2022

Secure Authenticated Key Exchange for Enhancing the Security of Routing Protocol for Low-Power and Lossy Networks

Sarah Mohammed Alzahrani
Wright State University

SECURE AUTHENTICATED KEY EXCHANGE FOR ENHANCING THE SECURITY OF ROUTING PROTOCOL FOR LOW-POWER AND LOSSY NETWORKS

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Cyber Security

by

SARAH MOHAMMED ALZAHRANI

Bach., Umm Al-Qura University, Kingdom of Saudi Arabia, 2015

2022

Wright State University

WRIGHT STATE UNIVERSITY

GRADUATE SCHOOL

05/25/2022

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Sarah Mohammed Alzahrani ENTITLED Secure Authenticated Key Exchange for Enhancing the Security of Routing Protocol for Low-Power and Lossy Networks BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Cyber Security.

_____

Bin Wang, Ph.D.

Thesis Director

_____

Michael Raymer, Ph.D.
Chair, Department of Engineering and Computer Science

Committee on Final Examination:

_____

Bin Wang, Ph.D.

_____

Meilin Liu, Ph.D.

_____

Zhiqiang Wu, Ph.D.

_____

Barry Milligan, Ph.D.
Vice Provost for Academic Affairs
Dean of the Graduate School

ABSTRACT

Alzahrani, Sarah Mohammed. M.S.C.S., Department of Computer Science and Engineering, Wright State University, 2022. Secure Authenticated Key Exchange for Enhancing the Security of Routing Protocol for Low-Power and Lossy Networks

The current Routing Protocol for Low Power and Lossy Networks (RPL) standard provides three security modes Unsecured Mode (UM), Preinstalled Secure Mode (PSM), and Authenticated Secure Mode (ASM). The PSM and ASM are designed to prevent external routing attacks and specific replay attacks through an optional replay protection mechanism. RPL's PSM mode does not support key replacement when a malicious party obtains the key via differential cryptanalysis since it considers the key to be provided to nodes during the configuration of the network. This thesis presents an approach to implementing a secure authenticated key exchange mechanism for RPL, which ensures the integrity and authentication of the received key while providing tamper-proof data communication for IoTs in insecure circumstances. Moreover, the proposed approach allows the key to be updated regularly, preventing an attacker from obtaining the key through differential cryptanalysis. However, it is observed that the proposed solution imposes an increase in the cost of communication, computation, power consumption, and memory usage for the network nodes.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

I received a lot of help and assistance while writing this thesis; firstly, I would like to thank my supervisor, Prof Bin Wang. Without his encouragement, support, and continuous optimism this thesis would hardly have been completed. I want to express my heartfelt appreciation to my family for their unconditional love, help, and support. I am grateful to my sisters for always being friends to me. I will be eternally grateful to my parents for providing me with the opportunities and experiences that have shaped who I am.

1. INTRODUCTION

The Internet of Things (IoTs) includes billions of connected devices that can detect, communicate, and process data. A physical device, such as a sensor, is a component of an IoT network and is given the capacity to obtain and share critical data with other devices. IoTs may be used to create a variety of critical systems, including health care systems, traffic management systems, and smart manufacturing systems. However, these IoT devices are vulnerable to various insider and outsider threats, putting users' security and privacy at risk. The IPv6 Routing Protocol for Low Power and Lossy Network (RPL) is described in RFC 6550 [1] by the IETF ROLL working group to enable effective routing for 6LoWPAN networks despite their limitations. Because of the resource-constrained nature of IoT networks, RPL is subject to a variety of attacks that may exhaust the node's resources and affect network performance.

This thesis shows that the standard RPL protocol has many security flaws, such as a lack of key management. Our focus is on RPL security enhancement. We propose an enhanced protocol to manage the key distribution over the nodes of the network and deal with the security weaknesses. Our proposal is implemented using Elliptic-curve cryptography and the SHA-256 algorithm. After many tests on the Cooja simulator, the results showed that our protocol guarantees a good level of security compared to the standard RPL protocol, which ensures the integrity and authentication of the received key while providing accurate and tamper-proof data communication for IoTs in insecure circumstances.

The rest of the thesis is organized as follows. Chapter 1 gives an overview of IoTs and their applications, followed by discussing certain IoT security concerns, and concludes with an overview of asymmetric cryptography. We briefly introduce the RPL protocol and related security challenges, focusing on key management in RPL in chapter 2. Chapter 3 is a

summary of some related work. Chapter 4 presents the proposed protocol in detail. The security analyses of the proposed protocol are discussed in Chapter 5. The details of the implementation and the evaluation of the performance in terms of computational cost and message overhead are presented in Chapter 6. Finally, we finish with conclusions and future work in Chapter 7.

1.1 Background

The Internet of Things (IoTs) and its applications are discussed in this section, and a wide range of applications such as smart cities, smart grids, smart environment, smart farming, and smart retail, etc. Furthermore, the most common threats to such a system are also investigated. Finally, an overview cryptography is provided at the end of this chapter.

1.1.1 Overview of Internet of Things (IoTs)

IoTs is a collection of interconnected devices with sensing, computational and communication capabilities. It is now being used in a variety of domains such as smart homes, self-driving cars, smart grids, smart fire alarms, etc. According to estimates, there will be over 75 billion IoT-connected devices by 2025 [2]. The design and resource constraints of many IoT devices have resulted in security vulnerabilities, even though IoT technology has provided enormous benefits to society. The two keywords security and privacy are frequently used to define IoTs' major challenge [3]. IoT devices provide limited user control, resulting in security and privacy concerns. As a result, users are vulnerable to various malicious activities [4].

1.1.2 IoT applications and challenges

IoT systems are used to develop various fields such as smart electric grids, retail, and farming. Security and privacy are critical in almost all IoT applications. Some security issues in IoT systems are covered in this section.

Smart City

The world is witnessing the rise of Smart Cities. Smart energy meters, security cameras, and smart health and home equipment provide remarkable conveniences and

increased quality of life [5]. These are the result of advancements in information technology that, while creating new economic and social opportunities, also represent a threat to our security and privacy concerns. By taking control of traffic management systems, for example, hackers might cause simultaneous traffic tie-ups on important city highways enough to create gridlock and impede law-enforcement teams from getting to the site of a crime.

Smart Agriculture

Smart agriculture technologies include monitoring soil moisture, maintaining microclimate conditions, controlling air temperature, controlling humidity, and selecting watering in dry areas. The application of such innovative features in agriculture can help farmers achieve higher yields and avoid financial losses. Smart farm owners use sensors to monitor humidity, rainfall, and temperature to properly irrigate crops and identify the best harvest dates. Manipulation of this sensor data might result in severe agricultural damage such as cutting off food to communities or even spreading disease.

Smart Home Automation

The smart home concept is based on providing comfort through connectivity and system automation for efficiency. This includes, for example, smart technologies for automatically controlling electronic devices to save energy and smart alarm systems to add a level of security. However, a hacker can potentially obtain illegal access to smart devices in people's homes and tries to harm them. For example, after the implementation of different home automation systems, the number of house thefts has grown dramatically [6]. Hackers utilize platforms like the Shodan search engine and the Mirai botnet to look for firmware vulnerabilities in commercially available smart devices.

Smart Retail Applications

The IoT revolution contributes to enhance various areas for companies and businesses such as inventory management systems, logistics, customer experience, supply chain management, etc. For instance, enterprises may use IoT devices to improve their operations. With the monitoring of products, inventory tracking, and management become more accessible as RFID and GPS technologies enable the tracking of every stage of a

shipment, including the journey and weather. An attacker may steal sensitive personal information such as credit card details, email addresses, mobile numbers, and other personal information if security mechanisms are not included in the smart store.

Smart Grid

A smart grid is an IoT application that allows providers and their customers to share electricity and data. It includes smart meters, renewable energy resources, and smart appliances. However, these systems are vulnerable to both physical and cyber-attacks. A cyber-attack on smart grid systems could plunge a city into darkness and smart meter security flaws might lead to fraud or data leaks.

1.1.3 IoT security issues

IoT applications include the sensing, network, middleware, and application layers. These layers employ various technologies that introduce several challenges and security vulnerabilities.

Node Capturing:

Sensors and actuators are examples of low-power nodes used in IoT applications. A wide range of malicious activities can affect these network nodes. A hacker can replace a legitimate node with a malicious one. Although this malicious node seems to be part of the system, it is actually under the hacker's control. This can affect the IoT application's overall security [7].

DoS Attack:

In this attack, a hacker floods a large number of unwanted requests to the target, thereby disrupting services to actual users. This type of attack isn't limited to IoT applications, but many IoT devices are weakly configured, making them ideal targets for DoS attack.

Man-in-the-Middle Attack:

An attacker can gain communication between two devices. It is a risky attack since the attacker masquerades as the original sender. Because the attacker knows the initial

communication, they can trick the receiver into thinking they are still receiving a valid message.

Data Transit Attack:

There is a great extent of data transit in IoT systems between sensors, IoT getaways, and clouds. These systems are vulnerable to several data breaches due to the employment of many connection technologies in this data exchange.

Routing Attack:

The routing attack is an attack on the network layer, including manipulation of routing information, wormhole attacks, and sinkhole attacks. In IoT systems, a malicious node may attempt to change routing directions during data transmission. An attacker advertises the shortest routing path to induce nodes to route traffic via it, which is known as a sinkhole attack. Attackers can also use a wormhole attack to capture packets from one position and route them to another location in the IoT network.

1.1.4 Cryptography

Cryptography is a term used to describe secure information and communication procedures that use mathematics to encrypt and decrypt data in order to convert communications in difficult-to-decipher ways. A cryptographic algorithm works in combination with a key to encrypt the plaintext. The same plaintext can be encrypted with different keys to produce different ciphertexts. The strength of the cryptographic method and the secrecy of the key are both critical to the security of encrypted data. Asymmetric cryptography and symmetric cryptography are the two types of encryption techniques. This section provides a basic introduction to asymmetric cryptography.

1.1.4.1 Asymmetric Cryptography

Encryption is used to protect data transferred from one system to another. In symmetric key cryptography, the transmitter and receiver utilize a shared secret key to encrypt and decrypt. There are several symmetric key methods that have been proven to be quite secure, but the major issue with this type of system is that the key must be exchanged

over a public network and is usually pre-deployed on each sensor node. Asymmetric key cryptography is used to solve problems that symmetric key encryption cannot handle. Asymmetric key cryptography generates two keys, one private key and one public key. The private key is kept private while the public key is made public. The communication is encrypted using the public key of the receiver, and the message is decrypted with the private key of the receiver.

1.1.4.2 Elliptic Curve Cryptography (ECC)

In recent years, researchers in the field of IoT have faced a serious problem in reducing the computational complexity and memory utilization of standard asymmetric cryptographic algorithms such as RSA, DSA, and ECC [8]. Among these algorithms, ECC appears to be most suitable since it consumes the least amount of memory and resources compared to the others [9]. It is a better asymmetric algorithm than RSA due to the level of security achieved by using a small key size. ECC with a 162-bit key size is as secure as RSA, which uses a 1024-bit key size. ECC has been accepted as an alternative and effective cryptographic algorithm by IEEE, ISO, and NIST. This cryptographic algorithm offers a wide range of security functions, including key exchange, encryption, message integrity, and authentication.

ECC was proposed by Koblitz and Miller in 1985. It is based on the algebraic structures of the elliptic curves over finite fields and on the difficulty of the elliptic curve discrete logarithm problem. Based on the math of elliptic curves over finite fields, ECC provides many types of algorithms: ECC digital signature algorithms such as ECDSA, ECC encryption algorithms like the ECIES, and ECC key agreement algorithms like ECDH. These algorithms use public and private key pairs, where the private key is an integer, and the public key is an elliptic curve point. The following is an example of a typical elliptic curve: $y^2 = x^3+ax+b$. Figure1 illustrates two examples of elliptical curves.

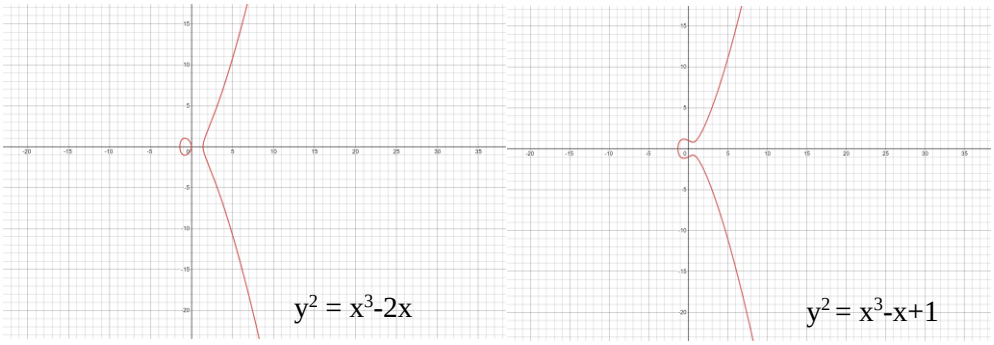

Figure 1: Examples of elliptical curves

1.1.4.3 The Elliptic Curve Diffie–Hellman Key Exchange (ECDH)

ECDH is a key agreement mechanism that allows two parties to create a shared secret via an unsecured channel using an elliptic-curve public/private key pairs. It is very similar to the classic Diffie–Hellman Key Exchange algorithm. Instead of modular exponentiations, it uses ECC point multiplication. The steps for two sensor nodes A and B to agree upon a shared secret key are as follows.

1. A generates a random ECC key pair: APrivKey, APubKey = APrivKey * G (where G is the generator point on the elliptic curve)
2. B generates a random ECC key pair: BPrivKey, BPubKey = BPrivKey * G (where G is the generator point on the elliptic curve)
3. A and B exchange their public keys APubKey and BPubKey.
4. A computes sharedKey (APrivKey* BPubKey). B computes sharedKey(BPrivKey * APubKey).
5. Now both A and B have the same shared = KeyAPrivKey * BPubKey= BPrivKey * APubKey

1.1.4.4 ECC-Based Hybrid Encryption / Decryption

An asymmetric encryption scheme based on ECC can be used for encryption by combining the key agreement mechanism, ECDH, with an asymmetric encryption scheme such as Advanced Encryption Standard (AES). ECC does not directly offer an encryption scheme. Instead, it provides a hybrid encryption scheme by using ECDH to generate a shared secret key to use in symmetric data encryption and decryption.

2. RPL

This chapter briefly introduces the routing protocol for Low Power Lossy Networks (LLNs), called Routing Protocol Low Power Lossy Networks (RPL). Then we discuss the key management in RPL. Finally, we summarize some attacks against RPL.

2.1 Overview

RPL is an LLN routing protocol standard established by the IETF ROLL task force and is specified in RFC 6550 [1]. It is a standard networking protocol for Internet of Things (IoTs) devices. It is a distance-vector tree-based method developed for IPv6 devices. RPL works on the IEEE 802.15.4 standard, with a 6LoWPAN adaption layer. The topological concept of Destination Oriented Directed Acyclic Graphs (DODAGs) is one of the main concepts of RPL. The DODAG topology looks like a tree and has leaves on the edges. DODAG nodes are all linked to a single sink node known as the DODAG root. These DODAGs are referred to as the DAG, and they span the whole network. Figure 2 shows a DAG with a single DODAG.

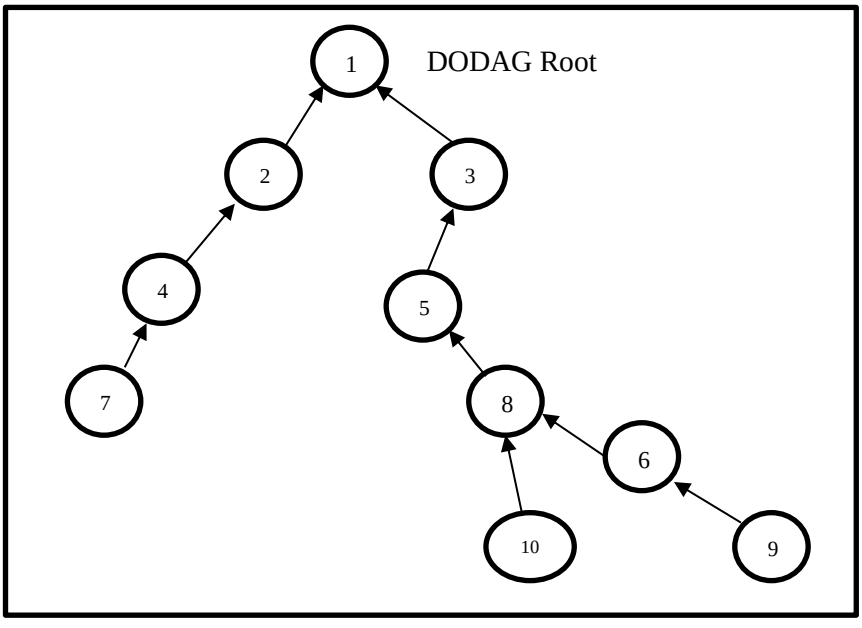

Figure 2: A RPL DAG with a single DODAG

The routing path in a DODAG is determined by Objective Function OF through defining the routing metrics, optimization objectives, and associated functions. The IETF has standardized two OFs: Objective Function Zero (OF0) and Minimum Rank Hysteresis Objective Function (MRHOF). OF0 finds the shortest path to the root using the minimal number of hops, while MRHOF uses the Expected Transmission Count (ETX) as a route optimization metric [10]. This metric is the probable number of transmissions necessary to send and receive packets successfully across a network. ETX allows RPL to find the stable minimum paths from the nodes to a root in the DAG.

Feature	Description
Target networks	LLN, IPv6 and 6LowPAN networks
Routing metrics	Based on Objective Function
Network Topology	Mesh, tree based on directed acyclic graph (DAG)
Control messages	DIS, DIO, DAO, and DAO-ACK.
Modes of security	UM, PSM and ASM
Modes of operation	Non-storing and storing

Table 1: Summary of RPL protocol

RPL's main concepts and features, as described in IETF are as follows:

RPL control messages:

RPL provides additional control messages to support the DODAG and the communication paths. These control messages are a new kind of Internet Control Messages Protocol for IPv6 (ICMPv6).

- DODAG Information Solicitation (DIS): A network node uses a DIS message to request information about its neighbor nodes.
- DODAG Information Object (DIO): A message is sent by the DODAG root to create a new DAG and then broadcast via the DODAG structure. It allows to scan RPL instances, get a preferred parent, and absorb the network configurations.

- Destination Advertisement Object (DAO): A message is used to reverse route information in order to keep track of the nodes visited on the upward path. Each node, besides the DODAG root, sends DAO messages to update the routing tables with their children's prefixes and to advertise their addresses and prefixes to their parents.
- Destination Advertisement Object Acknowledgement (DAO-ACK) A message is used to reply to a DAO message.

RPL Security:

IoT networks can use link-layer security mechanisms, such as the IEEE 802.15.4 MAC standard's security measures when they are available for securing message transmissions. These security measures provide four essential security services: access control, message integrity, message confidentiality, and replay protection.

Besides link-layer security, RPL has its security mechanisms, which are typically available in three different security modes:

- Unsecured Mode (UM): RPL control messages are transmitted without any security features in this mode. In this case, link-layer security could be implemented to meet security requirements.
- Pre-installed Mode (PSM): In this mode, the symmetric key is preconfigured on the network node. When nodes join the DODAG, they use the preinstall key to encrypt and decrypt control messages to protect the transmitted data.
- Authenticated Mode (ASM): Nodes in this mode have pre-installed keys, like in pre-installed mode. However, the pre-installed key can only be used to join an RPL instance as a leaf. Obtaining a key from an authentication authority is required before joining an authorized RPL instance as a router. It is up to the implementation to determine how to authenticate the network nodes and provides the key [1].

Furthermore, RPL provides an additional replay protection technique that uses Consistency Check (CC) messages [1], only enabled in PSM and ASM modes. These checks

compare a nonce provided within CC secure transactions with the stored value to see if the received nonce is reused by the node. As a result, a replay attack can be discovered.

2.2 Key Management in RPL

A node in RPL's secure modes is preinstalled with a symmetric key, which might be considered a serious security issue that leads to a single point of failure because replacement is not allowed in scenarios where malicious nodes obtain the key using a differential cryptanalysis attack. The node keys should be updated regularly to prevent such security concerns. The development of efficient key management mechanisms to generate, manage, and store keys are important research areas in RPL security [11].

2.3 RPL Security Threats

This section briefly discusses some RPL protocol-specific attacks.

Rank Attacks:

Depending on the type of OF implemented and the rank of neighbors, each node selects the preferred parent[1]. An attacker can modify rank values and thus significantly impact the routing DODAG [12]. A child node obtains all routing information via control messages without checking the integrity of its parent. It trusts the message it receives from its parent despite the parent being a malicious node.

Rank attack types include increased rank, decreased rank, and worst parent attacks [13]. In increased rank attack, the malicious node advertises a higher rank. This kind of attack aims to change the network routing topology and increase latency by forcing the victim node to choose another node as a parent. In decreased rank attack, the malicious node broadcasts a lower rank to other nodes, causing them to select a wrong preferred parent. It is difficult to identify “worst parent attack” since the attacker would choose the worst parent for itself but broadcast its real rank. Sahay et al. conclude that increasing the rank value results in an increase in energy consumption and the network traffic Whereas decreasing the rank value results in significant traffic and RPL DODAG disruptions. Further, the worst parent attack can increase the network delays [14].

Version Number Attacks:

In RPL, only the root node should update the DODAG's version number [1]. The global repair operation will be activated if a malicious node sends DIO messages with a higher version number, resulting in topology inconsistency and routing loops [15]. This results in the unnecessary reconstruction of the whole DODAG, thus increasing power consumption, network delay, and routing loops.

Local Repair Attacks:

When an RPL node loses contact with its preferred parent, it activates a local repair process [1]. In this attack, without facing any problems, a hacked node would broadcast local repair requests to its neighbors. This causes neighboring nodes to update their route that passes through the hacked node, leading to the unneeded reconstruction of the same network topology. The primary purpose of such an attack is to increase control overhead, power consumption, and reduce node resources [16].

DIS Attack:

DIS messages is delivered when new nodes attempt to join a DODAG in order to inform neighboring nodes and obtain the DODAG information [1]. In a DIS attack, an attacker might attack this capability by sending a large number of DIS messages to the neighbors. This attack can be implemented by multicasting DIS messages to multiple nodes simultaneously or delivering unicast DIS messages to a node [16]. This increases routing disruption, control overhead, and power consumption.

Neighbor Attacks:

The neighbor attack is also known as a route information replay. In this type of attack, the attacker collects information about control messages of network nodes and then multicasts them to neighbors, causing the nodes to update their routing tables with outdated data, thus resulting in an inconsistent topology. It causes an increase in network latency [16].

DAO Inconsistency Attacks:

RPL network uses flags to control critical topological mechanisms. The 'R' flag reflects a topological rank error. The 'O' flag reflects the direction of the packets. The 'F' flag

reflects that nodes cannot forward packets to the given destination [1]. An adversary may target this feature to launch attacks by adding the 'F' flag in the packets and forwarding them back to their parent. As a result, the parent nodes are forced to reject valid downward routes. Such attack increases node isolation and end-to-end latency.

ETX Manipulation Attacks:

The MRHOF uses the ETX metric in the RPL network to determine the best routing paths from nodes to a root in the DAG. Any parent node's ETX value must be less than a child node value. An attacker takes advantage of this rule by changing node ETX values to improve node position [17]. This helps malicious nodes to capture a large part of traffic for launching further attacks such as the Blackhole attack.

Replay Attacks:

This attack aims to store RPL control messages and send them later to network nodes [18]. Because hackers do not need to know the cryptography key used, this attack is possible even when RPL secure modes are enabled. Such an attack may result in expired or erroneous routing tables, poor routing service, and reduced packet delivery rates.

3. RELATED WORK

This chapter summarizes some earlier solutions for enhancing the RPL protocol. In [19], a security technique named VeRA is suggested. The technique provides security protocols against illegal version number and rank change cyberattacks. This technique employs hash chains to authenticate nodes whose rank or version number has changed. The researchers proposed a new approach [20], TRAIL, to address the decreased rank attack. They introduced a unique security method that employs a layered encryption chain to prevent an attacker from multicasting changed hash chains while guaranteeing rank integrity. However, the suggested security technique does not protect against a rank-replay attack, and both VeRA and TRAIL add memory usage to resource-constrained nodes. In [21], Glissa et al. proposed SRPL, a secure version of RPL protocol. The purpose of SRPL is to prevent an exploited node from unauthorized modifying control messages information that would cause rank manipulation to obtain a higher rank in the DODAG. Ghaleb et al. [22] suggested SecRPL mitigate the DAO falsification attacks. The suggested technique imposes restrictions on the amount of DAO packets transmitted to the destination node. The parent node keeps a counter for each child node in its sub-DODAG. When the number of DAOs from any child exceeds a specific limit, that child is labeled malicious, and then any further DAO is dropped by the parent.

In the typical version of RPL, the preinstalled mode assumes that a node seeking to join a protected network is already preconfigured with a shared key for all neighbors and the RPL root. This implies that if this shared key is compromised, the RPL DODAG's whole network is affected. As a result, we attempt to employ ECC for enhancing the secure authenticated key exchange in RPL for IoTs. Approaches [23][24][25] have been proposed that use a TPM for establishing nodes authentication and key exchange using RSA. The studies proposed adding

a chip to extend the capabilities of a standard wireless sensor node. As result, the cost and complexity of nodes will increase. TPM operates as a single point of failure. Manipulating such devices leads to an impact on the network performance and the level of security. Furthermore, there has been no sufficient detailed discussion of evaluation and simulation findings for validating the performance of the TPM based approach.

4. AUTHENTICATED KEY EXCHANGE PROTOCOL

In this chapter, the details of the design of the proposed security enhancement is presented.

4.1 Motivation

The IPv6 Routing Protocol for Low-Power and Lossy Networks RPL was introduced as the routing standard for the Internet of Things IoT. It supports a secure version of routing control messages while providing three security support modes known as Unsecured, Preinstalled, and Authenticated modes. The secure modes ensure the secrecy, integrity, and authenticity of messages. In the RPL's preinstalled mode, the network assumes that the key is preinstalled in the nodes and does not allow the key replacement even if an attacker obtains the network key via a differential cryptanalysis attack. To deal with this attack, the node key should be updated regularly. A key exchange mechanism is therefore required. Because of this, a trust establishment technique to validate the key source is also required where a new node needs to be authenticated with any existing node in the DODAG before the shared symmetric key is given to the new node.

Moreover, no existing approach investigated using ECC to secure communications and update the preinstalled key in RPL's secure modes. Compared with RSA, elliptic curve cryptography (ECC) has special advantages such as lower requirements of the size of the key and fewer parameters. So, ECC is more suitable to use for IoT devices that have limited resources, where an ECC with a 162-bit key size is as secure as RSA, which uses a 1024-bit key size.

4.2 Proposed Approach

The goal of our work is to enhance the security of RPL by proposing a scheme that secures key generation, exchange, and update. Before receiving a key, a node's trustworthiness must be established. In the secure DODAG, all nodes must have a symmetric key stored in the memory. This key is obtained from the root and will be updated periodically. If a new node wants to join the secure DODAG, the new node must first receive the proper key in order to encrypt or decrypt the RPL control messages. It must establish a trust relationship to get the key for secure communication with nodes in the secure DODAG. The new approach enables the key to be periodically replaced to avoid situations where an attacker may learn the key through differential cryptanalysis. In short, using a nonce to create a new key and ECC's asymmetric approach is used to pass this key used by the secure RPL network.

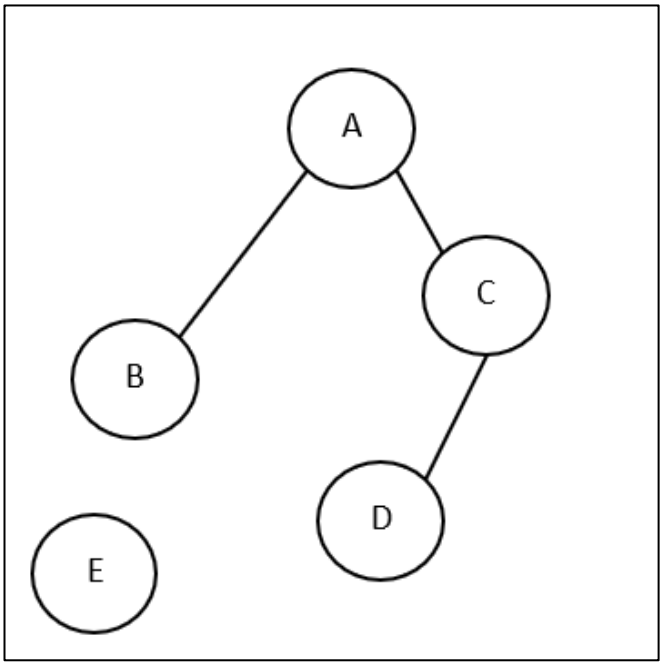

Figure 3: A new node wants to join the DODAG

The steps in trust establishment and key exchange are detailed below:

1. Pre-deployment phase

As a prerequisite for the proposed approach, each node must have a pre-shared secret K and a unique identification ID which are kept by the node itself. This K is not for encryption

between nodes, but only for authentication. Given the assumption that only the network nodes know K, each node can definitely be assured that anyone who uses K is indeed part of the network. So, it is necessary to keep K as a secret instead of sending it as plaintext during authentication. Otherwise, K can be used by an attacker who intercepts the transmission.

2. Trust establishment phase

Any new node must be authenticated before getting the symmetric key of a secure DODAG. Also, the new node has to first request the key from a neighboring node when it wants to join an existing DODAG. Consider the RPL DODAG in Figure 3, Node E sends a key request to Node B for obtaining the Key. Using this procedure shown in Figure 4, mutual trust of both nodes is established.

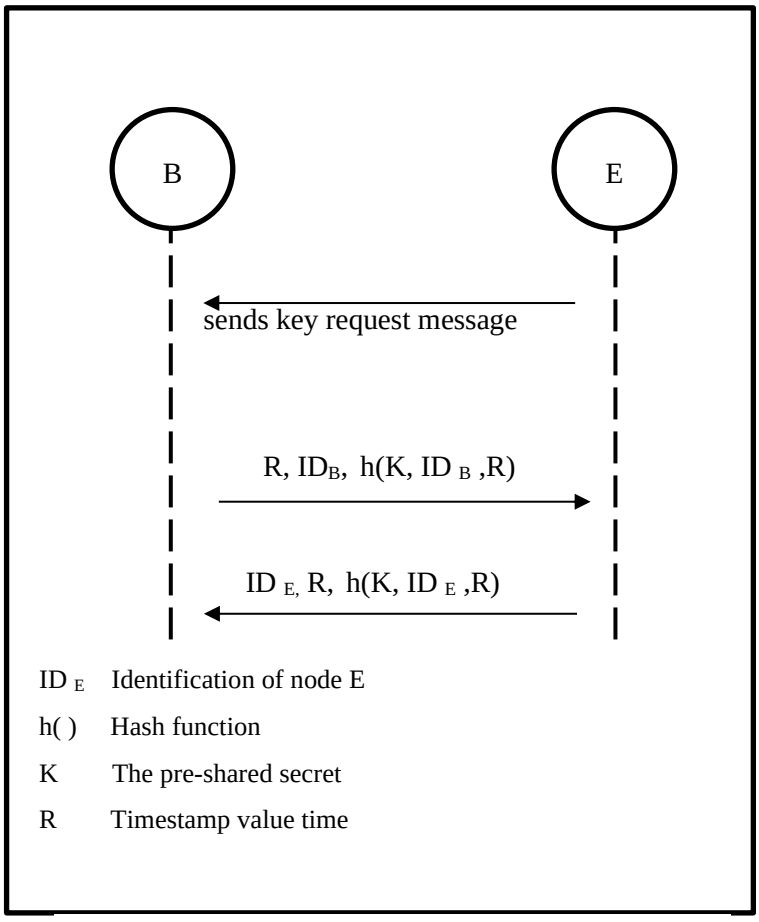

Figure 4: Authentication procedure

Node E computes $h(ID_E, K, R)$ and sends the hash value, ID_E, and R to Node B. Then, Node B uses the received ID_E and R and the stored k to compute $h(ID_E, K, R)$ and matches it with the received hash value. If both are the same, this means that the node is

authenticated. Otherwise, Node B would discard the request. Node B must also prove its trustworthiness before sending the symmetric key; a similar procedure is followed.

Algorithm1- Node E authentication procedure

BEGIN
Node E sends a key request message
Node E computes h(ID_E, k, R)
Node E sends the hash value, ID_E and R to node B.
Node B uses ID_E and R to compute h(ID_E, K, R)
Node B matches the computed hash value v1 with the received hash value v2
If (v1==v2), then
The node B is authenticated
END

Table 2 Node E authentication procedure

Algorithm 2 - Node B authentication procedure

BEGIN
Node B receives the key request message
Node B computes h(ID_B, k, R)
Node B sends the hash value, ID_B and R to node E.
Node E uses ID_B and R to compute h(ID_B, K, R)
Node E matches the computed hash value V1 with the received hash value V2
If (V1 == V2), then
The node E is authenticated
END

Table 3: Node B authentication procedure

3. Key exchange phase

After the mutual trust has been established, both nodes can then use the Elliptic Curve Diffie-Hellman ECDH operation to create a shared key for encrypting the symmetric key SK. Node B sends an encrypted message (AES-128) using the shared key, containing the symmetric key. Node E is now allowed to join the secure network and is capable of processing all secure RPL control messages.

Algorithm 3- Key exchange procedure between Node E and Node B
BEGIN Node B generates a random ECC key pair: BPrivKey and BPubKey. Node E generates a random ECC key pair: EPrivKey and EPubKey. Nodes B and E exchange their public keys BPubKey and EPubKey. Nodes B and E create a shared key by using ECDH Node B encrypts the SK by using the sheared key and AES-128 algorithm Node B sends the encrypted SK to Node E END

Table 4: Key exchange procedure

4. Key update procedure phase

The proposed solution is considered a dynamic key exchange protocol. The symmetric key needs to be periodically updated. The process is started by the root node broadcasting a key update message to its child nodes. The children respond to the parent to prove their trustworthiness following the same Authentication Procedure. After the mutual trust is established between the root and its child node, they use the Elliptic Curve Diffie-Hellman ECDH operation to create a shared key for encrypting the new symmetric key. Then, the root responds with the encrypted message that includes the new symmetric key. Then the child nodes repeat this process for any children they have until all nodes have obtained the new key.

Algorithm 3- Key update procedure
BEGIN The root node sends the key update message to its child nodes The children and their preferred parent start a mutual trust procedure The preferred parent sends the encrypted new SK to its children The child nodes repeats this process for all their child nodes END

Table 5: Key update procedure

A summary of SK update and exchange operation is shown in Figure 5. Note that the root starts this operation periodically, for example, hourly, by generating a random number (a nonce) as SK and encrypts it using the shared key before transmitting it to a child node. The child node decrypts the message from the root with the same shared key and obtains the new SK, then sends this SK to its children to use for secure controls messages.

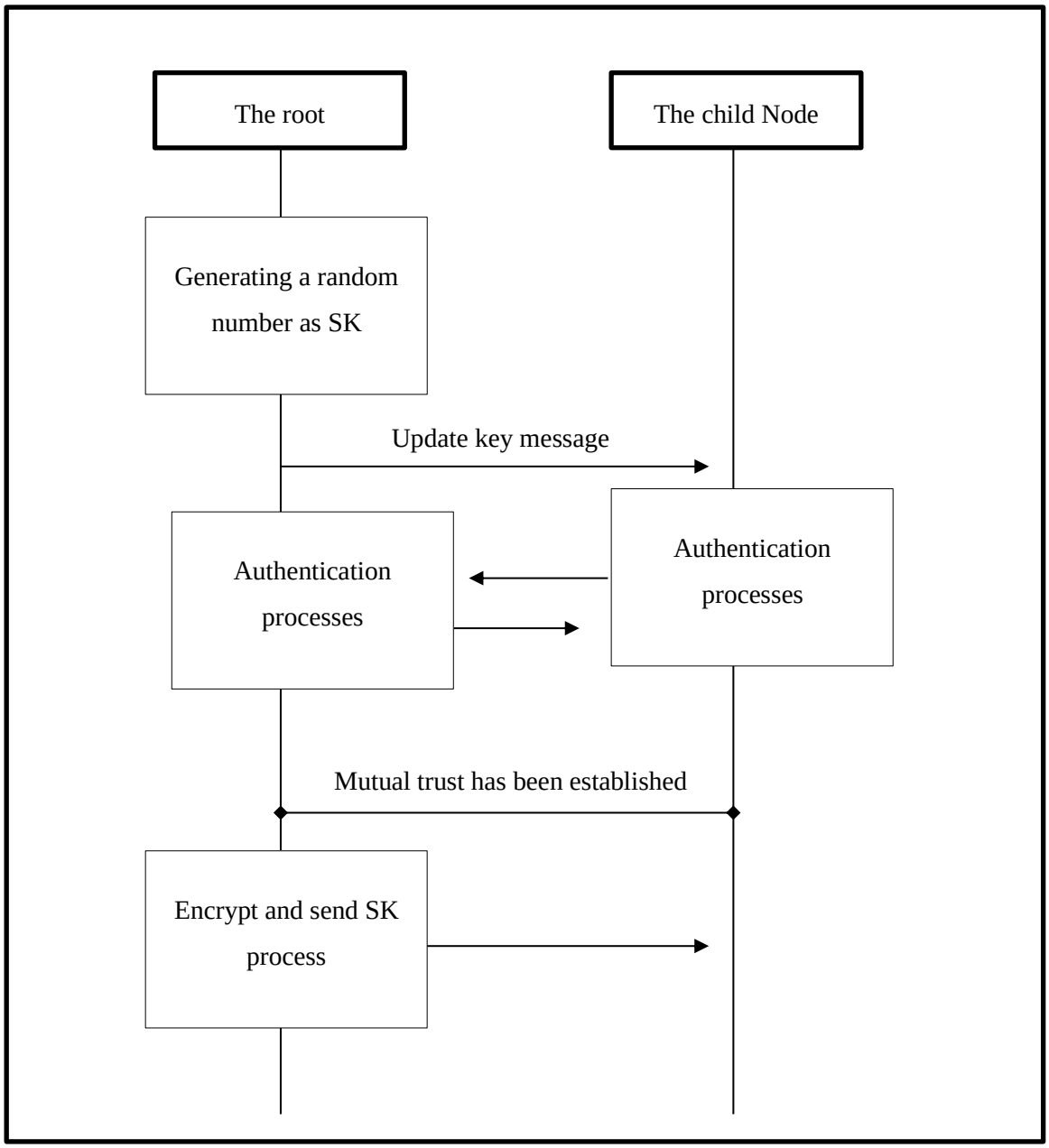

Figure 5: Summary of SK update and exchange operation

5. SECURITY

In this chapter, we discuss the security goals which we considered and analyze the security of the proposed protocol.

5.1 Security Goals

IoT network security solutions must meet one or more security requirements.

5.1.1 Authentication

It is possible that an attacker not only changes the packets but also injects faked packets into the network. The communication nodes must be able to validate the authenticity of the identity of the data source nodes

5.1.2 Confidentiality

Confidentiality is a critical aspect of IoT communication. It relates to encrypting data and keeping it secret.

5.1.3 Integrity

Integrity guarantees that packets sent from source to destination have not been tampered with by an unauthorized third party. If any change has occurred, it must be discovered.

5.1.4 Availability

Even if the node is attacked, the network should resist and preserve the availability of its resources and services.

5.1.5 Mutual authentication

Two nodes authenticate each other at the same time. Only authenticating the source node to the destination node is insufficient thus, the source must also be sure about the identity and authority of the destination.

5.2 Security Analysis

We investigated several security threats that the proposed scheme may face and analyzed the defense against these threats.

5.2.1 The new approach can resist an eavesdropping attack

The eavesdropping attack is on confidentiality which is one of the main objectives of the proposed solution. It is classified as high risk and is prevented by strong end-to-end encryption, which provides secure end-to-end communication. This ensures that unauthorized parties do not have access to data. Our approach uses the Elliptic Curve Diffie-Hellman ECDH and AES-128 techniques to create a shared key and encrypt the symmetric key SK. Therefore, without knowing the shared key, an attacker cannot figure out or steal the SK and the data.

5.2.2 The new approach can resist an impersonation attack

To impersonate a legitimate node to pass the verification test performed by another legitimate node, an adversary node must send a valid hash value h(ID, K, R) to the legitimate node. However, the authentication security assumption makes it impossible for the adversary node to get the pre-shared secret K to compute the valid authentication hash value to deceive the legitimate node successfully.

5.2.3 The new approach can resist a replay attack

In our proposed approach, it can be obviously seen that the timestamp is embedded in the mutual authentication phase. A replay attack can be prevented using timestamp R. The real message would have the current time, whereas the replay attack message would contain the old time.

5.2.4 The new approach can resist a man-in-the-middle attack (MITM)

A hacker can use the MITM attacks to capture any stream of data while remaining undetected. The suggested approach addresses this problem by enabling mutual authentication that is impossible to manipulate. In the authentication procedure, it is difficult to reverse the hash function, even with a partially known inputs ID and R. The K is impossible to recover, which will prevent the MITM attack.

5.2.5 Forward secrecy and backward secrecy

Our protocol can enhance the security levels of RPL protocol by providing backward and forward secrecy. Backward secrecy ensures that if the current symmetric key SK is compromised, an attacker will not be able to learn previously recorded communications encrypted using the previous SK. Forward secrecy ensures that if the SK is compromised, an attacker will not be able to discover future traffic encrypted using the new SK. Using a random number (a nonce) as SK can improve the protocol's security. It is extremely difficult for an adversary who has gotten the current SK to determine the keys created in the past or in the future.

In this section, we provide a security analysis of the proposed protocol as well as demonstrates that it supports several security features and is secure against a variety of common attacks.

6. IMPLEMENTATION AND EVALUATION

6.1 Implementation

We provide a C-based implementation that is specific to ContikiOS version 3. The proposed protocol is implemented as an extension of the existing implementation of rpl-udp [26]. We reuse most of the code of rpl-udp and other existing code, such as ECDH implementation [27] and SHA-256 [28], and we provide our versions of them that are compatible with the rest of ContikiOS. This section describes the implementation platform, the runtime environment, and the implementation choices we made.

Feature	Description
Flash memory/ROM	92 KB
RAM	8 KB
Operating voltage	3V
Micro-controller	MSP430F2617
Processor / CPU	16-bit RISC CPU @16MHz clock speed

Table 6: Z1 mote specifications

6.1.1 The used functions

To compute the shared key, we reused the existing implementation [27], nano-ecc, for ECDH. nano-ecc is a tiny ECDH implementation based on kmackay's micro-ecc [29]. It has a short code size of 6KB, and it supports four standard curves: secp128r1, secp192r1, secp256r1, and secp384r1. It also has no dynamic memory allocation. We used ecc.h and ecc.c in our implementation. We use #include "ecc.h" to use the functions. The ecc_make_key() function to get public/private key pair and the ecdh_shared_secret() function computes a ECC shared key given the node's private key and the other node's public key.

We employed the implementation of sha256.c [28], which implements the SHA-256 hashing algorithm. We used the hash_256() function, which outputs a 32-byte digest, to get the node's hash value before sending the value to some other node. RPL uses AES-CCM with 128 bits key size to encrypt the control messages. To preserve the memory usage, we used the same encryption algorithm to encrypt the new SK by using the generated ECC shared key. The set_key() function is used to set the given key. The aead() function is used to encrypt and decrypt the new SK in both nodes.

Simulator	Cooja
Mote type	Z1
Network	IPv6/RPL
MAC layer	IEEE 802.15.4
Duty cycling	ContikiMAC
Radio access	CSMA
Max number of nodes	40
Host system	i7-8550U CPU @ 1.80GHz, 16.0 GB Memory

Table 7: Emulation configuration

6.2 Experimental Environment

6.2.1 Contiki OS

Contiki OS is an open-source operating system for developing resource-constrained devices [30]. It specifically targets small IoT devices with limited memory, power, bandwidth, and processing power. Contiki supports a full IP network stack, with standard protocols such as UDP and TCP, along with the new low-power wireless standards such as 6LoWPAN [31], RPL, and CoAP [32]. 6LoWPAN stands for IPv6 over low-power wireless personal area networks. It uses compression technology to provide low-data-rate wireless, requiring devices with limited resources. RPL is a distance-vector IPv6 protocol for Low Power Lossy Networks (LLNs) that finds the best optimal path in a complex network of

devices with varying capabilities. Cooja [33] was used for simulations. it is a flexible Java-based cross-layer simulator included with the ContikiOS.

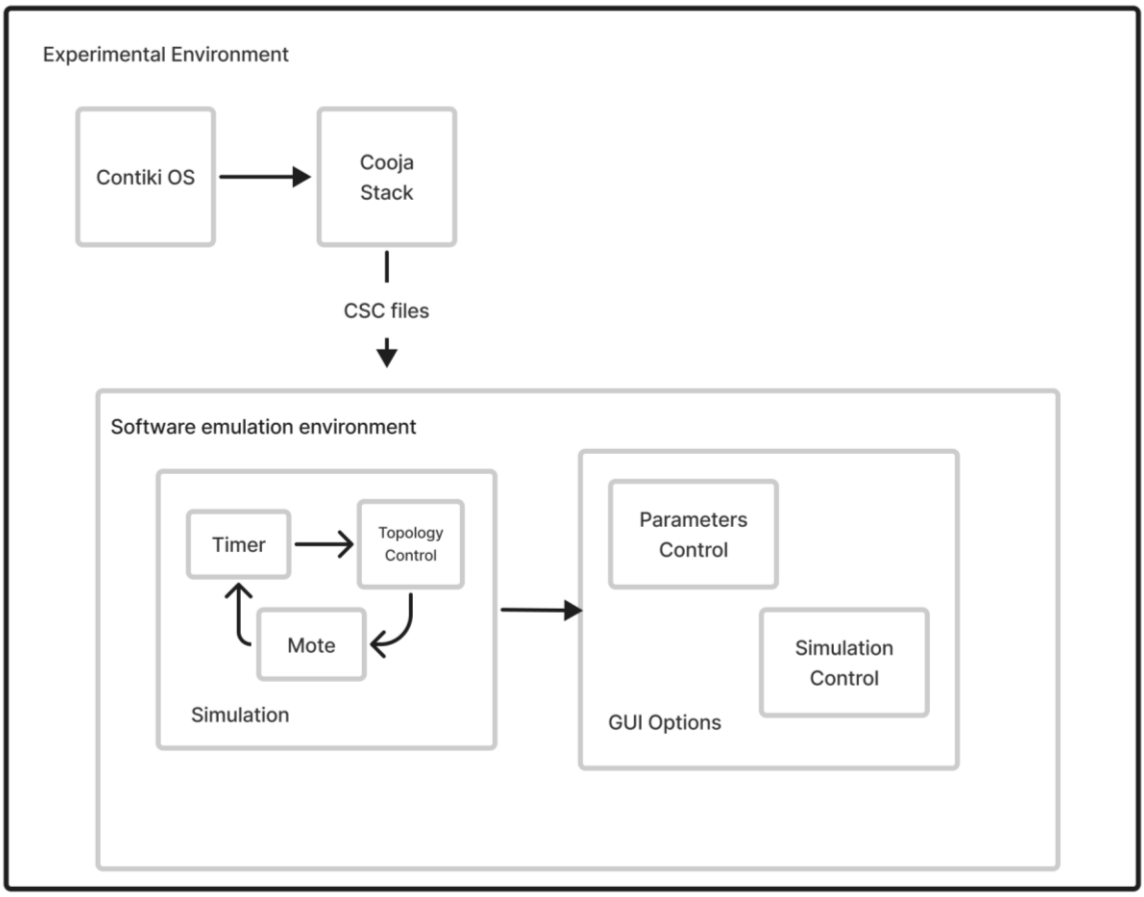

Figure 6: Simulation system architecture

6.2.2 Software emulation environment

It is quite challenging to create and debug software for large wireless networks. The Contiki network simulator, Cooja, makes this much more accessible by offering a simulation environment that allows developers to observe their applications in large-scale networks by using fully emulated hardware devices.

The Cooja simulator was used to perform software emulation. We employed the ContikiOS on a virtual machine to execute an emulation. We obtained the ContikiOS from a Github repository [34]. Mote Z1 is a popular mote that receives much support from ContikiOS. As a result, we chose Z1 mote as the hardware platform for the proposed protocol. Z1 mote features include a MSP430F2617 MCU, 8 kB RAM, 92 kB ROM, a cc2420 802.15.4

radio transceiver [35]. The specifications of the Z1 mote are shown in Table 6. We utilized Contiki's default configurations. Table 7 shows some of the configurations in the emulation setup. The proposed protocol is implemented using the C programming language. If assembly code is used, performance can be enhanced even further.

6.3 Performance Analysis

This section evaluates the proposed protocol in terms of communication and computation costs, power consumption, memory usage and finally concludes with performance results. In all our experiments, we calculated ROM consumption in byte, RAM consumption in byte, execution time in millisecond (ms), and power consumption in megawatt **(**mw).

6.3.1 Computational cost

The computational cost is defined as the execution time per time step. In this section, we analyzed the computation overhead in the proposed protocol. We estimated the simulation execution time to estimate the time it takes the proposed protocol to execute on real-time hardware. The computation times are obtained based on the Z1 platform. The computation times are calculated through the following steps:

1- Add the following command line before each operation in our protocol to get the uptime in clock ticks by default 128 ticks per second.

```
start_time = clock_time();
```

2- Add the following operation after each operation.

```
Printf(" Completion time: %lu / %lu \n", (unsigned long)clock_time()- start_time,
CLOCK_SECOND );
```

The different operations of cryptography we used in the protocol and their computation times are listed in Table 8. Note this computation cost will increase by the same cost with each new key update.

Operation	Notation	Computation time [ms]
Node authentication	SHA-265	7.8
Shared secret key establishment	ECDH	6530
AES-128 encryption	E	7
AES-128 decryption	D	7

Table 8: Execution time of the proposed protocol operation on Z1

The secure authenticated key exchange operations on a new node side:

- Establish authentication with a secure DODAG by computing the node's hash function $h(ID_E, k, R)$ using SHA 265 algorithm.
- Verify that the received $h(ID_B, k, R)$ equals the neighbor node's hash function.
- Establish a shared secret key (ECDH).
- Decrypt the received symmetric key SK using the AES-128 algorithm.

Thus, the time required for the new node to get the SK is $h(ID_E, k, R) + h(ID_B, k, R) + ECDH + D = 7.8+7.8 + 653 + 7 = 675.6$ ms

On the secure DODAG 's node side:

- Establish authentication with the new node by computing the node's hash function h(IDB, k, R) using SHA 265 algorithm
- Verify the new node's hash function $h(ID_E, k, R)$.
- Shared secret key establishment (ECDH).
- Encrypt the secure DODAG 's SK using the AES-128 algorithm.

Thus, the time required to send the SK is $h(ID_E, k, R) + h(ID_B, k, R) + ECDH + E = 7.8+ 7.8 + 653 + 7 = 675.6$ ms

6.3.2 Communication cost

To calculate the communication cost of my proposed scheme, we defined the following assumptions:

- SHA256 digest size is 32 bytes.

- Node ID size is 1 byte.
- Timestamp is 1 byte.
- Symmetric key SK size is 16 bytes. (128 bits)
- ECC Public key size is 32 bytes.

Phases	# Messages	# Bytes
Mutual authentication	3	32+1+1= 34
Shared secret key establishment	2	32+32= 64
Secure SK transmission	1	16
3	6	114

Table 9: Communication cost of key exchange

The communication of the protocol is divided into three phases: Mutual authentication, Shared secret key establishment, and Secure SK transmission. In the first phase three messages are exchanged. The first message is only to request the symmetric key SK. The other two messages exchange the hash function value. The second phase contains two messages to exchange the node's public key. The last phase contains one message to send SK. With each key update request, these actions will be repeated. Table 9 summarizes the communication costs of the protocol. Each row in this table corresponds to one phase in the proposed protocol, and the last row shows the total number of communication phases, messages, and transmitted bytes. The total communication cost for one key distribution is 114 bytes. This cost increases with each new key update.

6.3.3 Power consumption

This section analyzes the proposed protocol regarding the power consumed if Z1 sensor motes are used. The average value of each node's power consumption in the network is referred to as power consumption. The Powertrace tool is used to evaluate power consumption [36]. The Z1 datasheet represents the current power consumption values [37]. A hardware timer is used to measure power states. RTIMER SECOND specifies a clock

frequency which is 32,768 ticks. This statistic takes into account both communication costs for message sending and receiving and computation costs for cryptographic and hash functions. To collect results, simulations are run for 1 hour. The key refresh was every 30 min. The sensor node measured the power consumption value every 10 minutes.

The following steps are used to compute the power:

1. Check the number of ticks per second by the following command.

   printf(" Ticks per second %u \n," RTIMER_SECOND);

2. To add the powertrace tool to the project, add the following header to the Makefile

   APP += powertrace

3. Add the following header to the source file. #include "powertrace.h"

4. print power consumption every 600 seconds by the following command.

   powertrace_start(CLOCK_SECOND * 600)

The energy consumption is calculated by the formula:

$$\text{Power(mW)} = \frac{Energest\_Value \times Current \times Voltage}{RTIMER\_SECOND \times Runtime}$$

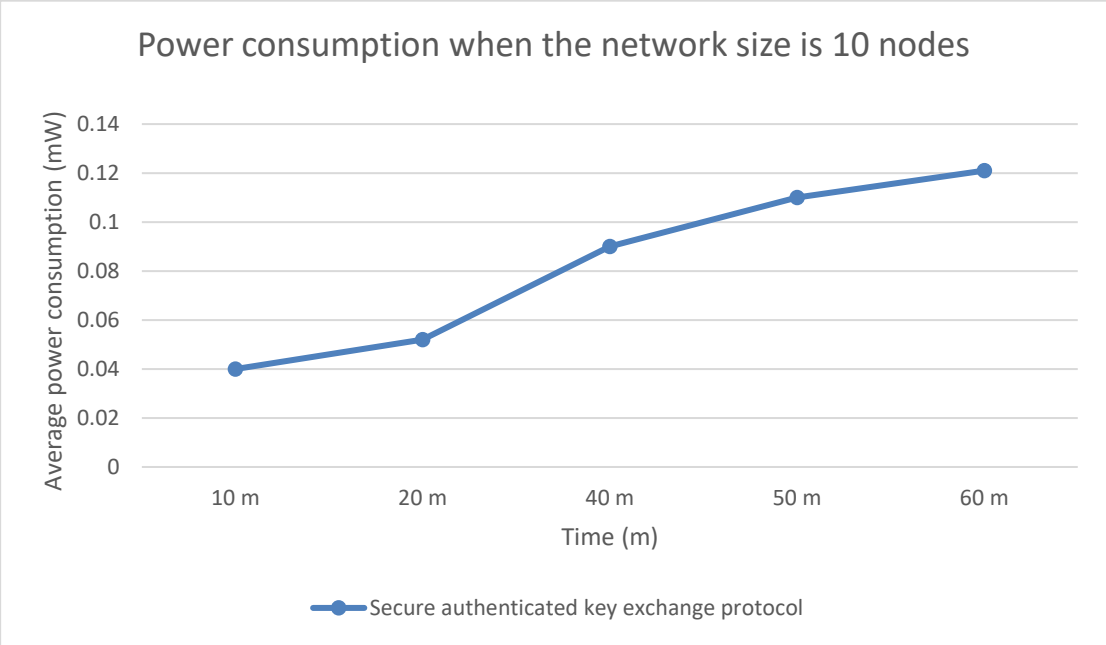

Figure 7: Power consumption when the network size is 10 nodes

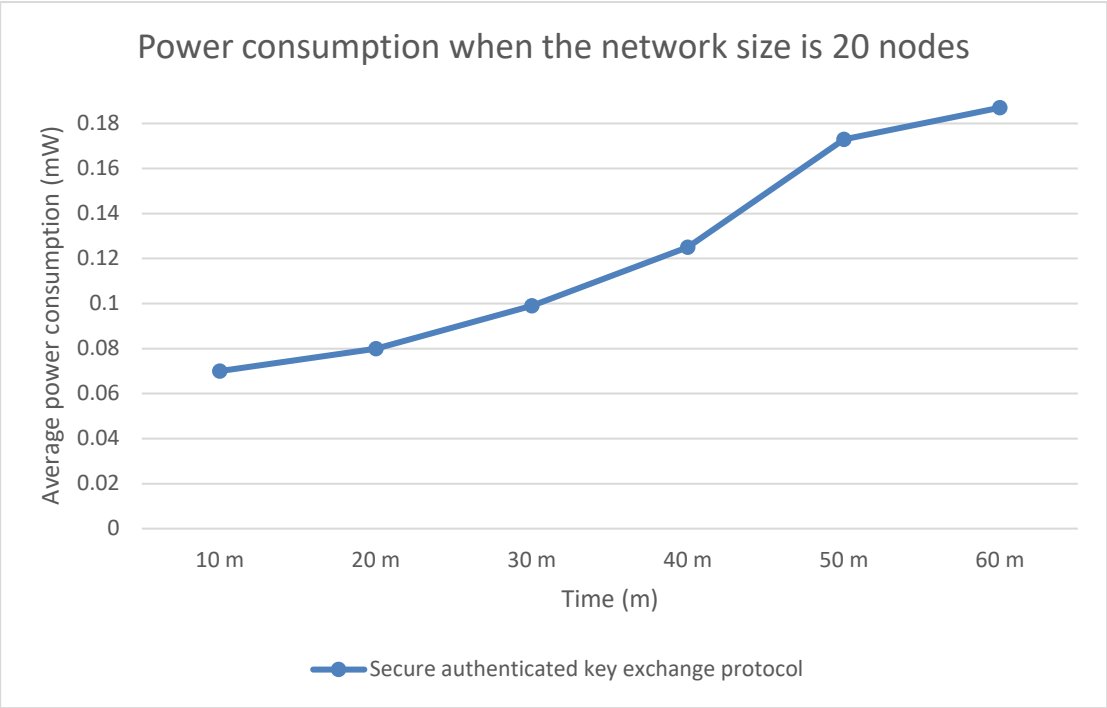

Figure 8: Power consumption when the network size is 20 nodes

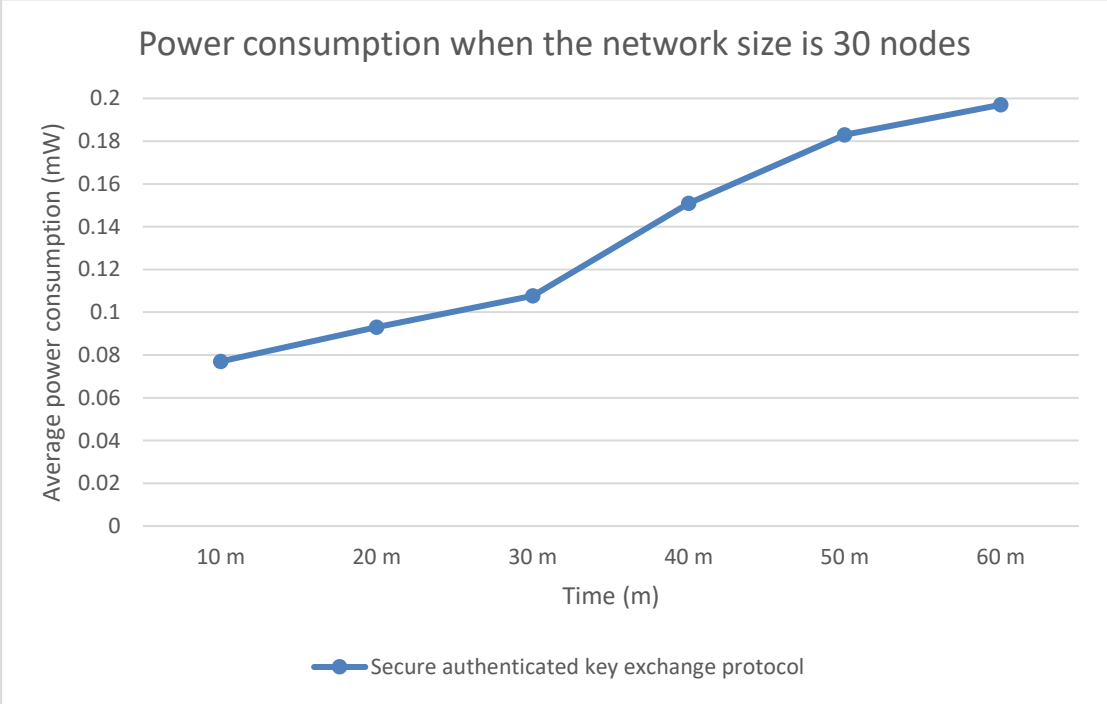

Figure 9: Power consumption when the network size is 30 nodes

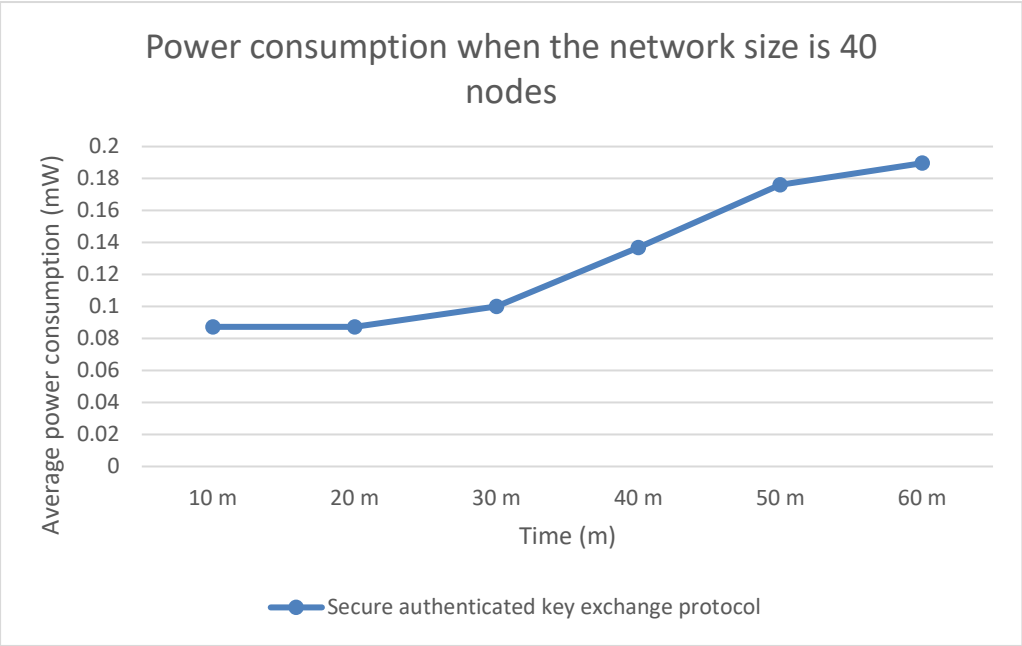

Figure 10: Power consumption when the network size is 40 nodes

Figures 7, 8, 9, and 10 show the average power consumption. The proposed protocol caused a significant rise in the overall power consumption during the experiment. The main reasons for the increase were the computation overhead of the cryptographic and hash operations and the larger size of the transmitted packets. These processes are required for each key refresh phase. The key distribution in this simulation is every 30 minutes, which occurred twice during this experiment time. As predicted, the conclusions drawn with different network sizes demonstrate that as the network size grows, the power consumption increases due to an increase in the number of packets sent between network nodes and the time each node spends processing packets and receiving the key.

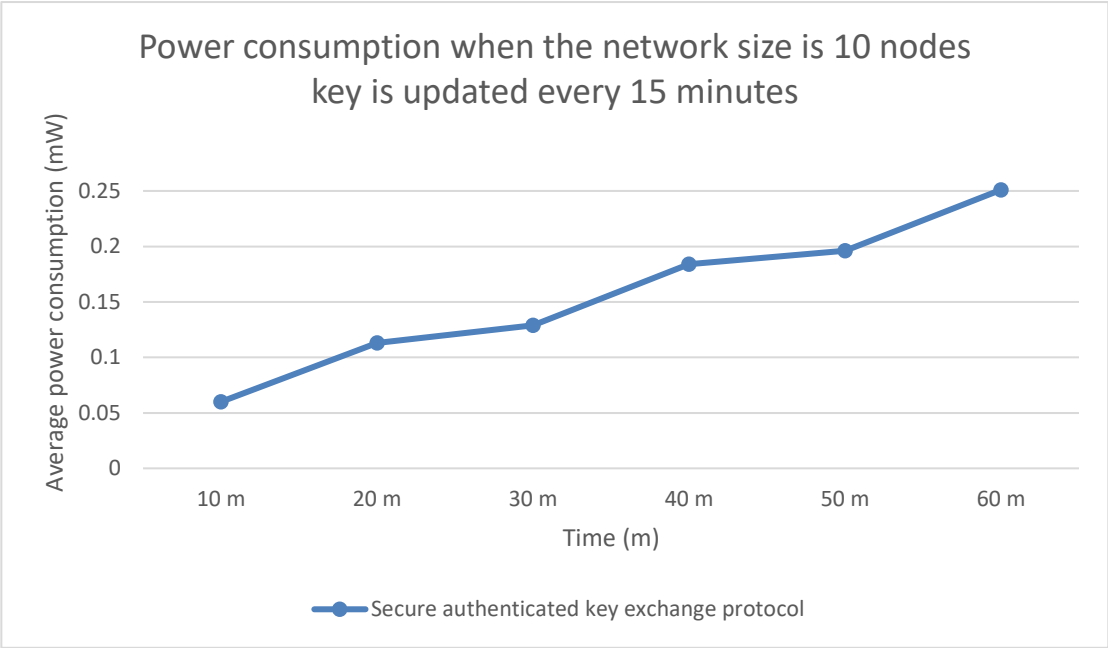

Figure 11: Power consumption when the network size is 10 nodes and key is updated every 15 minutes

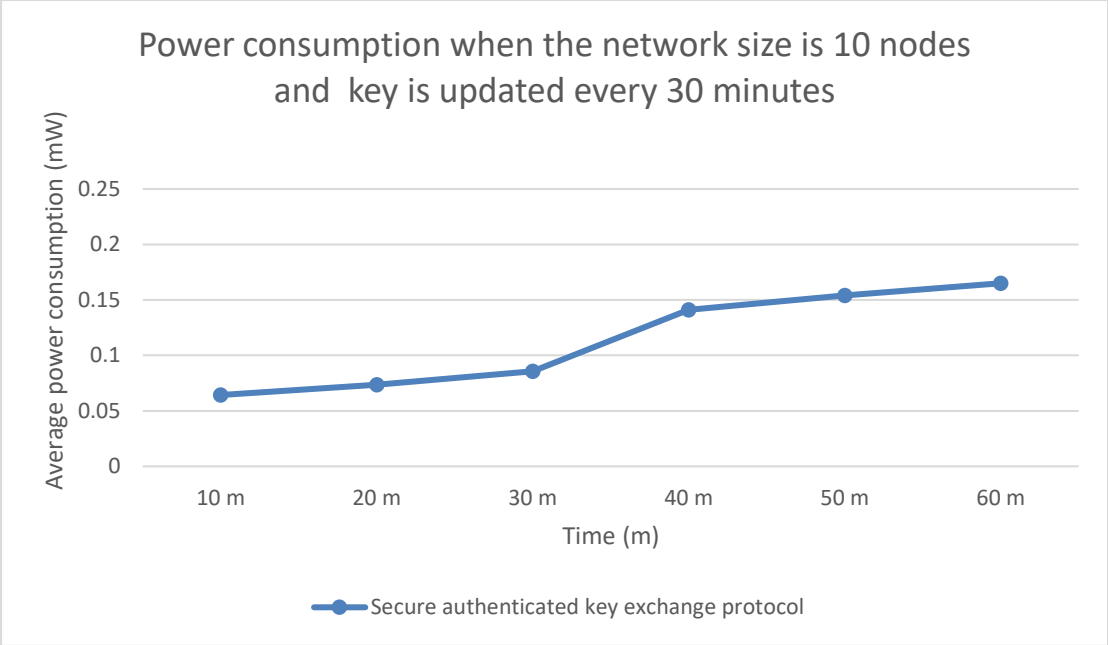

Figure 12: Power consumption when the network size is 10 nodes and key is updated every 30 minutes

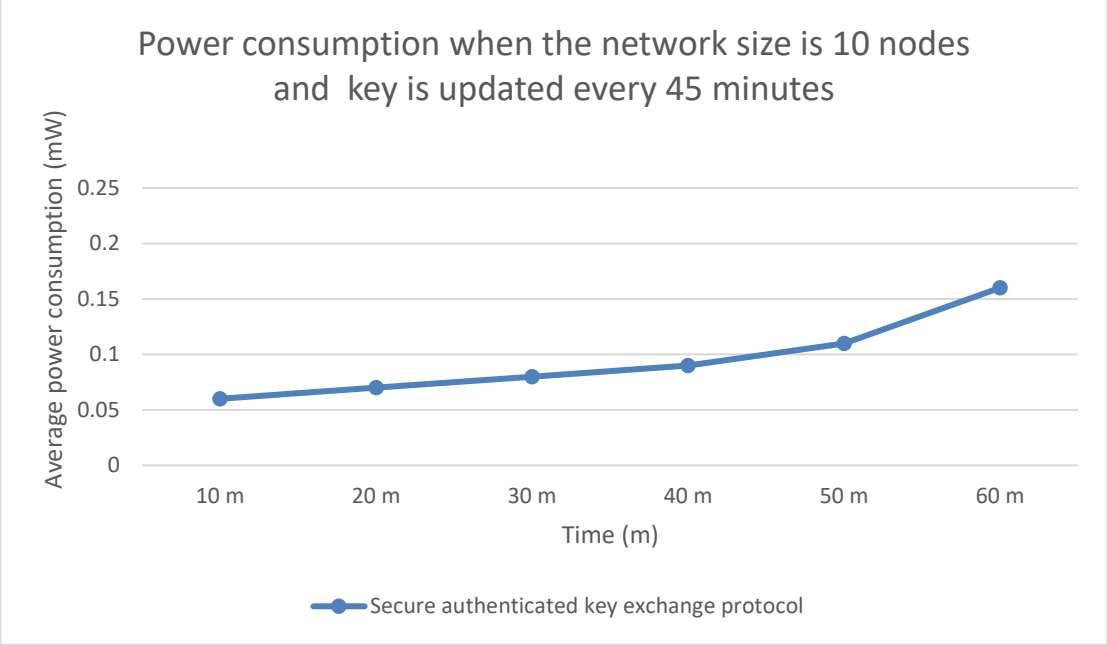

Figure 13: Power consumption when the network size is 10 nodes and key is updated every 45 minutes

Figures 11, 12, and 13 show the average power consumption vs. different key update period lengths. The proposed protocol caused a significant rise in the overall power consumption during the experiment. The main reasons for the increase were the computation overhead of the cryptographic and hash operations and the larger size of the transmitted packets. These processes are required for each key refresh phase. The key distribution in this simulation is every 15, 30, and 45 minutes. As predicted, the conclusions drawn with different lengths of key update periods demonstrate that as the update period decreases, the power consumption increases due to an increase in the number of packets sent between network nodes and the time each node spends processing packets and receiving.

```
sarah@sarah-VirtualBox:~$ msp430-size  /home/sarah/contiki/examples/ipv6/rpl-udp
/NodeE.z1
   text    data     bss     dec     hex filename
  54119     686    6196   61001    ee49 /home/sarah/contiki/examples/ipv6/rpl-ud
p/NodeE.z1
sarah@sarah-VirtualBox:~$
```

Figure 14: Memory consumption

6.3.4 Memory consumption

We used the "msp430-size" command to get the memory (RAM, ROM) consumption. Figure 14 illustrates the execution of the msp430-size command. In this case, the text column refers to the amount of ROM consumed by the proposed protocol in bytes. The RAM consumption is shown in the data and bss columns. Our protocol consumes 54.119 KB of ROM and 6.88 KB of RAM.

6.3.5 Initial key distribution time

We evaluated the performance of the first key exchange under different numbers of Z1 motes. The performance result of the proposed protocol is shown in Table 10 and Figure15. Considering 10 motes as an example, the first SK transmission may be completed in 133943ms. The total SK transmission time for 10 motes is 295345ms. The Packet Loss Rate (PLR) represents the ratio of lost packets to the total number of packets. Packet losses happen when one or more data packets traveling across the DODAG do not reach their destinations. With the increasing number of motes, the PLR value will be larger. This increase in lost packets results from increasing packet collision as the number of nodes increases.

Moreover, all Z1 Motes were booted synchronously, and they sent request messages to their preferred parents after the parent got the SK. Some packets were lost because many packets blocked the network. The network topology also influences the performance of the protocol, which is discussed in detail in the following section. An imbalanced load topology might cause a node to become temporarily inaccessible, increasing latency and significantly slowing down the mutual authentication and SK transmission procedures.

# Motes	First SK transmission[ms]	Total SK transmission[ms]	Packet loss rate
10	133943	295345	1.66%
20	136868	498168	3.3
30	138365	689386	6.64
40	138432	886064	12.28

Table 10: Initial SK distribution time

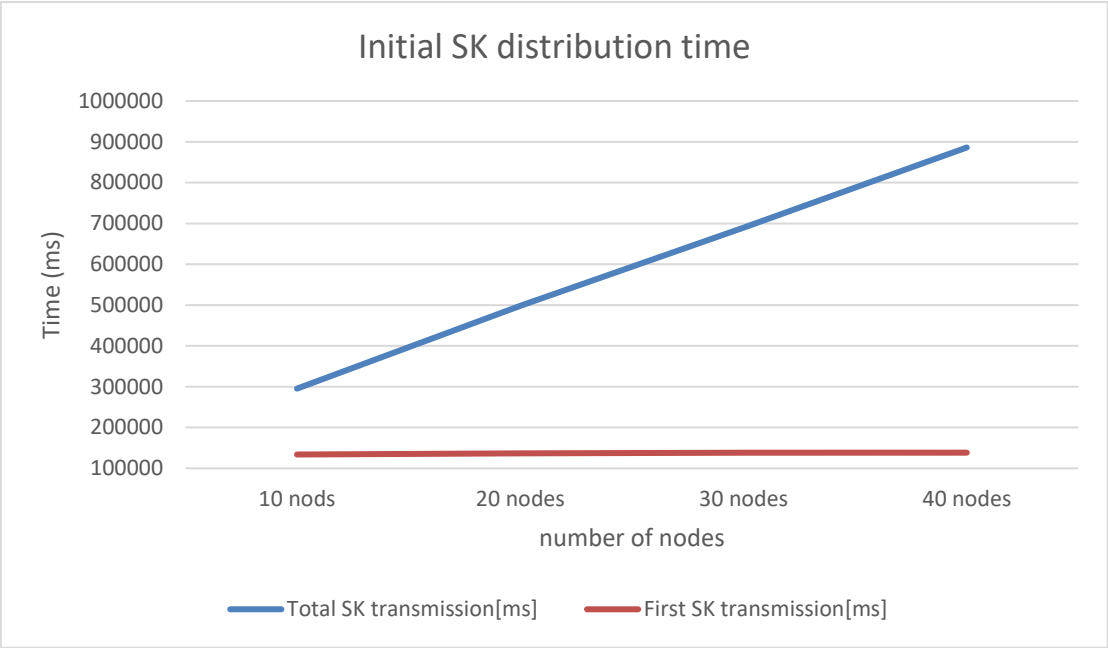

Figure 15: Initial SK distribution time

6.3.5.1 Topology issue

RPL was created with use in a low traffic network. However, because of the overhead communication of the new protocol, which includes sharing and often updating the key, heavy traffic is generated. In this scenario, RPL would be unable to handle the traffic well [38]. As a result, the network may face significant congestion, leading to packet loss and delay. When a child node is unable to change its chosen parent and continues to send data traffic even when the parent node is severely congested, the issue gets more serious [38]. This is due to the OFs, which enable each node to choose the parent with the shortest path to the root and the most reliable link by using the ETX metric, despite congestion. Consider the network topology given in Figure 16 to better understand the issue. Assume that the ranks of nodes 1,2, 3, and 4 are equal, and node 3 is overloaded in comparison to nodes 1, 2 and 4. In this scenario, the network's resources are not appropriately balanced, which may lead to future congestion. If this situation occurs in the proposed protocol, each node starts to send the key request message to the preferred parent, as a result of the unbalanced load, node 3 becomes overloaded and loses a considerable amount of packets. To address this problem,

we need to construct network topology with balance load and check the total number of children linked to the preferred parent. We will work to fix this problem in the future.

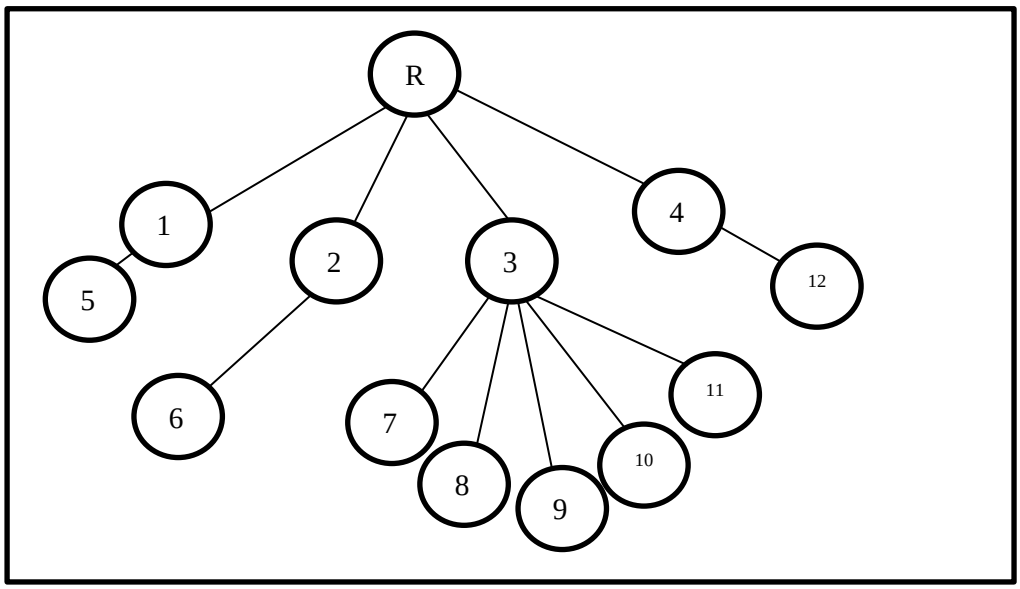

Figure 16: RPL topology with an unbalanced load

6.4 Comparative Analysis

To evaluate the overhead presented by the proposed protocol, simulations were used to accomplish a performance evaluation. Two RPL configurations are investigated to achieve this objective: RPL with pre-loaded keys, which is implemented as the existing implementation of rpl-udp [26], and RPL with secure authenticated key exchange protocol. Cooja has been used to execute simulations. It is set up to act as a Z1 sensor mote. A wireless channel is simulated using the Unit Disk Graph Medium model (UDG). A typical grid topology with an increasing number of nodes is used to analyze the protocol's performance in networks of various sizes. The default ContikiOS parameters are used to configure all RPL settings.

The impact of the suggested protocol on the network operation is compared to the standard RPL network through the following metrics: Network formation time, which is the time interval between the start of the simulation and the last node joining the DODAG and sending a hello message, and power consumption, which is the overall value of the power consumption of nodes in the network. To estimate the power consumption of a node. On the node, we run the powertrace tool in the background. This statistic takes into account both

communication costs for message sending and receiving and computation costs for cryptographic and hash functions. To collect results, simulations are run for 1 hour. The key refresh was every 30 minutes. The sensor node measured the power consumption value every 10 minutes.

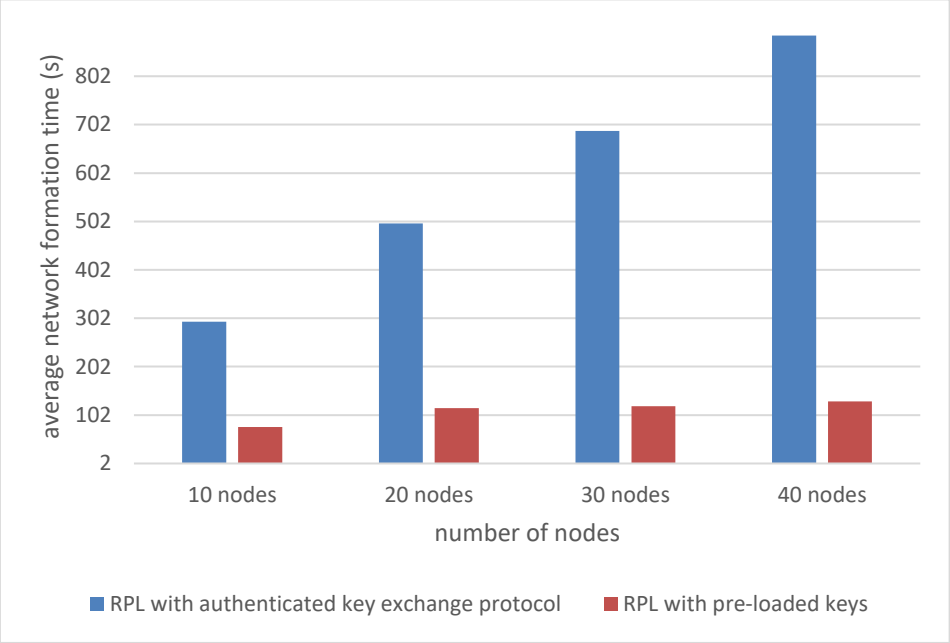

Figure 17: The network formation time

The overall network formation time is shown in Figure 17. As expected, the time it takes to establish a network grows as the size of the network expands. Regardless of the network size, the overhead introduced by the proposed protocol increases the network formation time by around 400%. This is explained by the fact that each node must exchange additional messages before joining the secure DODAG and receiving the network key. The network formation time with the authenticated key exchange can probably be reduced by employing algorithms to avoid collisions between packets of joining nodes. In future work, we intend to look at this possibility.

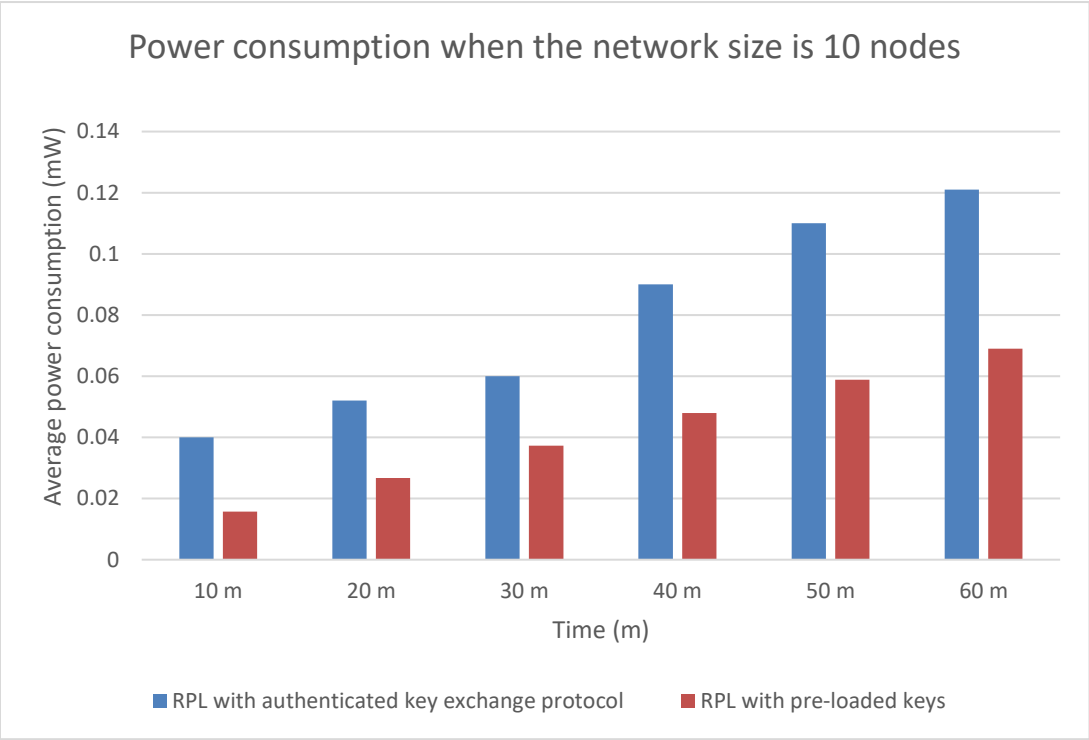

Figure 18: Power consumption when the network size is 10 nodes

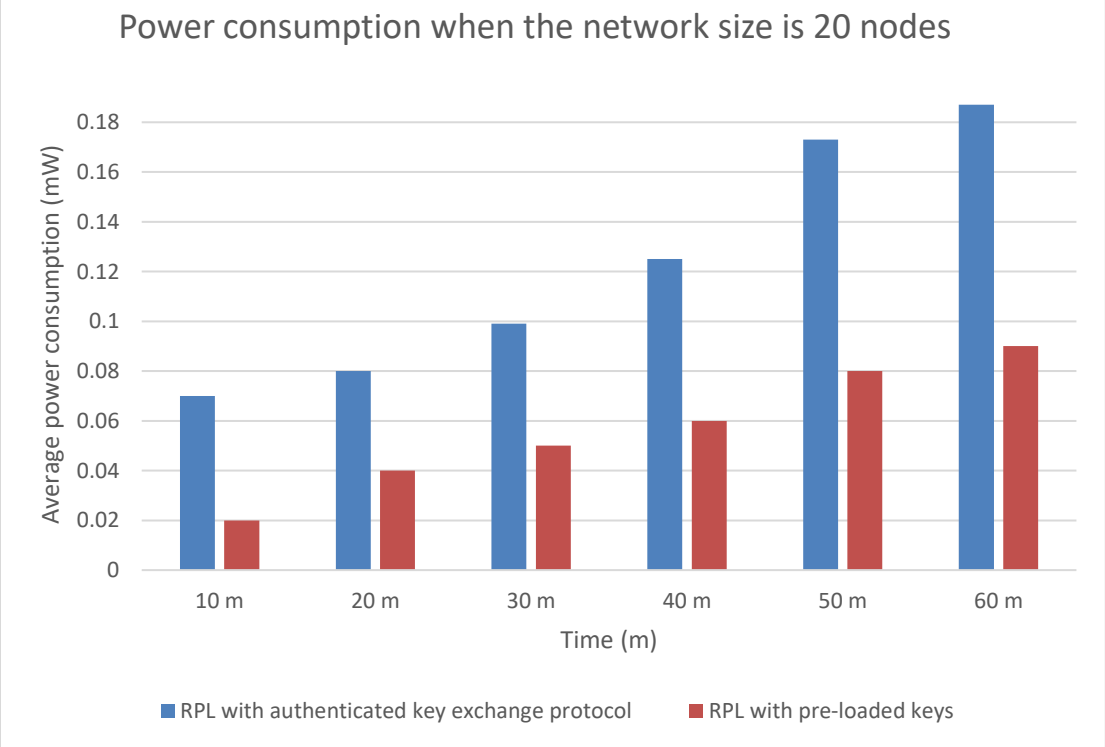

Figure 19: Power consumption when the network size is 20 nodes

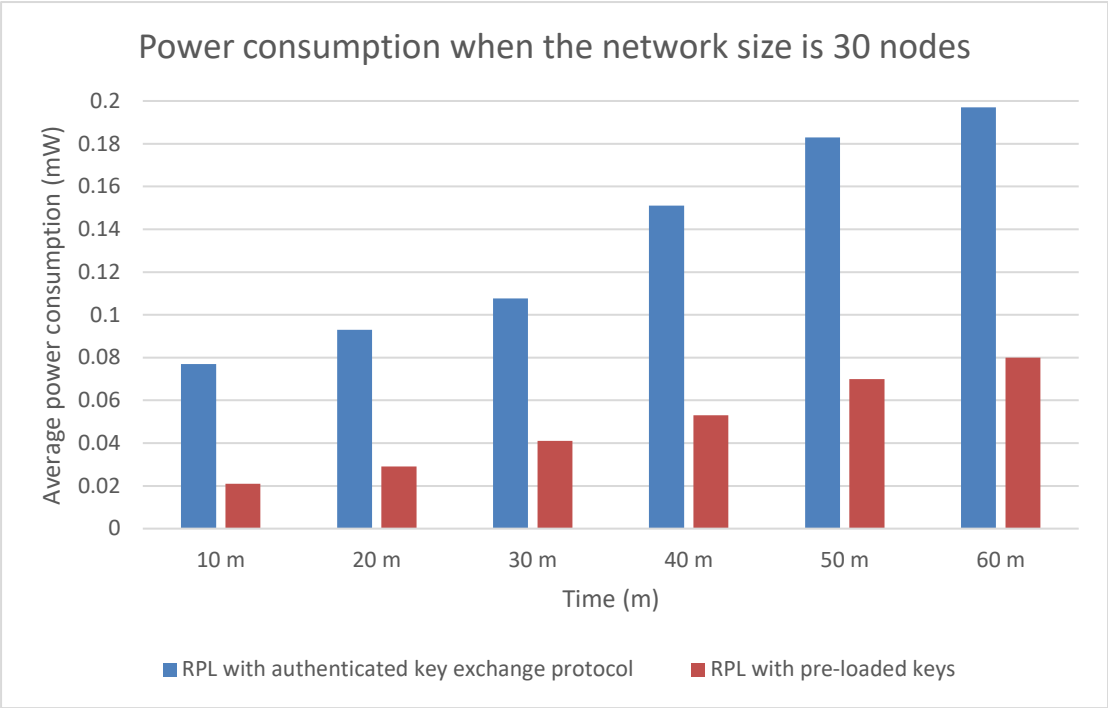

Figure 20: Power consumption when the network size is 30 nodes

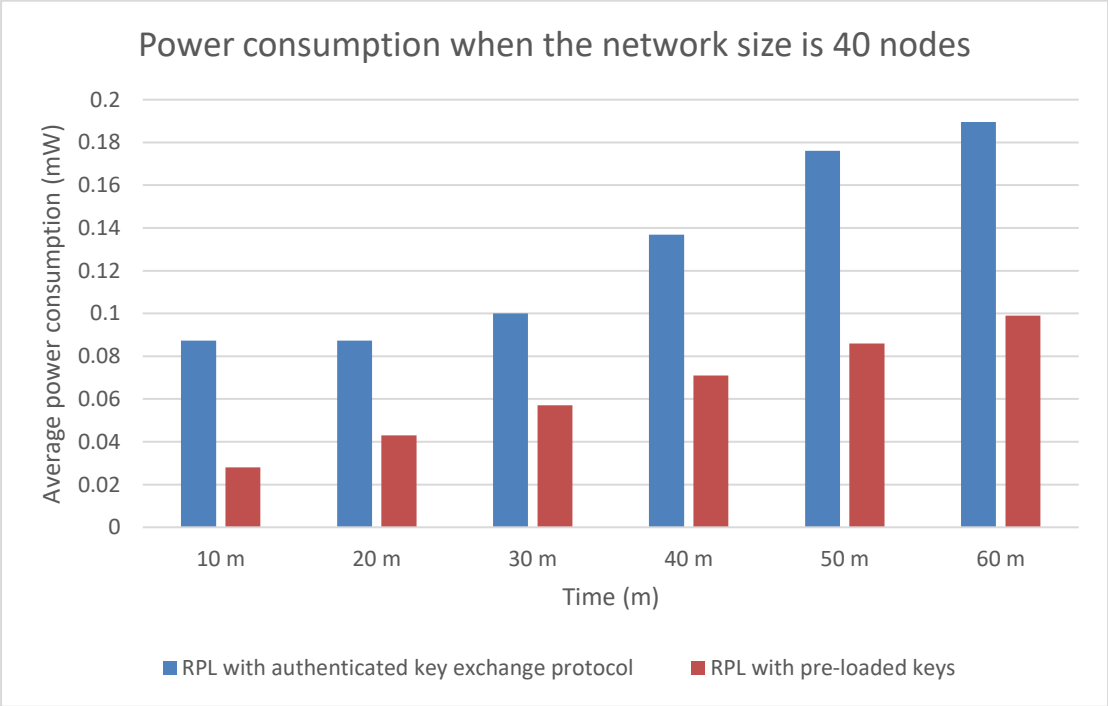

Figure 21: Power consumption when the network size is 40 nodes

Figures 18, 19, 20, and 21 show the average power consumption. As expected, the lowest power consumption is obtained with the RPL protocol with preload keys. In contrast, the proposed protocol with the secured key exchange causes a significant rise in the average power consumption. The main reasons for the increase were the computation overhead of the cryptographic and hash operations and the larger size of the transmitted packets. As predicted, the conclusions drawn with different network sizes demonstrate that as the network size grows, the power consumption increases due to an increase in the number of packets sent between network nodes and the time each node spends processing packets and receiving the key.

7. CONCLUSION

7.1 Summary

The RPL standard supports secure modes of operation. However, key exchange details were ignored. Our solution is to implement a secure authenticated key exchange, ensuring the integrity and authentication of received messages while providing tamper-proof data in insecure circumstances. To provide keys for secure RPL modes, we developed a trust establishment and key exchange mechanism based on ECC's asymmetric approach. This approach guarantees that nodes only transmit and use keys given by trustworthy nodes, unlike alternative solutions. Moreover, our protocol allows the key to be updated regularly preventing an attacker from obtaining the key through differential cryptanalysis. Our protocol successfully adds a level of security to the RPL protocol and prevents the distribution of erroneous routing information which impacts the overall network availability. The adjusting time of the key updating is the constraint on the proposed protocol. Using a short time interval will reduce the resources availability of the nodes while a longer duration will give attackers more time to compromise the keys. The proper time interval is determined to be 30 minutes according to our experiments. On the other hand, the results show that enabling the proposed protocol potentially impacts the RPL network. This impact can be translated into an overhead in computational and communication costs, an increase in energy consumption to around 100%, and an increase in the network formation time by around 400%. Additionally, as shown in the result, the protocol consumes 54.119 KB of ROM and 6.88 KB of RAM.

7.2 Future Work

Secure authenticated key exchange is important. It guarantees data authenticity and provides security protection for RPL. The suggested protocol offers a high level of security regardless of the computational and communication costs. In the future, we want to test the

proposed authenticated key exchange protocol in a real hardware environment instead of software emulation. Further optimization on the protocol is also possible to reduce resource overhead memory and processing.

8. GLOSSARY

Abbreviation	Definition
IoT	Internet of Things
IEEE	Institute of Electrical and Electronics Engineers
NIST	National Institute of Standards and Technology
ISO	International Organization for Standardization
DSA	Digital Signature Algorithm
ECC	Elliptic Curve Cryptography
RSA	Rivest Shamir Adleman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
LLNs	Low-power and lossy networks
IETF	Internet Engineering Task Force
RPL	Routing Protocol for Low power and Lossy Networks
DAG	Directed Acyclic Graph
DIS	DODAG Information Solicitation
DIO	DODAG Information Object
DAO	Destination Advertisement Object
DAO-ACK	Destination Advertisement Object Acknowledgement
OF	Objective Function
ETX	Expected Transmission Count
ICMPv6	Internet Control Messages Protocol version 6
UM	Unsecured Mode
PSM	Pre-installed Secured Mode
ASM	Authenticated Secured Mode

9. REFERENCES

[1] T. Winter, “RPL: IPv6 routing protocol for low-power and lossy networks,” RFC 6550, vol. 6550, pp. 1–157, 2012.

[2] “Number of IoT devices 2015-2025,” Statista. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/. [Accessed: 18-Jan-2022].

[3] A. E. Omolara et al., “The internet of things security: A survey encompassing unexplored areas and new insights,” Computers & Security, vol. 112, p. 102494, 2022.

[4] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, “A survey on IoT security: Application areas, security threats, and solution architectures,” IEEE Access, vol. 7, pp. 82721–82743, 2019.

[5] A. Gharaibeh, M. A. Salahuddin, S. J. Hussini, A. Khreishah, I. Khalil, M. Guizani, and A. Al-Fuqaha, “Smart cities: A survey on Data Management, security, and Enabling Technologies,” IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2456–2501, 2017.

[6] A. C. Jose and R. Malekian, “Improving smart home security: Integrating logical sensing into smart home,” IEEE Sensors Journal, vol. 17, no. 13, pp. 4269–4286, 2017.

[7] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, “A survey on IoT security: Application areas, security threats, and solution architectures,” IEEE Access, vol. 7, pp. 82721–82743, 2019.

[8] R. Hassan and T. Qamar, “Asymmetric-key cryptography for contiki,” Master's thesis, 2010.

[9] J. Lopez, “Unleashing public-key cryptography in wireless sensor networks,” Journal of Computer Security, vol. 14, no. 5, pp. 469–482, 2006.

[10] O. Gnawali and P. Levis, “The minimum rank with hysteresis objective function,” RFC 6719, 2012.

[11] A. Verma and V. Ranga, “Security of RPL based 6LoWPAN networks in the internet of things: A review,” IEEE Sensors Journal 20, vol. 20, no. 11, pp. 5666–5690, 2020.

[12] L. Wallgren, S. Raza, and T. Voigt, “Routing attacks and countermeasures in the RPL-based Internet of things,” International Journal of Distributed Sensor Networks, vol. 9, no. 8, p. 794326, 2013.

[13] A. Le, J. Loo, A. Lasebae, A. Vinel, Y. Chen, and M. Chai, “The impact of rank attack on network topology of routing protocol for low-power and lossy networks,” IEEE Sensors Journal 13, vol. 13, no. 10, pp. 3685–3692, 2013.

[14] R. Sahay, G. Geethakumari, and K. Modugu, “Attack graph — Based vulnerability assessment of rank property in RPL-6LOWPAN in IoT,” in 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), 2018.

[15] A. Mayzaud, R. Badonnel, and I. Chrisment, “Detecting version number attacks in RPL-based networks using a distributed monitoring architecture,” in 2016 12th International Conference on Network and Service Management (CNSM), 2016.

[16] P. Pongle and G. Chavan, “A survey: Attacks on RPL and 6LoWPAN in IoT,” in 2015 International Conference on Pervasive Computing (ICPC), 2015.

[17] S. Shreenivas and T. Raza, “Intrusion detection in the RPLconnected 6LoWPAN networks,” Proceedings of the 3rd ACM international workshop on IoT privacy, trust, and security (IoTPTS), 2017, pp. 31–38.

[18] A. Kamble, V. S. Malemath, and D. Patil, “Security Attacks and Secure Routing Protocols in RPL-based Internet of Things:Survey,” in International Conference on Emerging Trends & Innovation in ICT (ICEI), Pune, India, 2017, pp. 33–39.

[19] A. Dvir, T. Holczer, and L. Buttyan, “VeRA-version number and rank authentication in RPL,” 2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, 2011, pp. 709–714.

[20] M. Landsmann, M. Wahlisch, and T. Schmidt, “Topology Authentication in RPL,” in 2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2013.

[21] G. Glissa, A. Rachedi, and A. Meddeb, “A secure routing protocol based on RPL for internet of things,” in 2016 IEEE Global Communications Conference (GLOBECOM), 2016.

[22] B. Ghaleb, A. Al-Dubai, E. Ekonomou, M. Qasem, I. Romdhani, and L. Mackenzie, “Addressing the DAO insider attack in RPL’s internet of things networks,” IEEE Communications Letters, vol. 23, no. 1, pp. 68–71, 2019.

[23] W. Hu, P. Corke, W. C. Shih, and L. Overs, “SecFleck: A public key technology platform for wireless sensor networks,” in Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 296–311.

[24] W. Hu, H. Tan, P. Corke, W. C. Shih, and S. Jha, “Toward trusted wireless sensor networks,” ACM Transactions on Sensor Networks (TOSN), vol. 7, no. 1, pp. 1–25, 2010.

[25] S. Seeber, A. Sehgal, B. Stelte, G. D. Rodosek, and J. Schonwalder, “Towards a trust computing architecture for RPL in Cyber Physical Systems,” in Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013), 2013

[26] rpl-udp. [Online]. Available: https://anrg.usc.edu/contiki/index.php/RPL_UDP. [Accessed: 01-Apr-2022].

[27] nano-ecc [Online]. Available: GitHub - iSECPartners/nano-ecc: A very small ECC implementation for 8-bit microcontrollers. [Accessed: 06-Apr-2022].

[28] sha256 [Online]. Available: https://github.com/B-Con/crypto-algorithms/blob/master/sha256.c. [Accessed: 02-Apr-2022].

[29] K. MacKay, micro-ecc: ECDH and ECDSA for 8-bit, 32-bit, and 64-bit processors. [Online]. Available: https://github.com/kmackay/micro-ecc. [Accessed: 06-Apr-2022]. [Accessed: 02-Apr-2022].

[30] A. Dunkels, B. Gronvall, and T. Voigt, “Contiki-A lightweight and € flexible operating system for tiny networked sensors,” 29th annual IEEE international conference on local computer networks, 2004, pp. 455–462.

[31] “IPv6 over low power WPAN (6lowpan) -,” Ietf.org. [Online]. Available: http://www.ietf.org/html.charters/6lowpan-charter.html. [Accessed: 01-Apr-2022].

[32] Z. Shelby, K. Hartke, and C. Bormann, “The constrained application protocol (CoAP),” RFC 7641, 2014.

[33] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, “Cross-level sensor network simulation with COOJA,” in Proceedings. 2006 31st IEEE Conference on Local Computer Networks, 2006.

[34] contiki: The official git repository for Contiki, the open source OS for the Internet of Things. [Online]. Available: https://github.com/contiki-os/contiki. [Accessed: 26-Mar-2022].

[35] Zolertia, “Z1 features: Quick hardware tour," 2013. [Online]. Available: http://zolertia.sourceforge.net/wiki/index.php/Z1.[Accessed: 26-Mar-2022].

[36] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes, Powertrace: network-level power profiling for low-power wireless networks. 2011.

[37] D. Z1, Sourceforge.net. [Online]. Available: http://zolertia.sourceforge.net/wiki/images/e/e8/Z1_RevC_Datasheet.pdf. [Accessed: 26-Mar-2022].

[38] K. S. Bhandari, A. S. M. S. Hosen, and G. H. Cho, “CoAR: Congestion-aware routing protocol for low power and lossy networks for IoT applications,” Sensors (Basel), vol. 18, no. 11, p. 3838, 2018.