016.7513213.

[12] S. Azimi, C. Pahl, and M. H. Shirvani, “Particle swarm optimization for performance management in multi-cluster IoT edge architectures,” in *CLOSER 2020 - Proceedings of the 10th International Conference on Cloud Computing and Services Science*, 2020. doi: 10.5220/0009391203280337.

[13] T. A. Al-Janabi and H. S. Al-Raweshidy, “Optimised clustering algorithm-based centralised architecture for load balancing in IoT network,” in *Proceedings of the International Symposium on Wireless Communication Systems*, 2017. doi: 10.1109/ISWCS.2017.8108123.

[14] C. Iwendi, P. K. R. Maddikunta, T. R. Gadekallu, K. Lakshmanna, A. K. Bashir, and M. J. Piran, “A metaheuristic optimization approach for energy efficiency in the IoT networks,” *SoftwPract Exp*, vol. 51, no. 12, 2021, doi: 10.1002/spe.2797.

[15] M. Alazab, K. Lakshmanna, T. R. G, Q. V. Pham, and P. K. Reddy Maddikunta, “Multi-objective cluster head selection using fitness averaged rider optimization algorithm for IoT networks in smart cities,” *Sustainable Energy Technologies and Assessments*, vol. 43, 2021, doi: 10.1016/j.seta.2020.100973.

[16] R. A. F. Lustro, A. M. Sison, and R. P. Medina, “Performance analysis of enhanced speck algorithm,” in *ACM International Conference Proceeding Series*, 2018. doi: 10.1145/3288155.3288196.