WIRELESS IPV6 SECURE ROUTING FOR CONTIKIRPL

A PROJECT REPORT

Submitted by

JAVID AHMED.Z

Register No: 13MCO11

in partial fulfillment for the requirement of award of the degree

of

MASTER OF ENGINEERING

in

COMMUNICATION SYSTEMS

Department of Electronics and Communication Engineering

KUMARAGURU COLLEGE OF TECHNOLOGY

(An autonomous institution affiliated to Anna University, Chennai)

COIMBATORE - 641 049

ANNA UNIVERSITY: CHENNAI 600 025

APRIL - 2015

BONAFIDE CERTIFICATE

Certified that this project report titled **"WIRELESS IPV6 SECURE ROUTING FOR CONTIKIRPL"** is the bonafide work of **JAVID AHMED.Z [Reg. No. 13MCO11]** who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Ms.M. ALAGUMEENAAKSHI

PROJECT SUPERVISOR

Department of ECE

Kumaraguru College of Technology

Coimbatore-641 049

SIGNATURE

Dr. RAJESWARI MARIAPPAN

HEAD OF THE DEPARTMENT

Department of ECE

Kumaraguru College of Technology

Coimbatore-641 049

The Candidate with university **Register No. 13MCO11** was examined by us in the project viva –voice examination held on.....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The concept of Internet of Things (IoT) is a new wave in the technology market that came into existence for reducing the gap between small devices and the IP world making them smart. One of the key issues in 6LoWPAN is standardizing a routing protocol. To address this need, IETF ROLL working group has proposed a standardized routing protocol called RPL. RPL has gained maturity in these years prompting researchers to focus on the design and deployment of RPL in real world applications such as care, military, home automation. There are many low power operating systems that support RPL such as TinyOS and Contiki. These operating systems are widely used to implement Internet of Things.

With increased use of 6LoWPAN in wireless sensor networks, routing security has become a basic necessity. Though 6LoWPAN with RPL is a very attractive technology, it will not beneficial to the end user if it's not secure. Knowing the memory and power constraint nature of WSN motes, the options of security for routing information has become very limited. To address this issue the paper attempts to implement and study the effects symmetric algorithms such as RC5 and Skipjack have on the mote behavior. An optimum security strikes a balance between memory and power consumptions such that no compromise is made on the application of the WSN. Many researchers have worked and analyzed different security algorithms for TinyOS in the application layer. In this paper security is provided in the network layer for RPL control messages in ContikiOS. RPL occupies considerable amount of memory and severely reduces the scope of many cryptographic algorithms. This paper prioritizes the implementation and analysis of secure ContikiRPL with the help of various encryption and message authentication algorithms compatible to the resource constrained WSN environment using COOJA simulator. The experimental results encompasses not only on the effects of security implementations on the behavior of RPL but also on the cost of memory and power consumption.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Overview of Wireless Sensor Networks

A wireless sensor network (WSN) consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location.

The WSN is built of "nodes" – from a few to several hundreds or even thousands, where each node is connected to one (or sometimes several) sensors. Each such sensor network node has typically several parts: a radio transceiver with an internal antenna or connection to an external antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery.

Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and communications bandwidth.

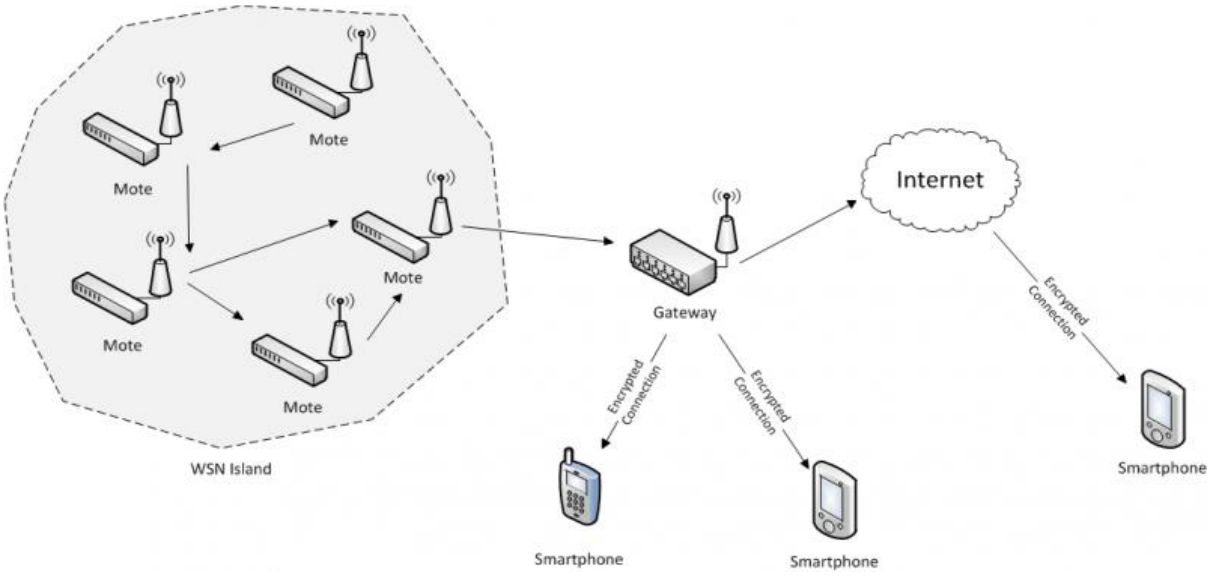

Figure 1 Wireless Sensor Network

1.2 Internet of Things

The Internet of Things (IoT) is generally thought of as connecting things to the internet and using that connection to provide some kind of useful remote monitoring or control of those things. IoT emerged with the idea that, even the tiniest of devices can be connected to the internet. There is plenty of research taking place in this area.

The Internet of Things (IoT) is the interconnection of uniquely identifiable embedded computing devices within the existing Internet infrastructure. Typically, IoT is expected to offer advanced connectivity of devices, systems, and services that goes beyond machine-to-machine communications (M2M) and covers a variety of protocols, domains, and applications. The interconnection of these embedded devices (including smart objects), is expected to usher in automation in nearly all fields, while also enabling advanced application.

The huge potential in IoT is basically due to the following reasons:

- Availability of very cheap and very low power tiny devices.
- Beyond static data, the potential lies in having and using live data.
- Availability of increasingly great amount of tools for data interpretation, knowledge mining and visualization.

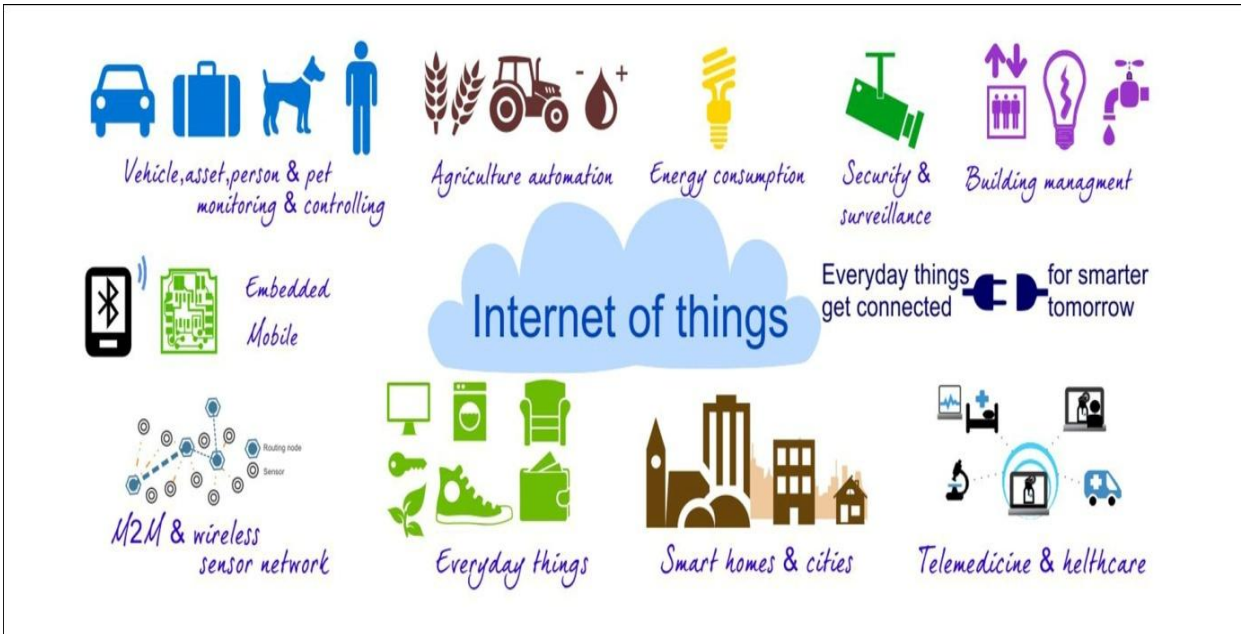

Figure 2 Internet of Things

1.3 6LoWPAN

The IoT environment consists of a huge number of devices with resource constraint characteristics such as short radio range, limited processing capability and short battery life. Therefore, the IoT implementation requires a communication protocol that can efficiently manage these conditions. 6LoWPAN is a promising solution with the idea of adding an adaption layer called 6LoWPAN in the network protocol stack for integrating low-power network such as IEEE 802.15.4 into IPv6. This solution can allow the use of the existing infrastructure (Internet Protocol (IP) network) to maximize the utilization of available resources while benefiting from the huge address space of IPv6.

6LoWPAN is an acronym of IPv6 over Low power Wireless Personal Area Networks. The 6LoWPAN network consists of one or more local LoWPANs, which are all connected by IPv6 to the Internet through a gateway (or border router). The LoWPAN devices are characterized by short radio range, low data rate, low power and low cost. The network, therefore, deals with small packet size, low bandwidth and requires resource saving for maintaining the life of network nodes. 6LoWPAN supports both star and peer-to-peer topology; however, the topology can be changed frequently because of uncertain radio frequency, mobility and battery drain.

In the typical model, IP is the only protocol used to connect different protocols from the data link and physical layer to multiple upper layer protocols. 6LoWPAN, however, utilizes the 6LoWPAN stack, a combination of LoWPAN adaptation layer and IPv6, to connect its WSNs to the Internet.

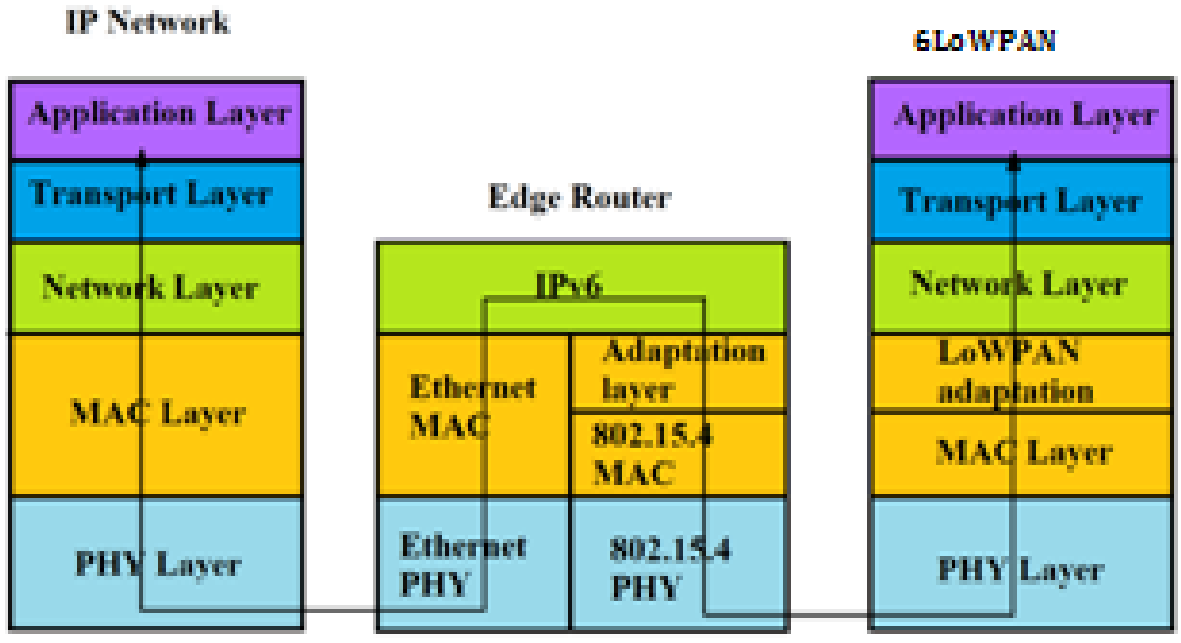

Figure 3 Bridge between 6LoWPAN and IP world using edge router

1.4 Routing for Low Power and Lossy Network (RPL)

One of the key issues in 6LoWPAN is routing. To address this need, IETF ROLL working group has proposed a routing protocol called RPL. RPL has gained maturity in these years prompting researchers to focus on the design and deployment of RPL in real world applications. But there are security concerns in RPL which prevents it from being used in arena where security is very vital.

It is a distance-vector and source routing protocol designed to work on multiple link layers. The RPL topology is based on Directed Acyclic Graph (DAG). Though DAG represents a tree like structure, unlike trees its nodes can be associated with more than one parent. The RPL network organizes as Destination Oriented DAG's (DODAG's). The most popular node or the node with access to internet is made the root. RPL is used to maintain and manage the network using control messages.

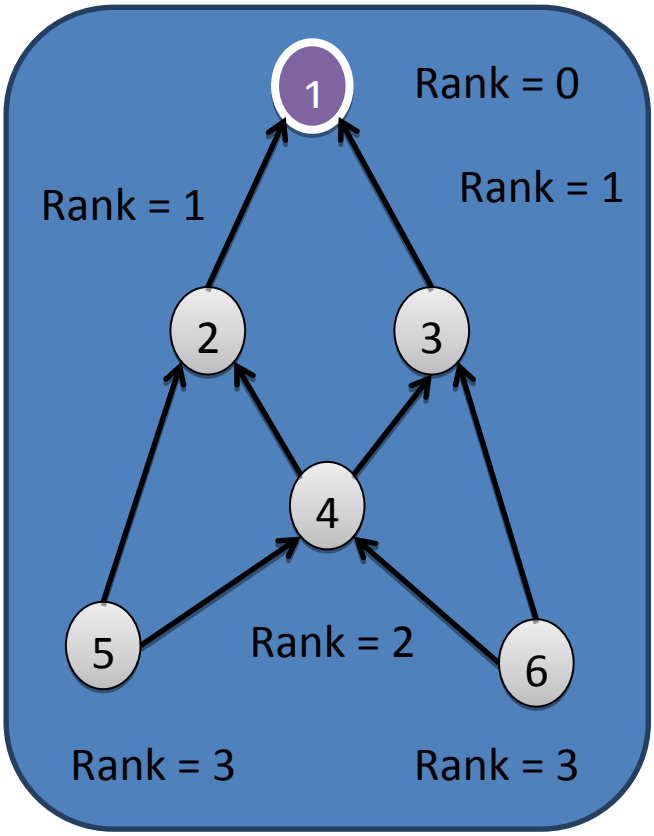

Figure 4 A simple DODAG

1.5 ICMPv6 RPL Control Message

Most RPL messages except the DAO/DAO-ACK messages have scope of a link. Their source address is a link local address and destination addresses is either all-RPL-nodes multicast address or a link-local unicast address.

The RPL control message is an ICMPv6 information message with request Type 155.

Type (8)	Code (8)	Checksum (16)
Base		
Option		

Figure 5 RPL control message

The code field identifies the type of RPL Control Message. The code are listed in the table 1.

Codes	RPL Control Message Types
0x00	DODAG Information Solicitation
0x01	DODAG Information Object
0x02	Destination Advertisement Object
0x03	Destination Advertisement Object Acknowledgement
0x80	Secure DODAG Information Solicitation
0x81	Secure DODAG Information Object
0x82	Secure Destination Advertisement Object
0x83	Secure Destination Advertisement Object Acknowledgement

Table 1 RPL message type codes

The higher order code (0x80 and above) indicates that the security is enabled and the structure resembles the figure 6.

Type (8)	Code (8)	Checksum (8)
Security		
Base		
Option		

Figure 6 Secure RPL Control message

The security provides parameters that provide confidentiality and authentication to the RPL message. Confidentiality is provided starting from the base to the end of the packet. Authentication is provided from security header to the end of the packet.

1.6 Security Fields

The fields in security section are shown in figure 7. The following section is described as in draft [11].

Counter is Time (T): If the Counter is Time flag is set then the Counter field is a timestamp. If the flag is cleared then the Counter is an incrementing counter.

Reserved: 7-bit unused field. The field must be initialized to zero by the sender and must be ignored by the receiver.

T (1)	Reserved (7)	Algorithm (8)	KIM (2)	Reserved (3)	LVL (3)	Flags (8)
Counter (32)						
Key Identifier						

Figure 7 Security header

Security Algorithm (Algorithm): The Security Algorithm field specifies the encryption, MAC, and signature scheme the network uses.

Key Identifier Mode (KIM): The Key Identifier Mode is a 2-bit field that indicates whether the key used for packet protection is determined implicitly or explicitly and indicates the particular representation of the Key Identifier field. The Key Identifier Mode is set one of the values from the table 2:

Mode	KIM	Meaning	Key Identifier Length (Octets)
0	00	Group key used. Key determined by Key Index field. Key source is not present. Key index is present.	1
1	01	Per-pair key used. Key determined by source and destination of packet. Key Source is not present. Key Index is not present.	0
2	10	Group key used. Key determined by Key Index and Key Source Identifier. Key Source is present. Key Index is present.	9
3	11	Node's signature key used.	0/9

		If packet is encrypted, it uses a group key, Key Index and Key Source specify key. Key Source may be present. Key Index may be present.	

Table 2 KIM Values

Resvd: 3-bit unused field. The field MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Security Level (LVL): The Security Level is a 3-bit field that indicates the provided packet protection. This value can be adapted on a per-packet basis and allows for varying levels of data authenticity and, optionally, for data confidentiality. The KIM field indicates whether signatures are used and the meaning of the Level field. Note that the assigned values of Security Level are not necessarily ordered-- a higher value of LVL does not necessarily equate to increased security. The Security Level is set to one of the values in the tables 3:

	KIM = 0, 1, 2	
LVL	**Attributes**	**MAC len**
0	MAC-32	4
1	ENC-MAC-32	4
2	MAC-64	8
3	ENC-MAC-64	8
4-7	Unassigned	N/A

Table 3 Security Level Field

The MAC attribute indicates that the message has a Message Authentication Code of the specified length. The ENC attribute indicates that the message is encrypted.

Flags: 8-bit unused field reserved for flags. The field must be initialized to zero by the sender and must be ignored by the receiver.

Counter: The Counter field indicates the non-repeating 4-octet value used to construct the cryptographic mechanism that implements packet protection and allows for the provision of semantic security.

Key Identifier: The Key Identifier field indicates which key was used to protect the packet. This field provides various levels of granularity of packet protection, including peer-to-peer keys, group keys, and signature keys. This field is represented as indicated by the Key Identifier Mode field and is formatted as figure 8:

Key Index (64)
Key Source (8)

Figure 8 KIM Format

Key Source: The Key Source field indicates the originator of a group key. This field is 8 bytes in length.
Key Index: The Key Index field allows unique identification of keys with the same originator. Each key originator is responsible to make sure that actively used keys have distinct key indices and that all key indices have a value unequal to 0x00. Value 0x00 is reserved for a pre-installed, shared key. This field is 1 byte in length.

1.7 RPL control messages

RPL messages are specified as a new type of ICMPv6 control messages. According to IANA, the RPL control message is composed of

(i) an ICMPv6 header, which consists of three fields: Type, Code and Checksum
(ii) a message body comprising a message base and a number of options.

The different type of messages describes in table are explained below:

1. DODAG Information Solicitation (DIS): The DIS message is mapped to 0x01, and is used to solicit a DODAG Information Object (DIO) from an RPL node. The DIS may be used to probe neighbor nodes in adjacent DODAGs. The current DIS message format contains non-specified flags and fields for future use.

Flags	Reserved	Options

Figure 9 DIS Format

2. DODAG Information Object (DIO): The DIO message is mapped to 0x01, and is issued by the DODAG root to construct a new DAG and then sent in multicast through the DODAG structure. The DIO message carries relevant network information that allows a node to discover a RPL instance, learn its configuration parameters, select a DODAG parent set, and maintain the DODAG. The DIO format is shown in figure 10.

<table>
<tr><td colspan="4">RPLInstanceID
(8)</td><td>Version Number
(8)</td><td colspan="2">Rank
(16)</td></tr>
<tr><td>G
(1)</td><td>0</td><td>MOP
(2)</td><td>Prf
(3)</td><td>DTSN
(8)</td><td>Flags
(8)</td><td>Reserved</td></tr>
<tr><td colspan="7">DODAGID
(128)</td></tr>
</table>

Figure 10 DIO Format

The main DIO Base Object fields are: (i) RPLInstanceID, is an 8-bit information initiated by the DODAG root that indicates the ID of the RPL instance that the DODAG is

part of, (ii) Version Number, indicates the version number of a DODAG that is typically incremented upon each network information update, and helps maintaining all nodes synchronized with new updates, (iii) Rank, a 16- bit field that specifies the rank of the node sending the DIO message, (vi) Destination Advertisement Trigger Sequence Number (DTSN) is an 8-bit flag that is used to maintain downward routes, (v) Grounded (G) is a flag indicating whether the current DODAG satisfies the application-defined objective, (vi) Mode of Operation (MOP) identifies the mode of operation of the RPL instance set by the DODAG root. Four operation modes have been defined and differ in terms of whether they support downward routes maintenance and multicast or not. Upward routes are supported by default. Any node joining the DODAG must be able to cope with the MOP to participate as a router, otherwise it will be admitted as a leaf node, (vii) DODAGPreference (Prf) is a 3-bit field that specifies the preference degree of the current DODAG root as compared to other DODAG roots. It ranges from 0x00 (default value) for the least preferred degree, to 0x07 for the most preferred degree, (viii) DODAGID is a 128-bit IPv6 address set by a DODAG root, which uniquely identifies a DODAG. Finally, DIO Base Object may also contain an Option field.

3. Destination Advertisement Object (DAO): The DAO message is mapped to 0x02, and is used to propagate reverse route information to record the nodes visited along the upward path. DAO messages are sent by each node, other than the DODAG root, to populate the routing tables with prefixes of their children and to advertise their addresses and prefixes to their parents. After passing this DAO message through the path from a particular node to the DODAG root through the default DAG routes, a complete path between the DODAG root and the node is established. The DAO format is shown in figure 11.

RPLInstanceID (8)	**K (1)**	**D (1)**	**Flags**	**Reserved**	**DAOSequence**
DODAGID (128)					
Options					

Figure 11 DAO Format

As shown in the figure 11, the main DAO message fields are: (i) RPLInstanceID, is an 8-bit information indicates the ID of the RPL instance as learned from the DIO, (ii) K flag that indicates whether and acknowledgment is required or not in response to a DAO message, (iii) DAOSequence is a sequence number incremented at each DAO message, (iv) DODAGID is a 128-bit field set by a DODAG root which identifies a DODAG. This field is present only when flag D is set to 1.

4. Destination Advertisement Object (DAO-ACK): The DAOACK message is sent as a unicast packet by a DAO recipient (a DAO parent or DODAG root) in response to a unicast DAO message. It carries information about RPLInstanceID, DAOSequence, and Status, which indicate the completion. Status code are still not clearly defined, but codes greater than 128 mean a rejection and that a node should select an alternate parent.

5. Secure Variants: The secure variants of the above message consists of the same fields with an added security section shown in figure 7.

1.8 DODAG Construction

The DODAG construction is based on the Neighbour Discovery (ND) process, which consists in two main operation:
(1) DIO messages are broadcasted by DODAG root to build downward routes from root to client nodes, (2) unicast DAO messages sent by client nodes to build routes in upward direction. In process of constructing a new DODAG, the root broadcasts a DIO to announce its Rank, its DODAGID information to facilitate nodes to learn their positions in the DODAG. This message received by a client node which is either a node willing to join the DODAG or an already existing node. When a node willing to join the DODAG receives the DIO it (i) adds the sender address to its list of parent, (ii) rank is computed by the Objective Function, such that the node's rank is higher than that of its parents, and (iii) forward the DIO with updated rank. The client node takes one node as the preferred parent among its parents list as the default node for forwarding inward traffic. When a node in a DODAG receives a second DIO message, it may proceed by three different ways (i) the DIO message is discarded (ii) the message is processed to maintain its position or (iii) improve its position by getting a lesser rank. A node changing the rank will have to discard its entire parent list and create a new list to avoid looping. The flowchart presented in figure 12 depicts the DODAG router operation. After the construction all client nodes will have a default upward route which can transmit inward traffic. The default route is formed by the preferred parent of each node. If the Mode of Operation flag in the DIO is not zero, downward routes from are supported and have to be maintained. In this case, all client nodes send a unicast DAO to the root to determine the downward route information. When travelling back to the DODAG root, visited nodes are recorded in the packet along the upward route, and complete route is then established between the DODAG root and the client node. RPL specifies two modes of operations to maintain downward routes in an RPL instance:

- **Storing mode:** in the storing mode, a DAO message is sent in unicast by the child to the selected parent, which is able to store DAO messages received by its children before sending the new DAO message with aggregate reachability information to its parent. the flowchart in figure 13 depicts the DAO process.
- **Non-storing mode:** in the non-storing mode, the DAO message is sent in unicast to the DODAG root, thus, intermediate parents do not store DAO messages, but only insert their own addresses to the reverse route stack in the received DAO message, then forwards it to its parent.

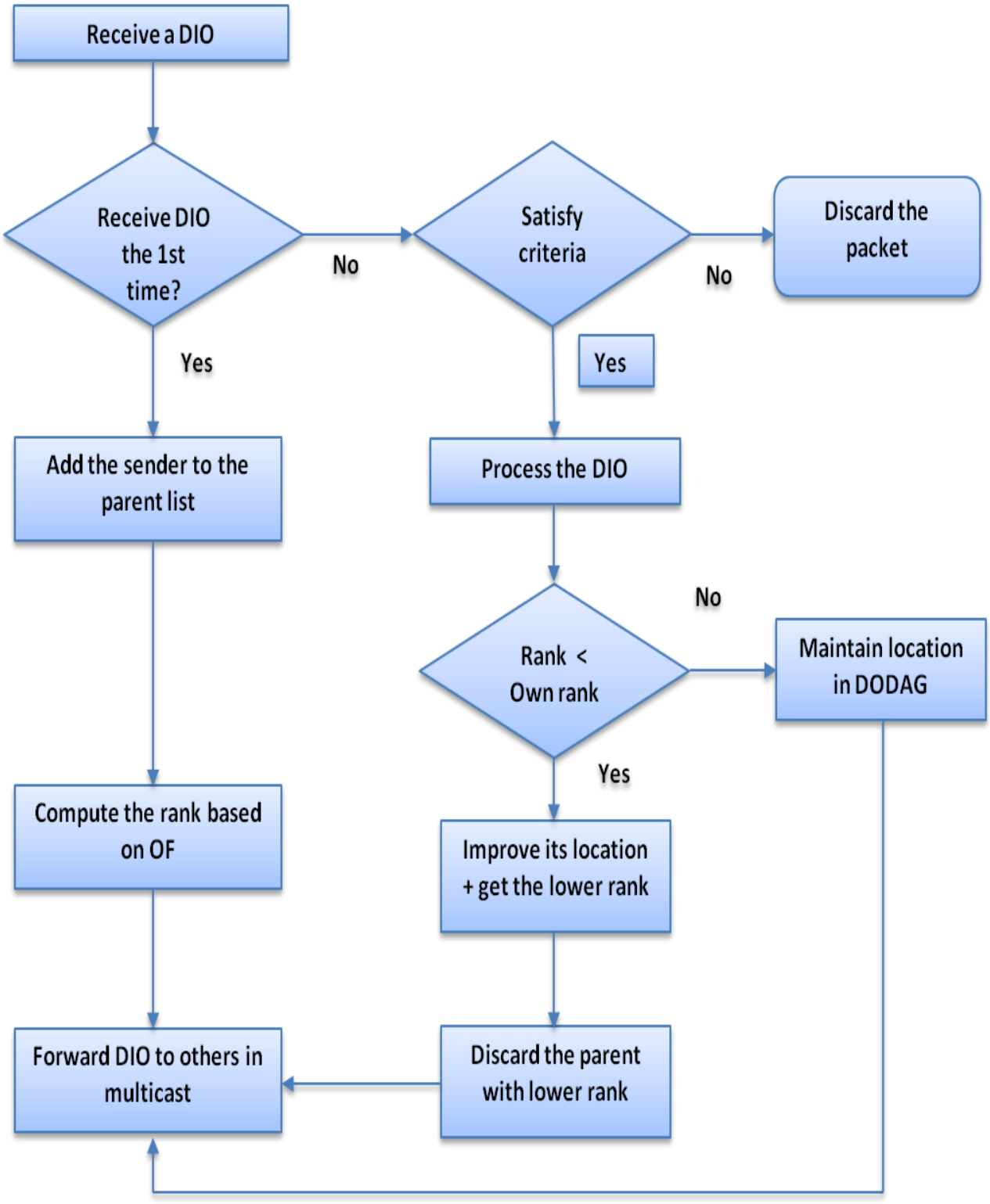

Figure 12 DIO Process

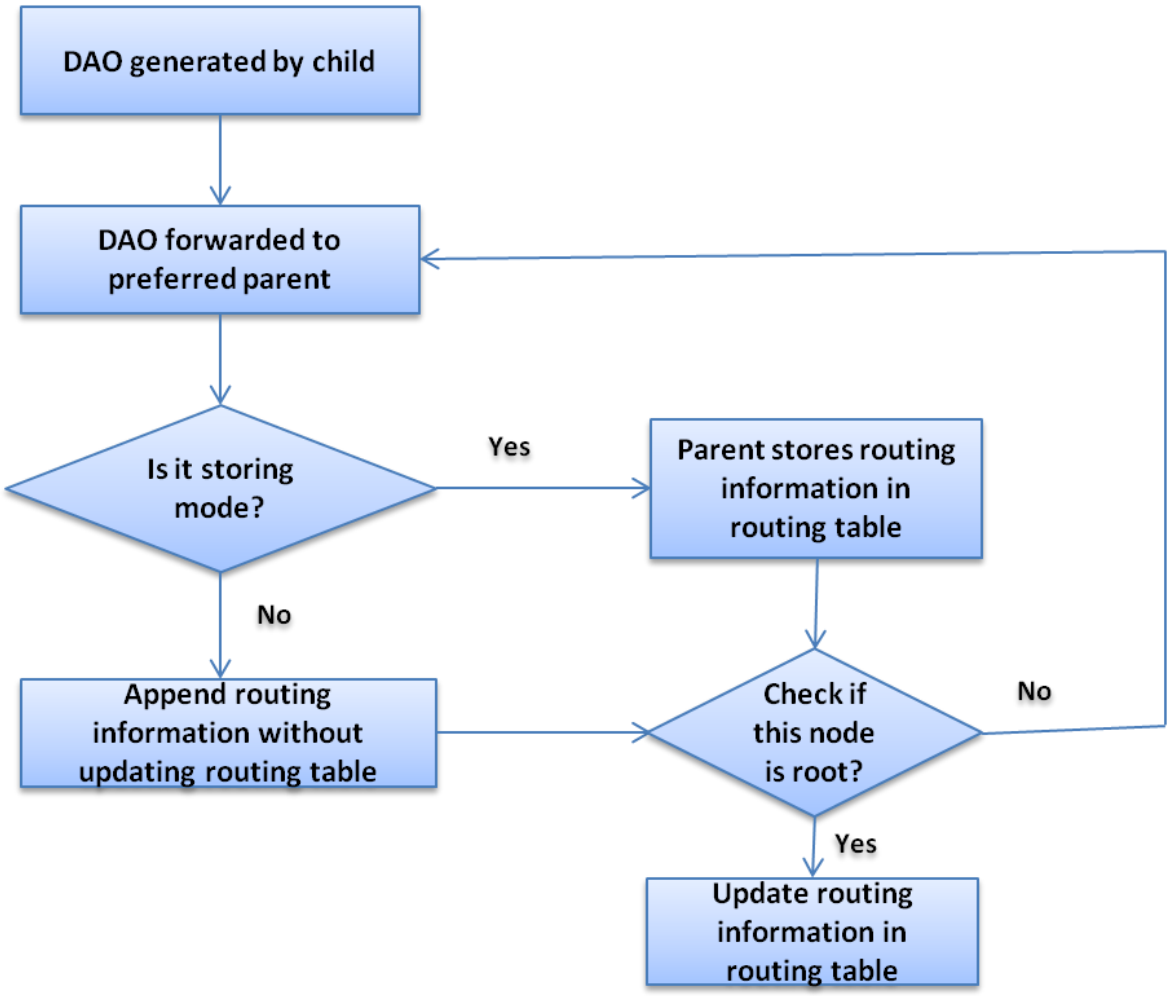

Figure 13 DAO process

1.9 Contiki-OS for Internet of Things

Contiki is an open source operating system used to facilitate Internet of Things. It is designed to operate in low power environment. The features of Contiki are listed below:

- **Memory Allocation:** Contiki is designed for systems with only a few kilobytes of memory. Therefore Contiki is highly memory efficient and provides memory allocation mechanisms: memb - memory block allocation, mmem - a managed memory allocator, and also supports malloc.

- **Full IP Networking:** Contiki has a full IP network stack, supporting IP protocols such as UDP, TCP, HTTP and new low-power standards like 6lowpan, RPL, and CoAP. The Contiki IPv6 stack is certified by IPv6 Ready Logo program. The stack was developed and contributed to Contiki by Cisco.

- **Power Awareness:** Contiki is designed to operate in low power environment. Contiki also provides mechanisms to calculate power consumption and understand where the power is being spent..

- **Dynamic Module Loading:** Contiki supports run-time loading and linking of modules. This is useful when the behaviour of application changes after deployment.

- **The Cooja Network Simulator:** Contiki can be used to make large wireless networks. Cooja, a Contiki network simulator is makes it easy for testing and deploying using a simulation environment.

- **Sleepy Routers:** This is a mechanism for making the routers sleep in between relaying messages. This is provided by ConikiMAC duty cycle mechanism.

- **Protothreads:** This saves memory and also provides a nice code flow. It is a combination of event-driven and multithreading programming.

- **The Rime Stack:** At times when IPv6 stack is a overburden in a network we can have rime stack to facilitate simple operations like unicast and multicast messages.

- **Build System:** This build system makes it easy for Contiki to be compiled in many platforms.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

The literature survey gives a brief on the resources that have been studied for better understanding and current research trends in the area of proposed research.

2.2 Security Threats and Framework

IETF has published many drafts that address the security needs and concerns for RPL. A detailed description about the working of RPL and possibility of establishing routing security is discussed in the IETF draft [15]. This draft defines the packet structure of RPL messages and in its secure and unsecure implementations. It further sheds light on the functionality of secure RPL. The provisions are made for key management mechanisms but is not implemented due to memory concerns. The drafts [16] and [17] explain the security framework for RPL along with the security threats and their countermeasures. The unsecured RPL can be subjected threats on its integrity and availability [17]. The countermeasures are application specific and it generally revolves around confidentiality and integrity. It also adapts the assessments, to the issues and constraints that are specific for the Low power and Lossy Networks. A systematic approach is used in defining and evaluating the security threats and identifying applicable countermeasures. These assessments provide the basis of the security recommendations for incorporation into low power, lossy network routing protocols [16]. The proposed work implements secure ContikiRPL based on [15]. It also takes into account some of the necessary consideration for security defined in drafts [16] and [17].

The security for any network depends on the vulnerabilities in the network that exposes the information to unauthorized individual. In its unsecured form RPL reveals its routing information and thus becomes vulnerable. Providing security to IPv6/6LoWPAN network in IoT is not easy considering that the WSNs is exposed to the untrusted internet. So Wallgren et al. [11] studied different routing attacks and its countermeasures for RPL.

2.3 Cryptographic Algorithms

The level of confidentiality and authentication depends on the choice of the algorithm. The papers [1], [6], [12] and [10] give insight on cryptographic algorithms and their performance in memory and power constraint environment. Chang et al. [1] discusses the energy consumption of implementing security for messages communicating in wireless medium. It was observed that the energy spent in CPU for encryption contributes only little to the energy consumption compared to the radio ON time. The lifetime of the mote is reduced by half when security is implemented. Jongdeog Lee et al. [6] have evaluated the cost of security in wireless sensor networks using TinyOS. It deals extensively about many cryptographic and message authentication algorithms which are implemented and practically analyzed upon different motes. The analysis was concluded in favor of RC5 for wireless sensor motes. Y. W. Law et al. [18] has also analyzed cost measures for different cipher modes and emphasized on the ciphers which are good candidates for WSN. It concludes on using different algorithms for different applications. C. Karlof et al. [2] has proposed TinySec

as a link layer security architecture for TinyOS in which he recommends the use of RC5 and Skipjack. As per the recommendation from various researchers, the proposed work discusses the suitability of RC5 and Skipjack algorithms for establishing secure ContikiRPL.

2.4 Internet of Things

Jean-Philippe Vasseur, and Adam Dunkels in their book [7] have explained the concept of interconnecting smart objects with IP. The digital revolution gave rise to the concept of smart object. It bridges the gap between the physical world and the internet. The IP stack to adapt this bridge using 6LoWPAN is discussed. The book also briefs on the concept of edge router and border router. It discusses different operating system and commercial and open source IP stacks to be built for internet of things.

2.5 RPL and its performance

The papers [9], [13] and [15] discusses elaborately on the execution and performance of RPL. Olfa Gaddour et al. [13] has done a first survey on RPL. They describe the advantages of RPL over other protocols. The operation of RPL and its functional definitions are given by the draft [15].

2.6 ContikiOS

Joakim Eriksson et al. [8] have implemented an accurate power profiling tool based on COOJA simulator and MSPSim emulator. This implementation makes calculating power easy inside the simulation environment. Powertrace app inside COOJA can be used to log the time spent by a mote in CPU processing, low power mode, radio on and radio off time. Adam Dunkels et al. [3] presented an elaborate discussion on IPv6 implementation and low power mechanisms in Contiki. The low power mechanism is achieved using ContikiMAC mechanisms. Adam Dunkels et al. [4] have developed and proposed Contiki operating system as a lightweight and compact system capable of dynamic loading. The operating system can be used to build and compile at run time. Farooq et al. [5] have analyzed the node throughput and channel utilization of Contiki implemented on IEEE 802.15.4. Many researchers and companies are moving towards ContikiOS for TinyOS. Ko et al. [10] has discussed about the interoperability of contikiRPL and tinyRPL. Tsiftes et al. [14] showed that Tmote sky running with IPv6 routing of contikiRPL has improved battery life of several years and packet delivery to sink as 0.6 packets every second.

RPL implementation occupies a considerable amount of memory in Contiki. This reduces the scope of security options to be implemented on a resource constrained environment. To the best of our knowledge this is the first paper to implement and analyze security for ContikiRPL.

CHAPTER 3

SECURE CONTIKIRPL

3.1 Proposed Architecture

The existing RPL does not have any security. The proposed method is to implement security in the network layer. The confidentiality and authentication is provided by using RC5 and Skipjack along with CBC-MAC. The routing security is provided by creating secure RPL messages. The location of secure routing in network layer is shown in the figure 14.

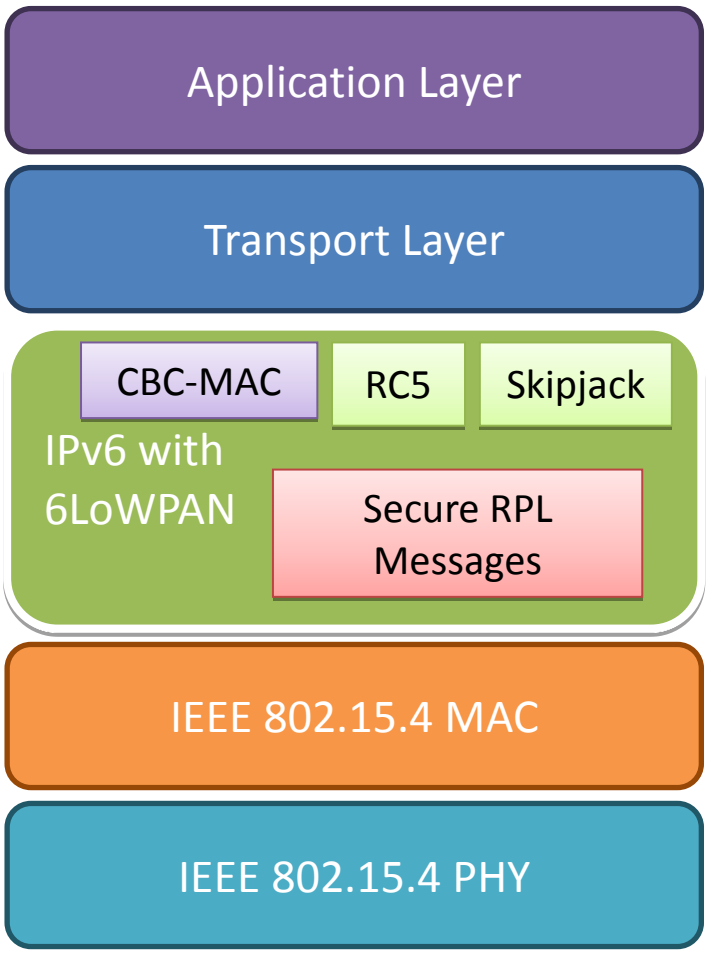

Figure 14 Position of secure RPL messages

- **Physical Layer:** It is used to provide data transmission and as well as interface to physical layer management utility. Tmote sky uses 250kbps 2.4GHz IEEE 802.15.4 Chipcon Wireless Transceiver which enables it to interoperate with other IEEE 802.15.4 devices. The onboard antenna has range of 50m indoor and 125m outdoor. The radio used is CC2420.

- **MAC Layer:** It enables transmission of MAC frames using physical channel. CSMA is generally used in the MAC layer. Contiki uses ContikiMAC mechanism to save power.

- **Network Layer:** 6LoWPAN adaptation layer is used in the network layer. This enables the IPv6 packets from the internet to communicate with the wireless device. The work involves implementing security for routing protocol (RPL) used in 6LoWPAN.
- **Transport Layer:** It decides whether the communication is UDP or TCP. Contiki has the option to completely disable TCP for the purpose of saving memory. In this work UDP application is used.
- **Application Layer:** The application layer can have protocols like HTTP, CoAP etc.

The unsecured control messages easily disclose traffic information and enable an outside node to join the network. In order to address this, the work provides security to the control messages by the parameters that are defined in the security header. The outside node which receives the control message will verify if the security parameters are compatible with its parameters. If it's compatible then the node joins the network. Otherwise it is barred from joining.

3.2 Existing vs Proposed Packet Structure

The work uses secure variants of RPL control messages. The security is provided by RC5 and skipjack algorithms and CBC-MAC algorithm.. The comparision of existing and proposed packet structure is shown in figure 15.

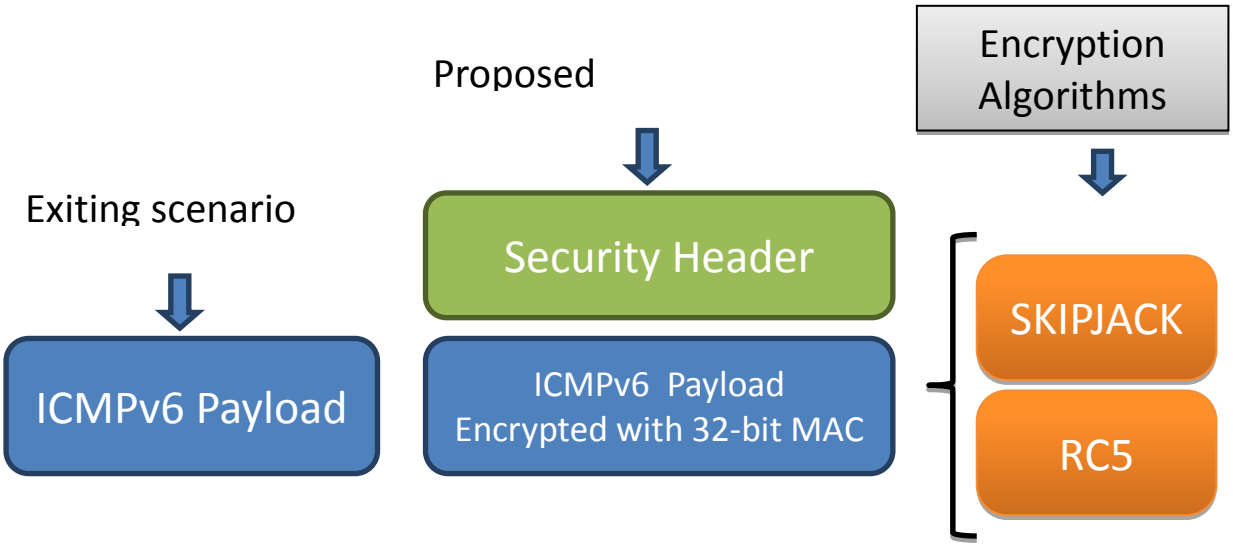

Figure 15 Existing vs Proposed method

3.3 Security Header

The security header fields gives flexibility to implement any cryptographic algorithm through an 8-bit algorithm field in the security header section. The security level field LVL defines varying levels of data authenticity and optionally data confidentiality. The Key Identifier Mode (KIM) determines whether the key used for confidentiality and authentication is determined implicitly or explicitly. A sample security header format with

values is shown in figure 16. In the implementation RC5 is given value 2 and Skipjack is given value 3 as value 1 has been allocated to the AES algorithm as default option in the IETF draft. The sample header in Fig 16 portray the usage of RC5 algorithm. The sample value for KIM is 0, LVL is 1 which implies that the key is determines by the key index field and security level includes both encryption and authentication. The counter field T is set to 0. Thus the counter field can have any non repeating random values. Since the KIM field is zero, only key index field exists. The sample value is 3, but it can take any value based on the implementation. The sample format has to be compared with security header format in figure 7 to gain better understanding.

0 (T)	**0000000 (Reserved)**	**00000010 (Algorithm)**	**00 (KIM)**	**000 (Resv)**	**001 (LVL)**	**00000000 (Flag)**
00100100110010111100110010101010 (Counter)						
00000011 (key index)						

Figure 16 Sample security header format

3.4 Securing nodes in the same network from each other

The work uses the key index field to announce to the receiving mote the key used by the sender mote. If the receiving mote supports the key index specified by the sender then it processes the message. A mote can support more than one key index. The root mote is designed to support the entire key index supported by its sub ordinate motes. The communication between mote supporting different key index can be made through the root node. This facilitates the implementation of filters in the root mote,

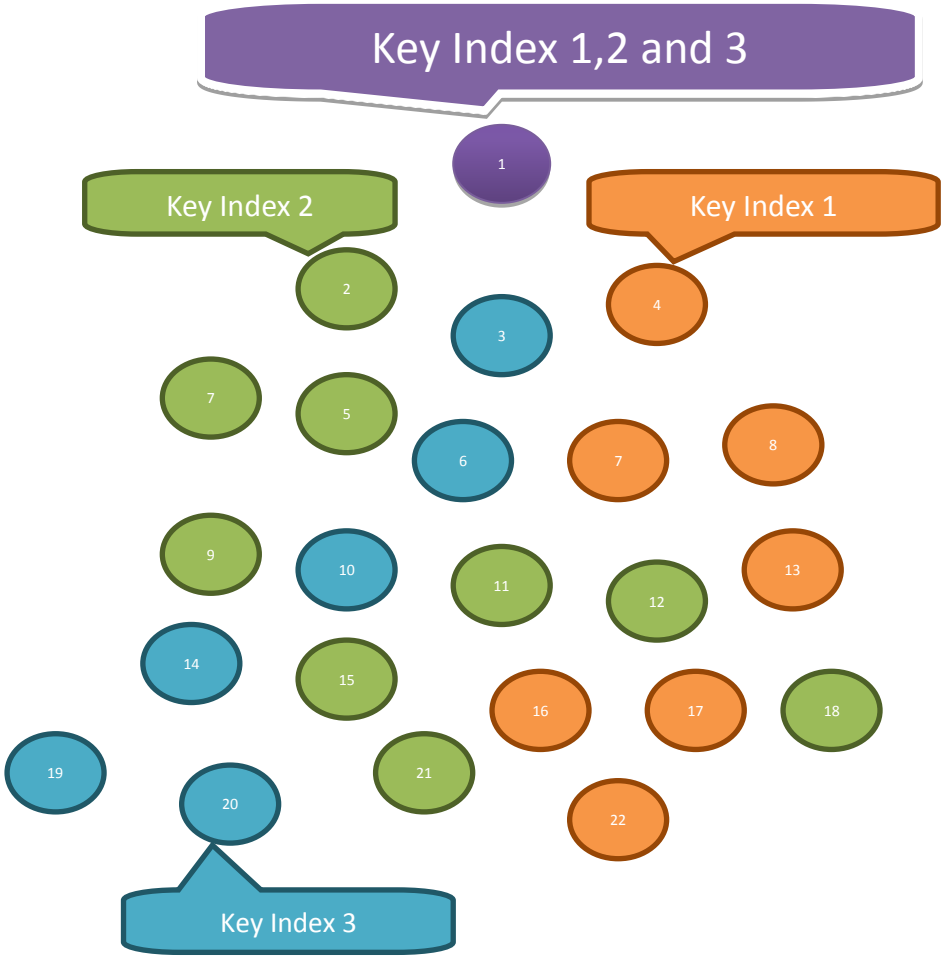

Figure 17 Motes transmitting with varying key index

The advantage of the using key index is that the routes for a specific application can be secured because the mote can take as a parent only that which supports its key index. This makes the network in figure 17 to appear as three networks with root mote as a router connecting these three networks.

3.5 Implementation of secure ICMPv6 RPL control messages

The first step is to add a security header with fields described in figure 7. The confidentiality and authentication is defined by the values in security header. When we mean security the authentication is mandatory and confidentiality depends on the value of security field in the security header. The authentication is done using CBC-MAC. The confidentiality is provided starting from the bits of the security header to the end of the packet. But message authentication code is generated from the security section to the end of the packet. The diagrammatic representation of the steps to construct secureRPL messages is shown in figure 18.

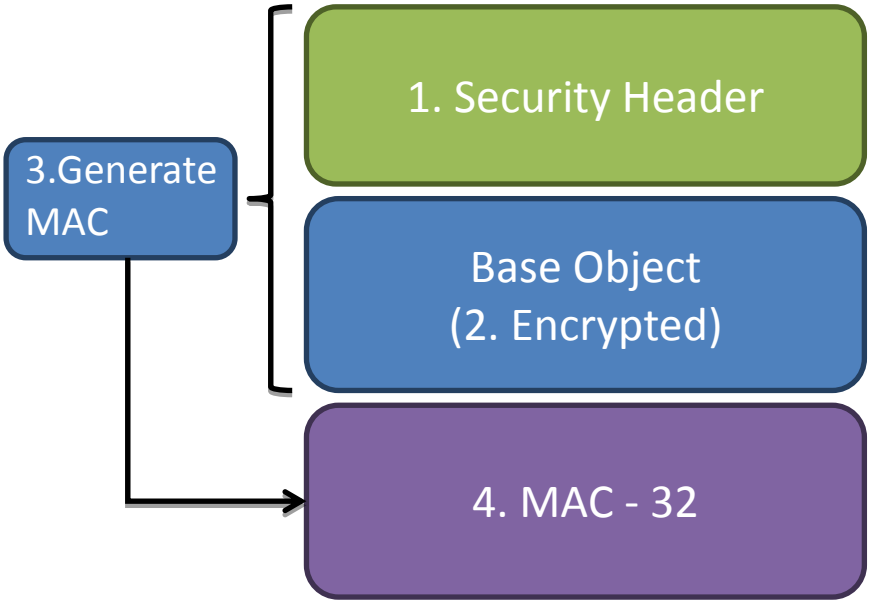

Figure 18 Steps to construct Secure RPL Messages

The steps for constructing secure RPL packets are:

5.1 Attach the security header on top of the base object.
5.2 Encrypt only the base object.(optional) .
5.3 Generate MAC for the entire packet.
5.4 Append the generated MAC at the end

3.6 Encryption Algorithm

3.6.1 Block Cipher

Block ciphers are known to be the most efficient in terms of energy consumption and latency when compared to other security algorithms. This is the reason why they are preferred for use in WSN environments. The block ciphers studied are RC5 and Skipjack. The choice of algorithm is due to the memory constraint nature of Tmote sky. Other standard algorithms like AES did not fit in to the available memory space. Also C. Karlof [2] recommended the use of RC5 and Skipjack as encryption algorithms for wireless sensor mote using TinyOS.

3.6.2 RC5

An advantage of RC5 is its flexibility. Unlike other encryption algorithms, RC5 has a variable block size (32, 64,or 128 bits), number of rounds (0...255), and key size (0...255 bits). The values of those parameters determine the level of security of the algorithm.

3.6.3 SKIPJACK

SKIPJACK is a block cipher developed by the U.S. National Security Agency (NSA). The algorithm is an unbalanced Feistel network with 32 rounds. Skipjack uses an 80-bit key for encryption and decryption and the size of the data blocks is 64 bits. Since its parameters like key size and block size are constants, SKIPJACK is not as flexible as AES or RC5.

3.7 Message Authentication Code

Without authentication, the receiver cannot know who the source of the message is or if the data has been manipulated. Attackers can take advantage of this weakness and flip bits in the message, which results in predictable changes to the plaintext. These changes allow the attackers to infer information about the original message. Authentication code is used to address this vulnerability

3.7.1 CBC-MAC

Cipher block chaining MAC (CBC-MAC) is the customary way to construct a MAC from a block cipher. CBC-MAC carries the encryption as CBC, and takes the last result as a MAC. An important property is that CBC and CBC-MAC must use a different key since otherwise even if changes are made to the message the MAC tag could still be valid. In addition, CBC-MAC is not secure for arbitrary-length messages. The draft [15] recommends use of CBC-MAC because the mote could not afford to have different algorithms for encryption and message authentication. The advantage of using CBC-MAC is that it can use the existing cryptographic algorithm to generate message authentication codes, thus saving significant memory.

CHAPTER 4
GETTING STARTED

4.1 Analysis

The analysis is done using Tmote sky in COOJA simulator. The analysis will discuss the following implementation based on the choice of algorithms

1.Unsecured

2.RC5

1. MAC
2. ENC - MAC

3.Skipjack

1. MAC
2. ENC-MAC

The data for analysis is obtained from mote outputs which are saved in a log file. Perl script is used to analyze the log file for calculate power consumption and convergence time of the deployed network. The analysis is illustrated in figure 19.

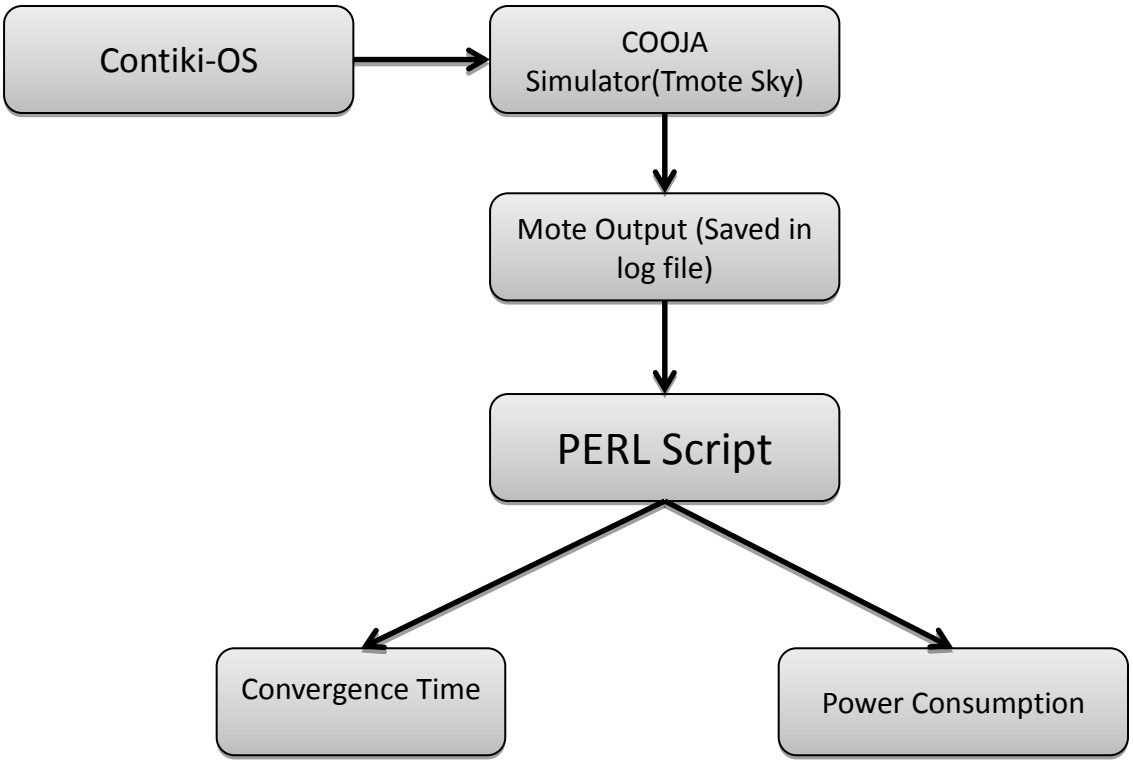

Figure 19 Steps for analyzing log file

4.2 Technologies Used

The programming and scripting languages used are:

- **C - language:** It is a widely used computer programming language.
- **Make files:** They are text files following a particular syntax. A software can be build from scratch using Make Utility.
- **PERL:** It is a scripting language used to manipulate and process text files.
- **Apache ant:** It is a Java based build tool. It is written in XML and has the advantage that it is portable, open standard and very simple to understand.

The work environment set up are :

- **Operating system:** Ubunto Linux 14.04.
- **System Processor :** Intel core i5 @ 3.20GHz
- **System RAM capacity:** 8GB.

Contiki based parameters are tabulated in table 5.

PARAMETERS	UNSECURED	RC5-MAC	RC5-ENC-MAC	SKIPJACK-MAC	SKIPJACK-ENC-MAC
Contiki Version	**2.7**	**2.7**	**2.7**	**2.7**	**2.7**
DIO Interval Minimum (Seconds)	**10**	**10**	**10**	**10**	**10**
DIO Doubling rate (Seconds)	**18**	**18**	**18**	**18**	**18**
DIO Redundancy Rate (Seconds)	**20**	**20**	**20**	**20**	**20**
Algorithm	-	**RC5(2)**	**RC5(2)**	**SKIPJACK(3)**	**SKIPJACK(3)**
Security Level (LVL)	-	**0**	**1**	**0**	**1**
Counter	-	**0**	**0**	**0**	**0**

Table 4 Contiki based parameters

COOJA based parameter are

- **Radio medium:** Unit Disk Graph Medium (UDGM): Distance Loss
- **Mote startup delay (ms):** 1000
- **Number of Motes:** 50

4.3 Contiki-OS Structure

The ContikiOS is written in basic C language. The figure 20 shows the different folders that come with Contiki 2.7. The apps folder consists of application that can be run on Contiki. The core consists of source code for the IP stack. The CPU folder has source codes for processors that can run Contiki. The platform consists of readily available motes that can be used to build Contiki. The Makefile.include compiles and builds the entire Contiki system.

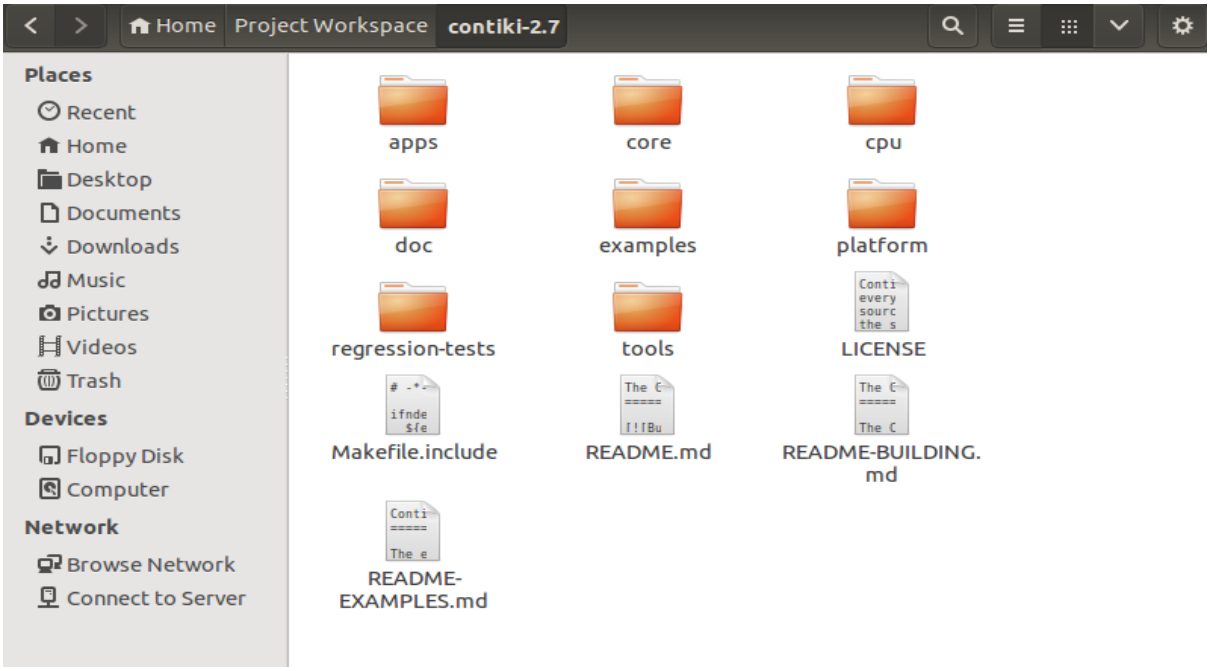

Figure 20 Contiki 2.7

A security algorithm folder created for this project shown in figure 21 shows many algorithms of which only RC5 and Skipjack are supported.

Figure 21 Security Algorithm

The RPL folder shown in figure 22 contains the source code for RPL functions. The files modified are

- Makefile.rpl

- rpl-icmp6.c
- rpl-conf.h
- rpl-private.h

The new files created for secure RPL are

- rpl-security.c
- rpl-security.h

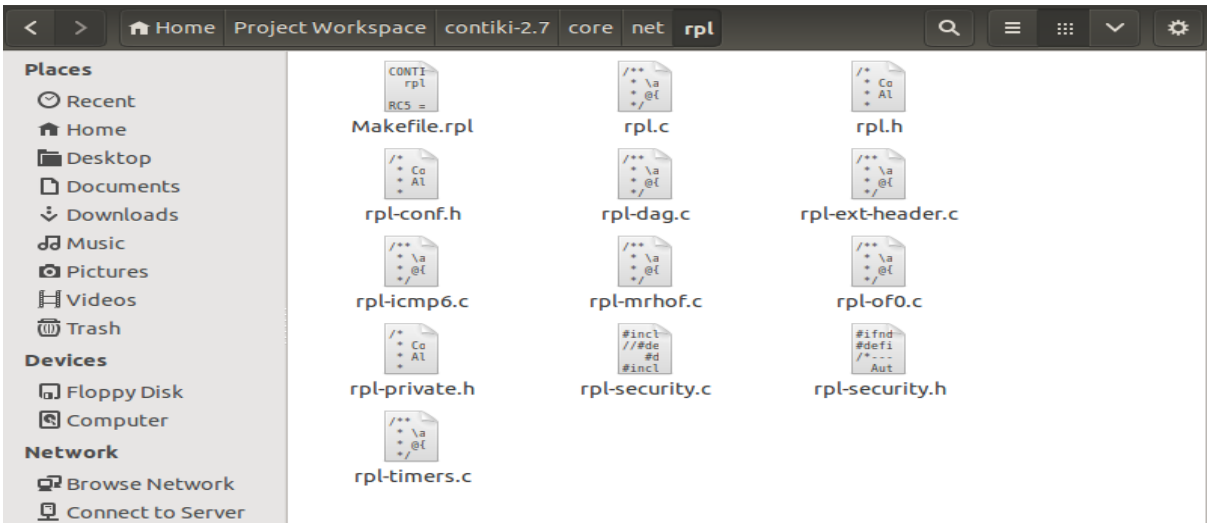

Figure 22 RPL source files

The rpl-security.c provides security header creation, encryption and MAC mechanisms. The rpl-conf.h file should have the definition in figure 23 added to it. The definitions can be changed based on the security requirement.

```
/*
 *Default security configuration
 */

#define RPL_SECURITY_COUNTER_SET INCREMENTING_COUNTER
#define RPL_ALGORITHM RC5
#define RPL_KIM KIM_MODE_0
#define RPL_LVL LVL1
#define RPL_COUNTER 0
#define RPL_KEY_INDEX 0x00
#define RPL_KEY_SOURCE 0x00
```

Figure 23 Security configuration in rpl-conf.h file

When changes are made to the RPL_ALGORITHM field, relevant changes have to be made in the Makefile.rpl file shown in figure 24. The '#' symbol denotes comment. Depending on the RPL_ALGORITHM the Make file is changed.

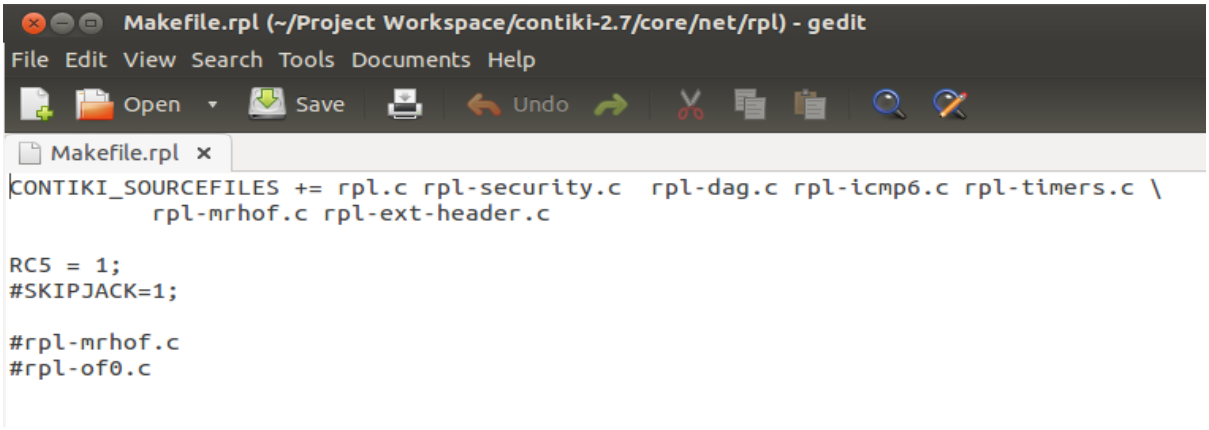

Figure 24 Makefile for RPL

After all the changes are made in RPL, a rpl-udp application is run using a server and many clients motes. The udp-server.c and udp-client.c can be used to run rpl-udp. After compilation a .sky file will be formed which is a complete build. A simulation environment called COOJA provided by Contiki makes life simple. The udp application can be compiled and build on simulation motes for testing purposes. The rpl-udp files are shown in figure 25.

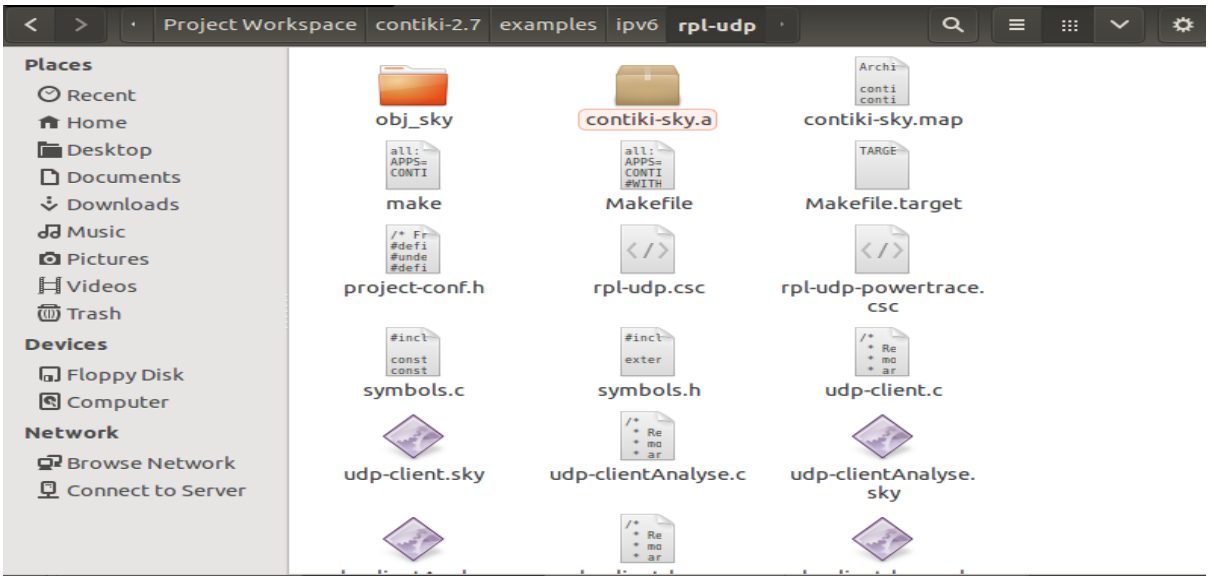

Figure 25 Source files for rpl-udp

In order to save memory, project-conf.h file should have the definitions shown in figure 26 which disables the TCP configuration (UIP_CONF_TCP) and limits the maximum number of neighbor (NBR_TABLE_CONF_MAX_NEIGHBOURS) and route(UIP_CONF_MAX_ROUTES).

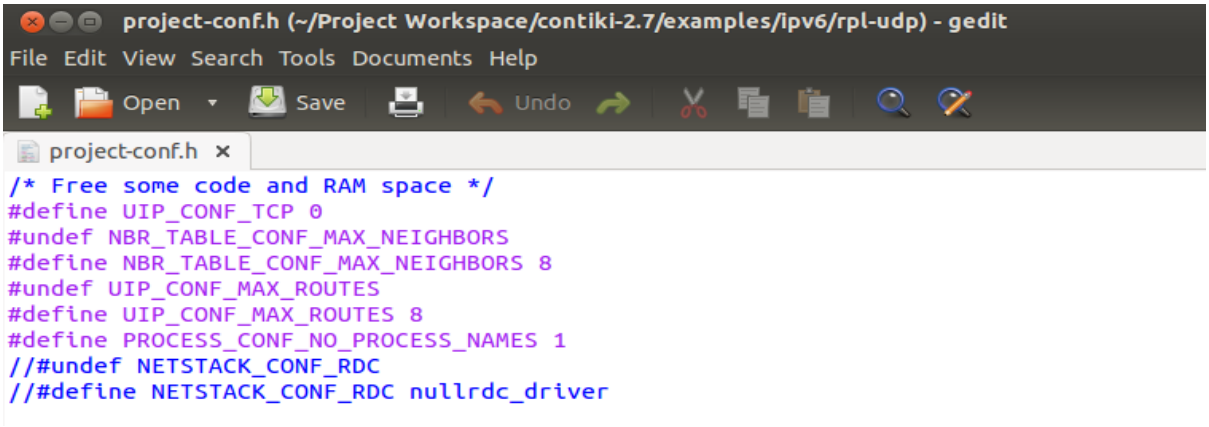

Figure 26 Project configuration file to save memory

4.4 COOJA Simulator

COOJA is a java based simulator. To start the simulator 'ant' command has to be run from the terminal by navigating to the Contiki (source path)/tools/cooja. When new simulation is created the window shown in figure 27 will be displayed. The radio medium can be used to change the medium as lossy or lossless. The default is Unit Disk Graph Medium (UDGM): Distance loss which denotes that the medium will have loss if the motes are far apart. There is also an option to include constant loss medium or multipath ray tracer medium. Change the simulation parameters as per requirements and click create button to create a simulation.

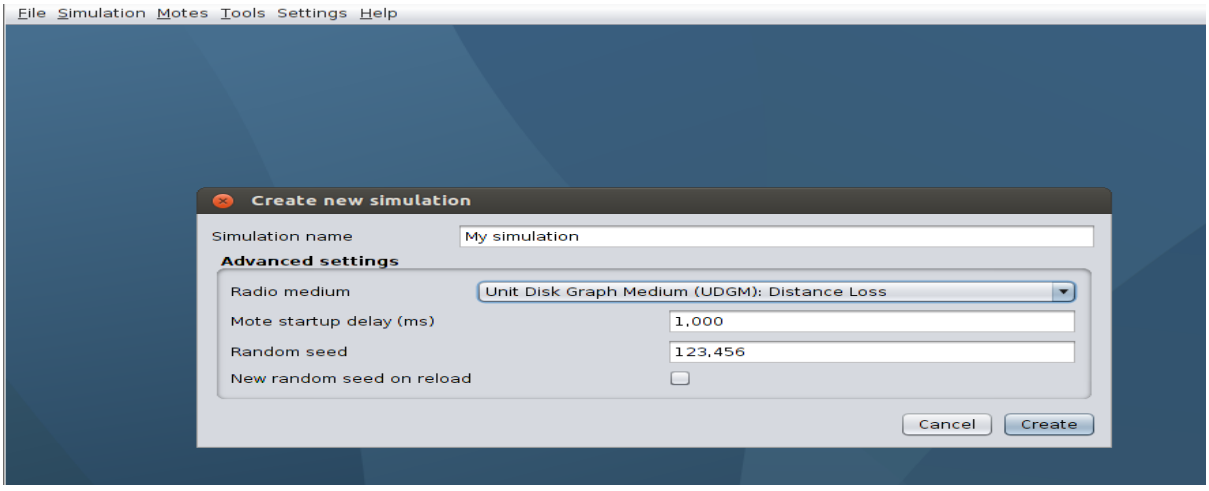

Figure 27 Creating new simulation in COOJA

When a new simulation is created the window shown in figure 28 appears. The figure shows multiple windows such as

- **Network:** An area used to create and position motes. The view menu in this window has options that can be used for better understanding of the motes being simulated.

- **Simulation control:** It is used to start, stop, pause or reload the simulation when relevant changes are made in the source code. The speed of the simulation can also be controlled in this window.
- **Mote output:** The motes running in the Network window will send outputs which are generally print statements given in the source code. These outputs are captured in the Mote output window.
- **Timeline:** This window displays the active time of each mote in which it sends or receives data.

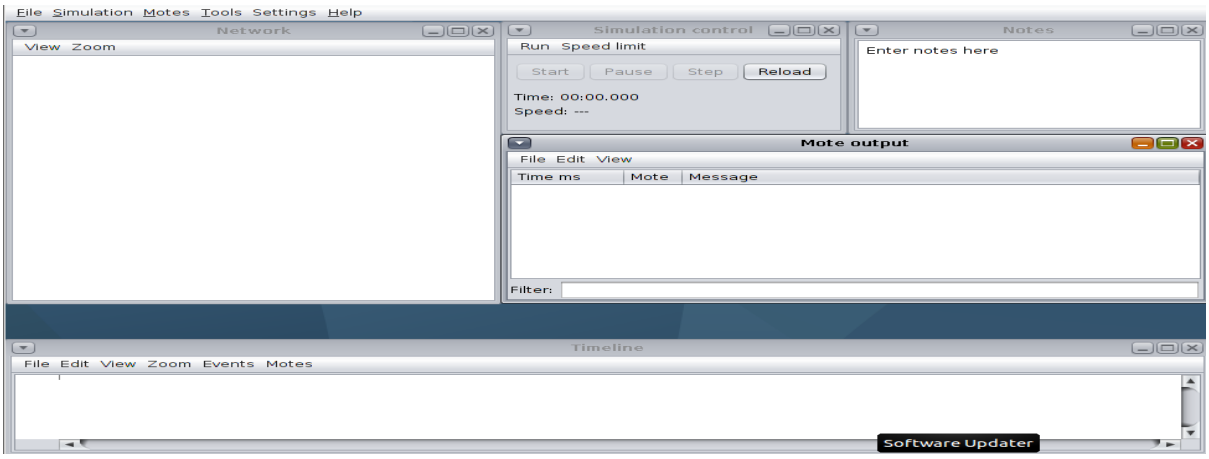

Figure 28 New Simulation opening page in COOJA

The motes can be added using the motes menu as shown in figure 29. The motes used in the simulated in sky mote. There are also other motes that can be used. A COOJA mote is also available to test any application specific implementations.

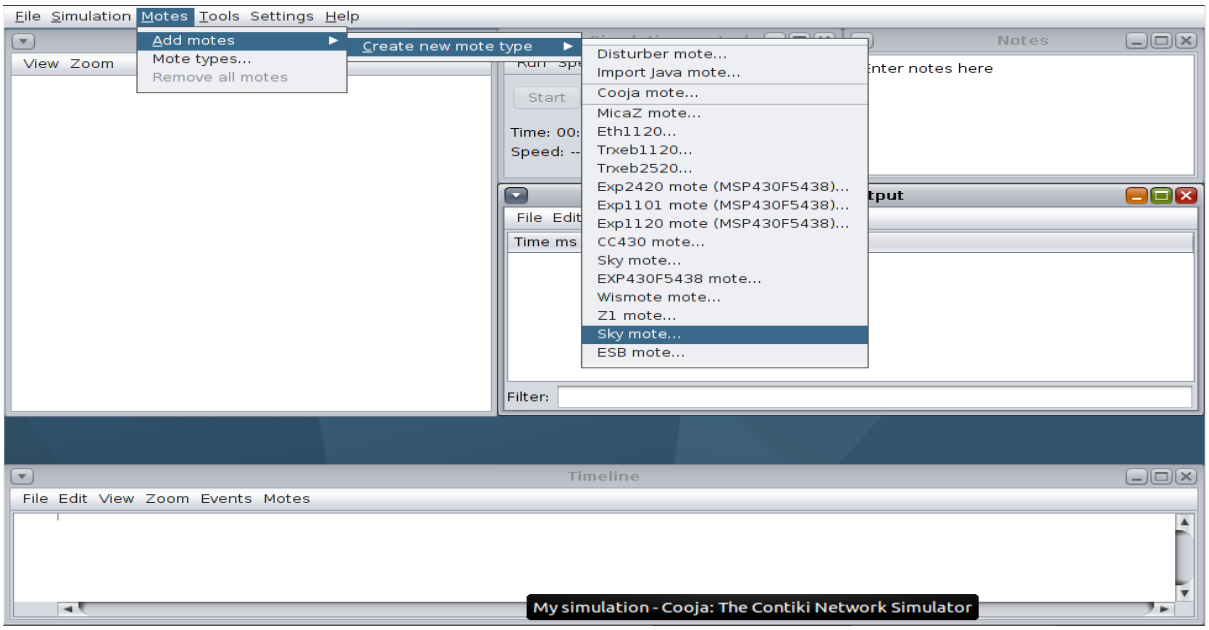

Figure 29 Creating a new mote in COOJA

When sky mote is selected the window similar to figure 30 appears. The required firmware can be selected using browse button. Then the firmware has to be compiled and create button has to be selected.

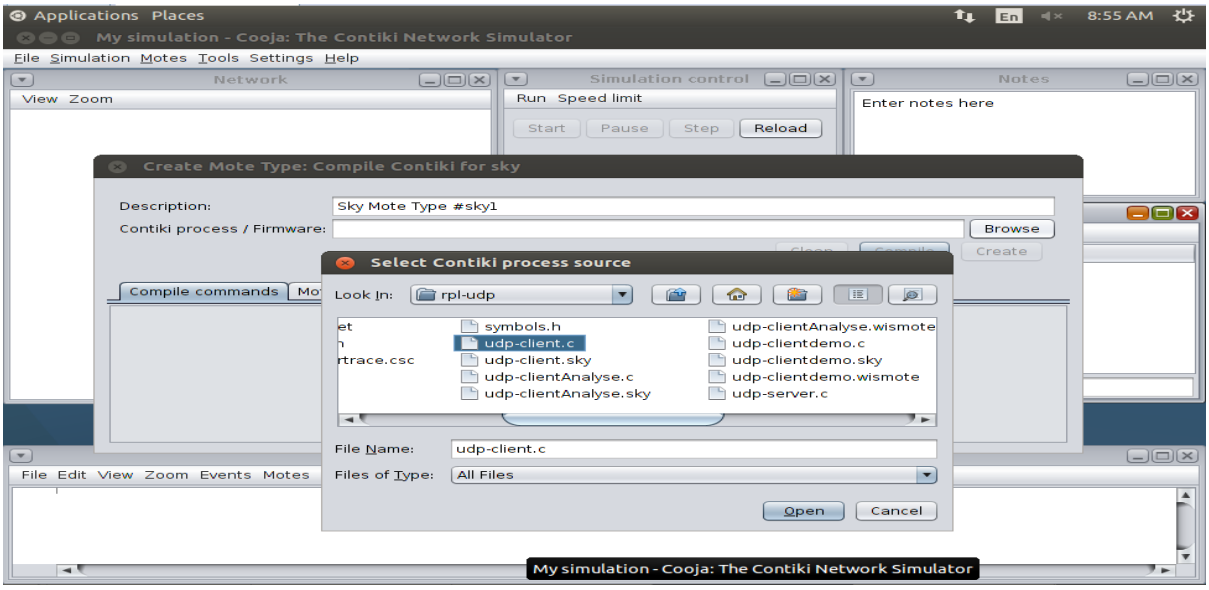

Figure 30 Choosing firmware for sky mote

Then the window shown in figure 31 appears. The number of motes and positions can be specified in this dialog box.

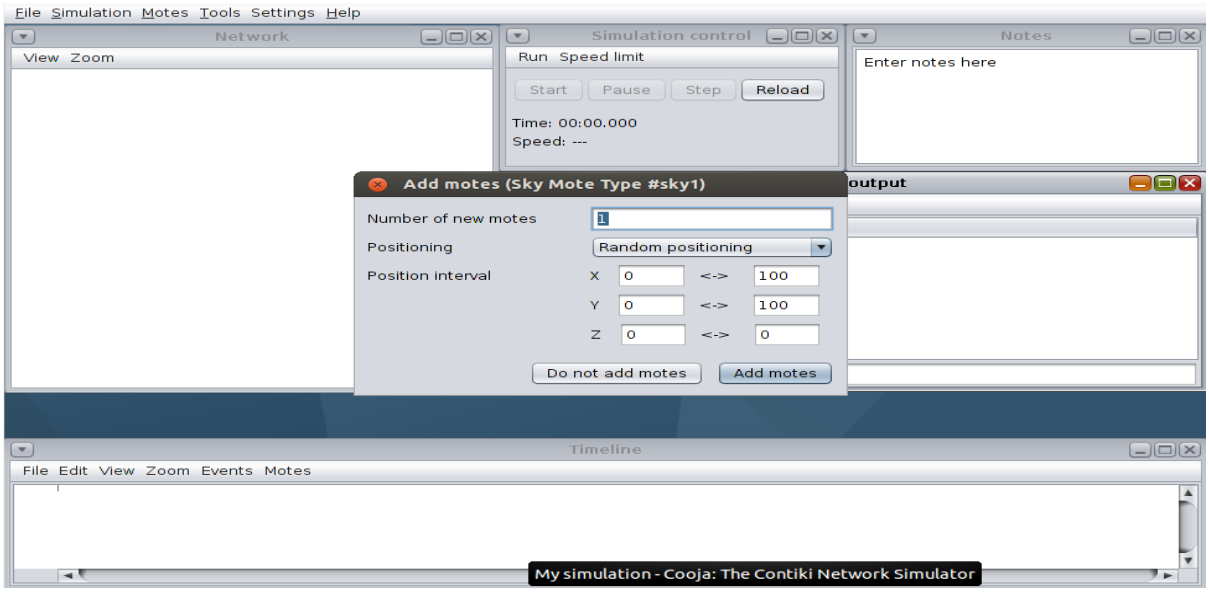

Figure 31 Number and positioning of motes

When the number of motes are 20 and they are added, the network window displays 20 motes as shown in figure 32. When one mote is clicked a green circle is formed around it to indicate range of communication of that mote.

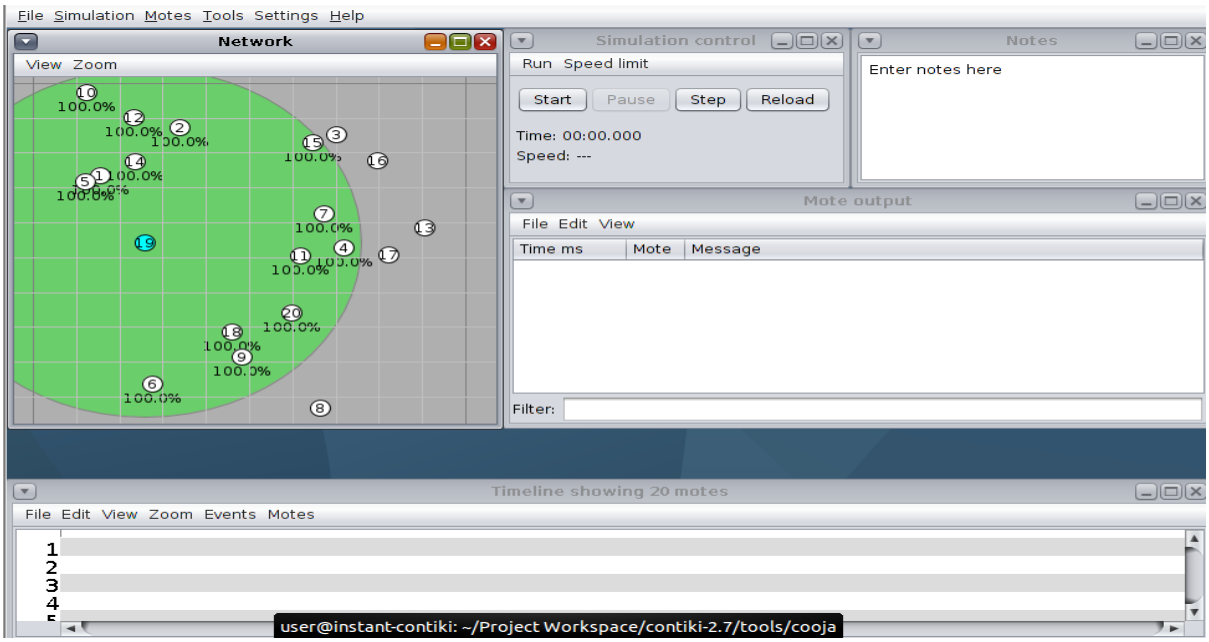

Figure 32 Demo setup consisting of 20 motes

Now start the simulation and it can be observed that the mote outputs are displayed in the relevant window with the time and mote ID in figure 33. This can be saved and later processed for required analysis.

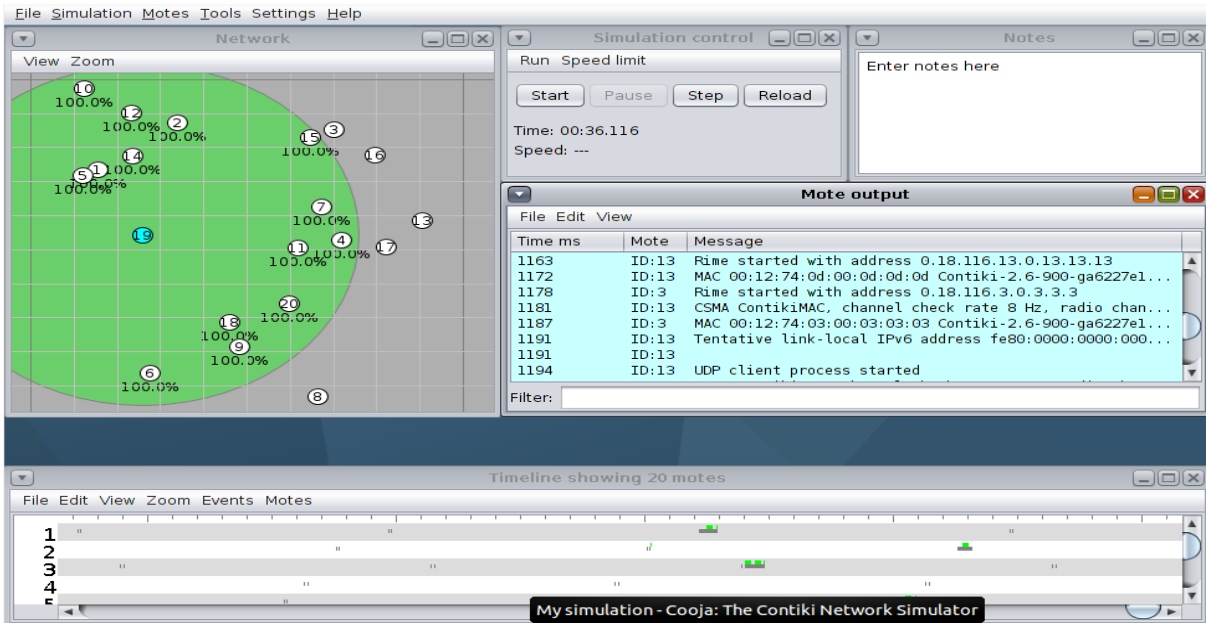

Figure 33 Mote output after simulation started

CHAPTER 5

RESULTS AND DISCUSSION

5.1 COOJA Setup

The figure shows the setup environment simulated for 50 motes. The mote output window displays the output of each mote along with its mote ID. The time is also displayed along with the corresponding output. This makes it very easy for analysis purposes.

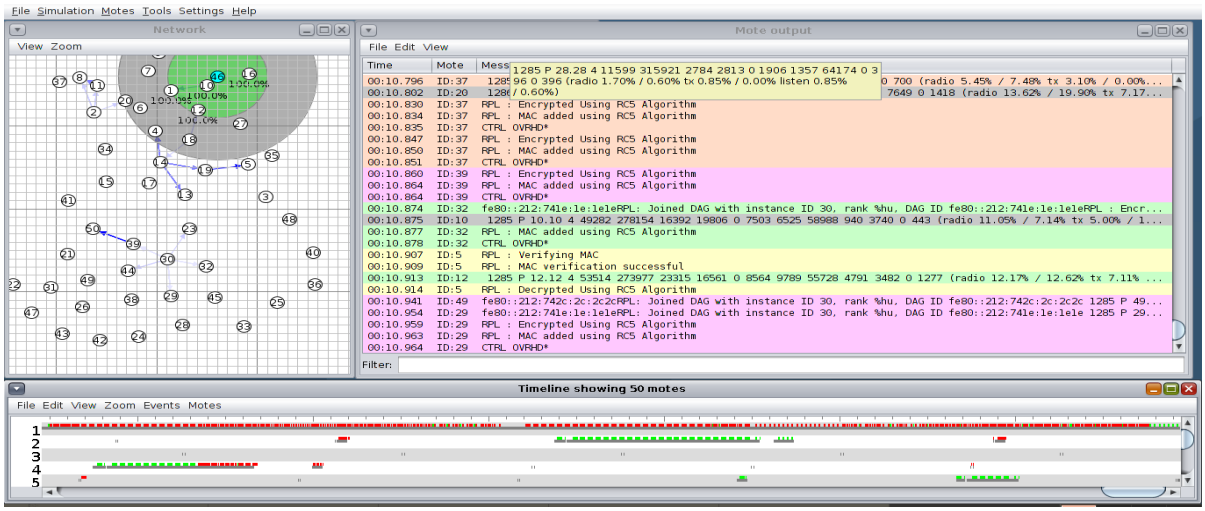

Figure 34 COOJA Screenshot

The mote output in the figure 34 shows the successful encryption and decryption of RC5 implementation. The message is encrypted during transmission and MAC is added. At reception MAC has to be verified and then messages have to be decrypted. The mote output also has a filter facility to display only required outputs. The pattern matching is done using any word that may come in all the required outputs.

Otherwise the simulation script editor in COOJA can be used to display, manipulate and process the mote outputs.

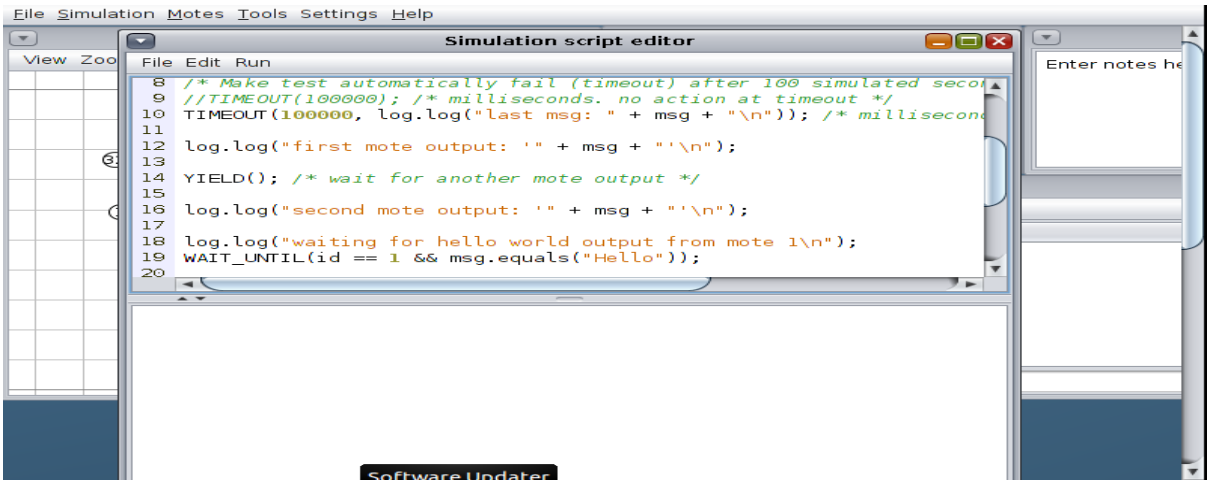

Figure 35 Simulation script editor

5.2 PERL Script

5.2.1 Network Convergence Time

The network convergence time is the time taken by all the motes to join a particular network. This can be calculated by obtaining the difference in time of first DIS message sent by any mote to the last mote joining the network. The time information can be obtained from the mote output in COOJA simulator. The outputs are stored as log files which are processed by the perl script, which is a well know scripting language for proceesing text files. The perl script is given below.

```
#!/usr/bin/perl

printf "\n\n RPL Metric - Network Convergence Extraction ====> \n";

# /*****************************************************************/

$directory_file =  "/home/user/Javid_Contiki_WorkSpace/Log_Ananlysis/New/";

unless ( opendir( DIR_HANDLE, $directory_file ) ) {

        die("The Directory file could not be opened \n ");

}

$results_file_name =

">>/home/user/Javid_Contiki_WorkSpace/Log_Ananlysis/AnalysisResults/results.log";

unless ( open( F_HANDLE_RESULTS, $results_file_name ) ) {

        die("The results log file could not be opened \n ");

}

# /*****************************************************************/

my $file_names;

while ( readdir(DIR_HANDLE) ) {

        $file_names = $_;

        #$file_names = readdir(DIR_HANDLE) || "";

        if ( ( $file_names ne "." ) && ( $file_names ne ".." ) ) {

                print $file_names;

                print "\n";

                netw_convergence( $directory_file . $file_names, F_HANDLE_RESULTS );

                }

}

close F_HANDLE_RESULTS;
```

```
# /*****************************************************************/
sub netw_convergence {
        my ( $debug_logfile, $results_handle ) = @_;
        $F_HANDLE1 = "0";    #Temporary initiazation
        #print "Debug file name $debug_logfile";
        unless ( open( F_HANDLE1, $debug_logfile ) ) {
                die("The log file could not be opened \n ");
        }
#/*****************************************************************/
        #Going to extract the lines with String "DIS Sent"
        #Going to extract the time information
        $handle_value        = "F_HANDLE1";
        $linein              = <$handle_value>;
        $first_time_DIS      = 0;
        $time_value_first_DIS = 0;
        @words               = ();
        while ( defined($linein) ) {
                #extraction line
                if ( $linein =~ m/\bDIS sent\b/ ) {
                        @words = split( /[\t ]+/, $linein );
                        # print @words;
                        # print $words[0];
                        if ( $first_time_DIS == 0 ) {
                                $time_value_first_DIS = $words[0];
                                $first_time_DIS      = 1;
                                print
"First DIS Match at time: $time_value_first_DIS \n";
                        }
                }    # end of if
                $linein = <$handle_value>;
```

```
        if ( defined($linein) ) {
                chomp($linein);
        }
    }    # end of while

    # /*****************************************************************/
    #Going to extract the last line with "DAG Joined"
    #Going to extract the time information
    $handle_value              = "F_HANDLE1";
    $time_value_last_DAG_JOINED = 0;
    seek $handle_value, 0, 0;
    $linein = <$handle_value>;
    @words  = ();
    while ( defined($linein) ) {

        #extraction line
        if ( $linein =~ m/\bJoined DAG\b/ ) {
                @words = split( /[\t ]+/, $linein );
                # print @words; print "\n";
                # print $words[0];
                $time_value_last_DAG_JOINED = $words[0];
        }    # end of if
        $linein = <$handle_value>;
        if ( defined($linein) ) {
                chomp($linein);
        }
    }    # end of while
    print "The LAST DAG Joined Time: $time_value_last_DAG_JOINED \n";
print "The  DAG Joined Time: $time_value_last_DAG_JOINED \n";
    $network_conv_time =
```

```
        ( $time_value_last_DAG_JOINED - $time_value_first_DIS ) / 1000;
        print "The Network Convergence $network_conv_time \n";
        print $results_handle (
"The Network Convergence Time for file $debug_logfile : $network_conv_time seconds\n "
        );
        close F_HANDLE1;
        # closing the log file after doing the text processing
}

# /*********************************************************************/
```

5.2.2 Power Consumption

The powertrace app given in Contiki can be used to obtain the time spent by a particular mote in CPU processing, LPM mode, transmission and reception. This information is stored in a log file and proceesed using perl script. The relevant time informations are taken from all the motes from parsing the log file and power consumed by each mote is calculated. Then finally average power consumed by all the motes in the network is calculated. The script used to calculate power in given below.

```
#!/usr/bin/perl
$voltage =3;
$power_cpu = 1.800 * $voltage;
$power_lpm = 0.0545 * $voltage;
$power_tx = 17.7 * $voltage;
$power_rx = 20.0 * $voltage;
for($i = 0; $i < 1000; $i++) {
   $max_radio[$i] = 0;
   $min_radio[$i] = 10000;
}
while(<>) {
   if(/P (\d+)\.\d+ (\d+) (\d+) (\d+) (\d+) (\d+) (\d+) (\d+) (\d+) (\d+) (\d+) (\d+) (\d+)/) {
      $node    = $1;
      $seq     = $2;
```

```
        $all_cpu     = $3;
        $all_lpm     = $4;
        $all_tx      = $5;
        $all_rx      = $6;
        $all_idle_tx = $7;
        $all_idle_rx = $8;
        $cpu     = $9;
        $lpm     = $10;
        $tx      = $11;
        $rx      = $12;
        $idle_tx = $13;
        $idle_rx = $14;
        $total_time = $all_cpu + $all_lpm;
         $nodes{$node} = 1;
         $radio_now = $tx + $rx;
        $idle_now  = $idle_tx + $idle_rx;
        $cpu_now = $lpm + $cpu;
        $dutycycle = $radio_now / $cpu_now;
        $dutycycle_for_node[$node][$seq] = $dutycycle;
        $idle_for_node[$node][$seq] = $idle_now / $cpu_now;
        if($seq > $max_seq) {
           $max_seq = $seq;
        }
    }
}
foreach $j (keys %nodes) {
    $avg = 0;
    for($i = 0; $i < $max_seq; $i++) {
        $avg += $dutycycle_for_node[$j][$i];
    }
```

```
    $idle_avg = 0;
    for($i = 0; $i < $max_seq; $i++) {
        $idle_avg += $idle_for_node[$j][$i];
    }
    print $avg / $max_seq . " " . $idle_avg / $max_seq . " $j\n";
    $total_avg += $avg;
    $total_idle += $idle_avg;
}
print "\n";
print STDERR "Idle percentage " . $total_idle / $total_avg . "\n";
 print STDERR "CPU Power" .$all_cpu*$power_cpu/$total_time. "\n";
 print STDERR "LPM Power" .$all_lpm*$power_lpm/$total_time. "\n";
 print STDERR "TX Power" .$all_tx*$power_tx/$total_time. "\n";
 print STDERR "RX Power" .$all_rx*$power_rx/$total_time. "\n";
 print STDERR "Total Power" .($all_cpu*$power_cpu + $all_lpm*$power_lpm + $all_tx*$power_tx + $all_rx*$power_rx)/$total_time. "\n";
```

5.3 Impact on Convergence Time

The simulation was carried out with variable number of nodes (20, 50 and 100) and the convergence time was estimated. It can be seen from figure 36 that the secured versions take more time for convergence compared to the unsecured version. The convergence time for RC5 is higher than that of SKIPJACK implementation. The convergence time vividly shows a foremost dependence on the distribution of motes. For a fixed Network setup, the convergence time is evaluated for the various security implementations under consideration. Modification in setup ends up exhibiting slightly varying results towards convergence of the network.

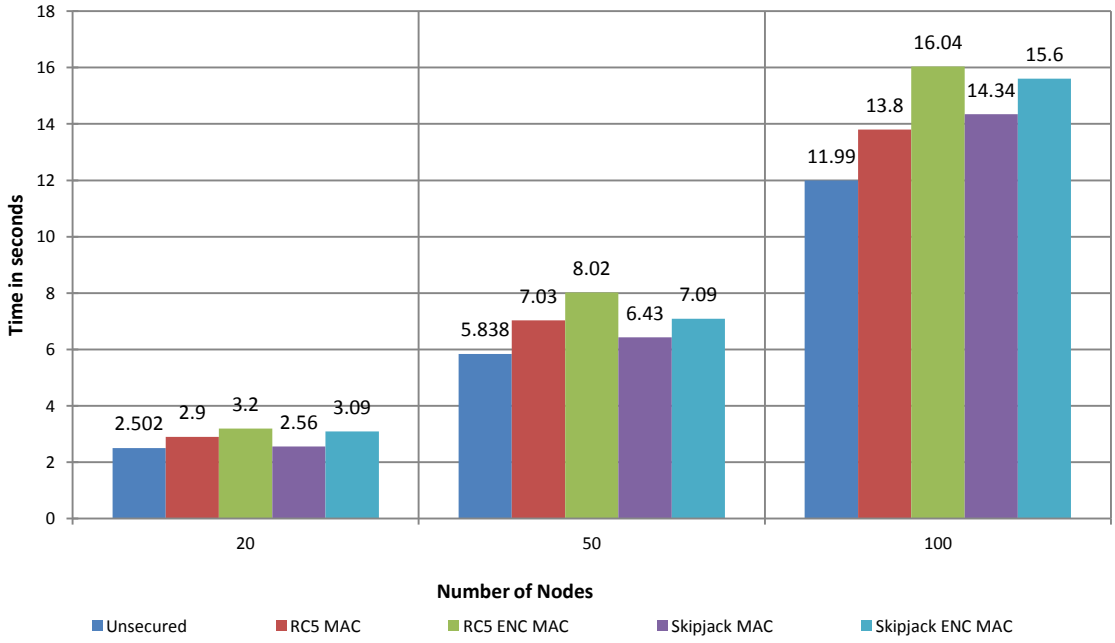

Figure 36 Convergence Time

5.4 Impact on Memory

In any wireless sensor network memory plays a vital role. When the memory of the mote is increased, the power consumption also increases. All microcontrollers generally posses two types of memory, ROM and RAM. The ROM size is very limited and it usually has the compiled codes stored in its memory. The RAM is a volatile memory and its size is even more limited than the ROM size.

The size can be found by using the size command followed by the file name. The file name usually ends with .sky (it varies depending on the platform). The resulting output of the size command is shown in the figures 37,38,39,40 and 41. The text column in the figures denotes the ROM size; the bss column denotes the RAM size; the dec and hex column represents the total size in decimal and hexadecimal formats.

Unsecured:

```
user@instant-contiki:~/Entire Backup/Javid_Contiki_WorkSpace/TESTVERSION/
   text    data     bss     dec     hex filename
  41061     188    5658   46907    b73b udp-clientAnalyse.sky
```

Figure 37 Unsecured UDP Application

RC5-MAC:

```
user@instant-contiki:~/Entire Backup/Javid_Contiki_WorkSpace/TESTVERSION/Javid_C
ontiki_WorkSpace/Deployment Version/contiki-RC5/examples/ipv6/rpl-udp$ size udp-
clientAnalyse.sky
   text    data     bss     dec     hex filename
  43707     188    5812   49707    c22b udp-clientAnalyse.sky
```

Figure 38 RC5 with only MAC

RC5-ENC-MAC:

```
user@instant-contiki:~/Entire Backup/Javid_Contiki_WorkSpace/TESTVERSION/Javid_Con
   text    data     bss     dec     hex filename
  44627     188    5812   50627    c5c3 udp-clientAnalyse.sky
```

Figure 39 RC5 with encryption and MAC

SKIPJACK-MAC:

```
user@instant-contiki:~/Entire Backup/Javid_Contiki_WorkSpace/TESTVERSION/
   text    data     bss     dec     hex filename
  45176     188    8296   53660    d19c udp-clientAnalyse.sky
```

Figure 40 Skipjack with only MAC

SKIPJACK-ENC-MAC:

```
user@instant-contiki:~/Entire Backup/Javid_Contiki_WorkSpace/TESTVERSION/
   text    data     bss     dec     hex filename
  47414     188    8296   55898    da5a udp-clientAnalyse.sky
```

Figure 41 Skipjack with encryption and MAC

In wireless networks, nodes may need to relay messages from others to reach their destination. With Contiki, even relay nodes, so-called routers, can be battery-operated because of ContikiMAC radio duty cycling mechanism which allows them to sleep between each relayed message. Contiki can be compiled even without Contiki-MAC. In order to illustrate the importance of the work, the graph describing the memory consumption with and without Contiki-MAC and RPL is shown in figure 42.

The figure 42 illustrates the memory constrained nature of the work and justifies the reason for choosing RC5 and SKIPJACK algorithms for providing security.

Implementing Contiki-MAC consumes a lot of memory, but reduces power consumption to approximately 1mW from around 60mW. The memory consumption with both RPL and Contiki-MAC consumes 41.6KB of ROM and thus leaving only 7KB for security and application oriented implementations. Different algorithms can be implemented for application layer security when contiki-MAC and RPL are not included. Since the current

work revolves around battery operated motes and securing RPL, both Contiki-MAC and RPL has to be implemented.

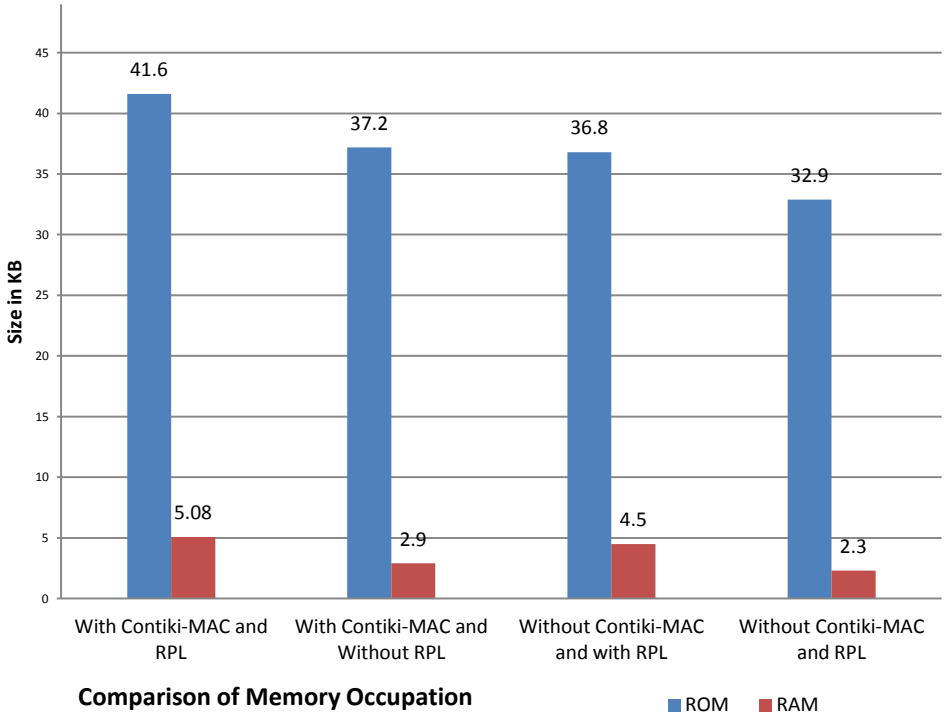

Figure 42 Memory Occupation

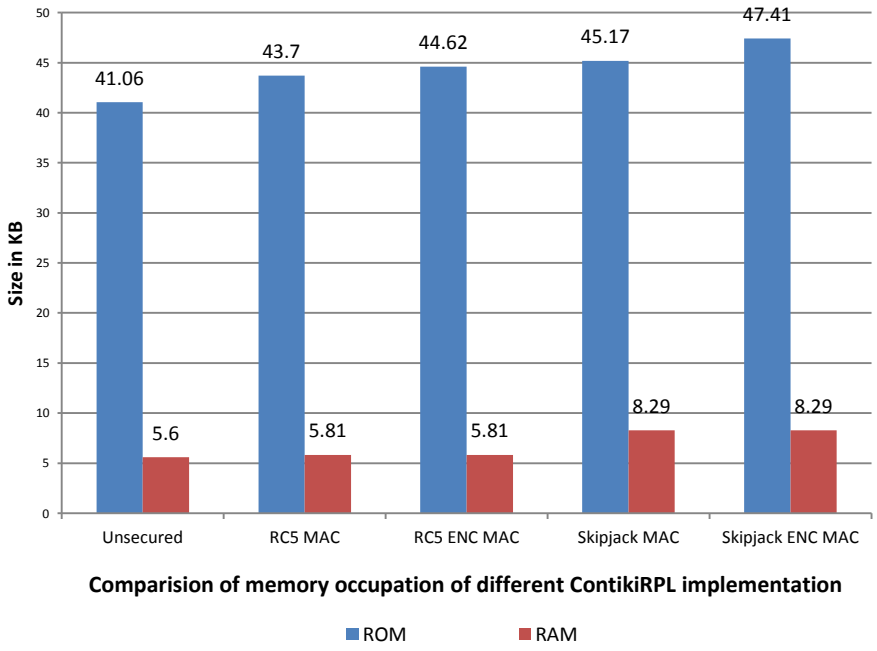

Figure 43 Memory Occupation

Based on the results from the size command the ROM and RAM consumption is charted in figure 43. The percentage increase in memory size, while implementing RC5 with only CBC-MAC is 4.5% for ROM and 13% for RAM. When confidentiality is entrenched with MAC, there is an increase of 9% ROM and 15.82% RAM size. Similar calculation for

SKIPJACK with only CBC-MAC yields 9.7% for ROM and 63% for RAM. With added confidentiality, there is an increase of 15% ROM and 63% RAM size. The increased memory consumption is calculated against the unsecured contikiRPL version. With respect to memory we can clearly see that RC5 is a better contender. Also SKIPJACK uses almost all the ROM size defined for Tmote sky, which abruptly shrinks the scope of implementation of any UDP socket application.

5.5 Impact on Power Consumption

Wireless Sensor Motes are mostly designed to be battery operated which limits their flexibility in power and reduces the life time of the network. COOJA uses an accurate power profiling tool [8] which is used to estimate the power consumption of motes in the simulated environment. The powertrace trace app provided by Contiki helps obtain the time spent by the mote in CPU processing, LPM, transmission and reception. With this time information the power is calculated. The comparison plot for power is depicted clearly in Fig. 44. Naturally the secure implementations consume more power than the unsecured implementation. When comparing the two algorithms, Skipjack appears to be more power efficient than RC5. Figuire 45 shows the pattern in power consumption levels after one hour observation of network communication. The total power consumption of each mote is categorized under four heads namely CPU, Low Power Mode (LPM), Transmission and reception power. It can be seen that a major variation in power results from the increased CPU and reception power. It is obvious from the fact that the confidentiality and authentication requires more CPU processing time.

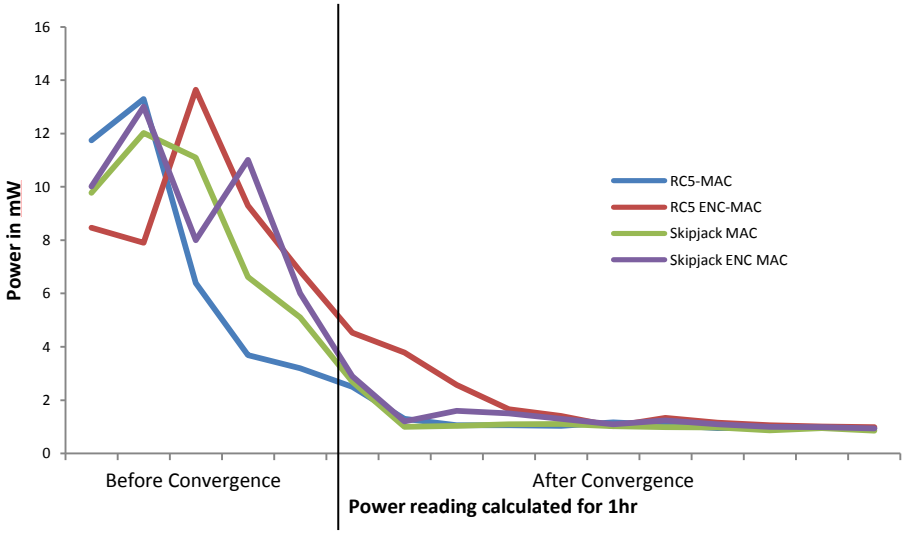

Figure 44 Power analyzed for 1 hour

From figure 44 it is evidently shown that the power consumption is high before the convergence of deployed network. High power consumption in the initial stage can be attributed to the mote startup and increased overhead to setup a network. After convergence, the power slowly stabilizes to values between 0.90 to 1.5 mV which creates a hope of building a secure ContikiRPL for eminent applications. The major difference is seen in reception power though encryption and authentication consumes some amount of extra CPU power.

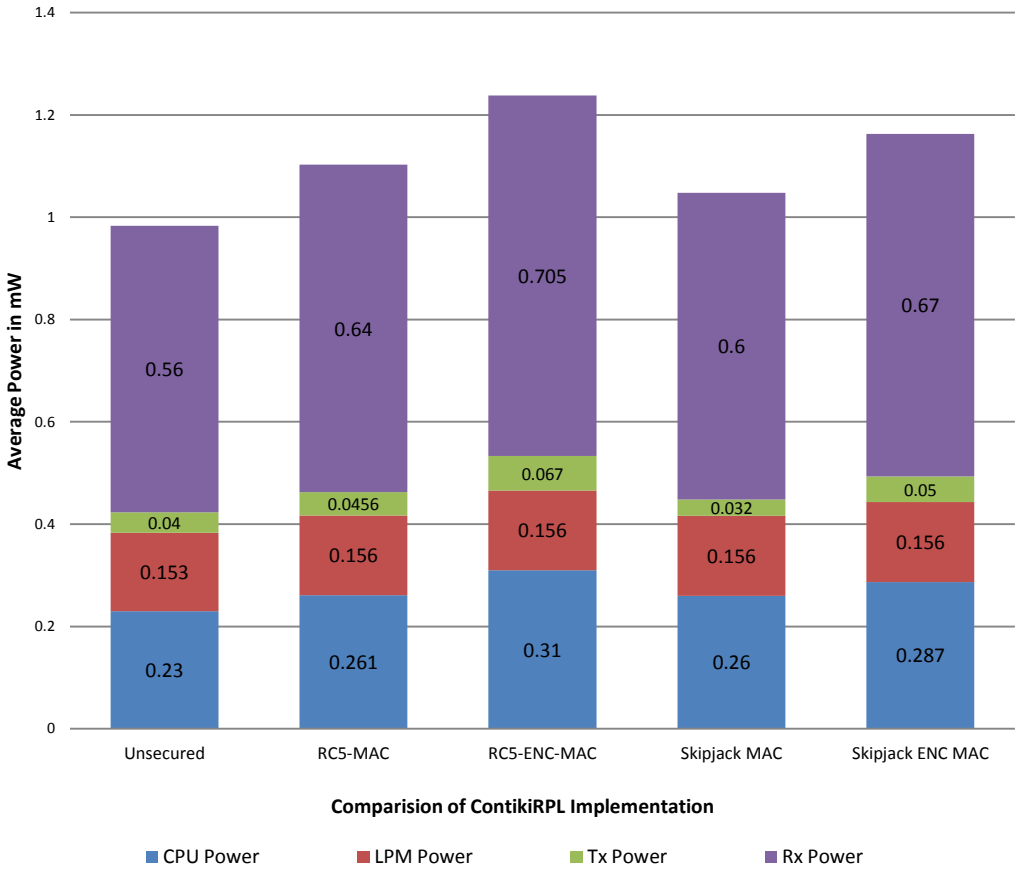

Figure 45 Power Comparison

There is not much change in the transmission and LPM power. When Contiki-MAC is not used the power consumption is close to 60mW. Thus we sacrifice some memory to drastically reduce the power consumed which is mandate for a resource constrained environment.

5.6 Increased security by varying key index

The motes in the same network can have different values for key index field shown in figure 8 such that the motes having the same key index use the same key. Some motes can have the privilege to support multiple key index values. The root mote will have to support all the key index supported by its child mote as shown in figure 17. Thus we can secure different routes with different keys since each mote will take as parents only those that can support its key index. This improves security and provides more privacy when required.

CHAPTER 6

CONCLUSION AND FUTURE WORK

The task of implementing secure ContikiRPL ready for portability into any hardware motes has been made successful using COOJA simulator with two cryptographic primitives combined with message authentication. The performance of secure ContikiRPL in terms of convergence, memory and power consumption profiling has been estimated with RC5 and Skipjack along with CBC-MAC and is compared against unsecured version of RPL. Based on the simulation results SKIPJACK shows better performance in terms of convergence time and power consumption. However SKIPJACK's memory occupation reaches the brim of ROM size which makes it unsuitable for any further implementation of real world applications. In spite of its convergence time and power consumption, RC5 occupies considerably less memory providing enough space confidentiality, authentication and further application oriented implementation. Hence it is concluded that RC5 with CBC-MAC could be the right choice for secure ContikiRPL implementation which can further be exploited for any real world application.

The border router can be established using raspberry pi or beagle-bone board by providing Tmote sky as a slip radio. The border router can be designed in raspberry pi by launching a web-server. Memory is not a constraint in this type of implementation. There is scope for different algorithms when motes with more memory such as wismote are used. As Contiki is becoming more popular everyday it is expected to be designed to be compatible with a variety of motes. In future a better key management mechanism along with standardized algorithms for encryption and message authentication can be implemented and studied.

REFERENCES

[1] C. Chang, D. J. Nagel, and S. Muftic, "*Balancing security and energy consumption in wireless sensor networks. Mobile Ad-Hoc and Sensor Networks*", 4864/2007:469–480, November 2007. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[2] C. Karlof, N. Sastry, and D. Wagner. TinySec: "*A link layer security architecture for wireless sensor networks*". In SenSys 2004, pp. 162–175.

[3] Dunkels, Adam, Joakim Eriksson, Niclas Finne, Fredrik Osterlind, Nicolas Tsiftes, Julien Abeillé, and Mathilde Durvy. "Low-Power IPv6 for the internet of things." In *Networked Sensing Systems (INSS), 2012 Ninth International Conference on*, IEEE, 2012, pp. 1-6.

[4] Dunkels, Adam, Bjorn Gronvall, and Thiemo Voigt. "Contiki-a lightweight and flexible operating system for tiny networked sensors." *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE, 2004.

[5] Farooq, Muhammad Omer, and Thomas Kunz. "Contiki-based IEEE 802.15. 4 node's throughput and wireless channel utilization analysis." *Wireless Days (WD), 2012 IFIP*. IEEE, 2012.

[6] Jongdeog Lee, Krasimira Kapitanova, and Sang H. Son, "*The Price of Security in Wireless Sensor Networks"*, Computer Networks Volume 54, Issue 17, 2010, pp.2967–2978.

[7] Jean-Philippe Vasseur and Adam Dunkels, "*Interconnecting Smart Objects with IP, The Next Internet*".

[8] Joakim Eriksson, Fredrik Osterlind, Niclas Finne, Adam Dunkels and Thiemo Voigt "*Accurate Power Profiling for Sensor Network Simulators*", IEEE 6th International Conference on Mobile Adhoc and Sensor Systems, October 2009**,** pp.1060 – 1061**.**

[9] JP Vasseur, Navneet Agarwal, Jonathan Hui,Zach Shelby, Paul Bertrand, Cedric Chauvenet, "*RPL The IP routing protocol designed for low power and lossy networks*", IPSO Alliance, April 2011.

[10] Ko, J., Eriksson, J., Tsiftes, N., Dawson-Haggerty, S., Terzis, A., Dunkels, A., & Culler, D. (2011, April). Contikirpl and tinyrpl: Happy together. In *Workshop on Extending the Internet to Low Power and Lossy Networks (IP+ SN)*, 2011.

[11] Linus Wallgren, Shahid Raza, and Thiemo Voigt, "*Routing Attacks and Countermeasures in the RPL-Based Internet of Things*," International Journal of Distributed Sensor Networks, vol. 2013, Article ID 794326, 11 pages, 2013.

[12] N. R. Potlapally, S. Ravi, A. Raghunathan et al. "*A study of the energy consumption characteristics of cryptographic algorithms and security protocols*", IEEE TMC, vol. 2. December 2005, pp.128–143.

[13] Olfa Gaddour and Anis Koubaa, "*RPL in a nutshell*: A survey", Computer Networks, vol. 56. 2012, pp.3163–3178.

[14] Tsiftes, Nicolas, Joakim Eriksson, and Adam Dunkels. "Low-power wireless IPv6 routing with ContikiRPL." In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, ACM, 2010, pp. 406-407.

[15] T. Winter, P. Thubert, A. Brandt et al. "*RPL: IPv6 Routing Protocol for Low power and Lossy Networks*", draft-ietf-roll-rpl-19 (work in progress), March 2011.

[16] T. Tsao, R. Alexander, M. Dohler et al. "*A Security Framework for Routing over Low Power and Lossy Networks*" draft-ietf-roll-security-framework-07, January 2012.

[17] T. Tsao, R. Alexander, M. Dohler et al. *"A Security Threat Analysis for Routing Protocol for Low-power and lossy networks (RPL)*", draft-ietf-roll-security-threats-10, September 2014.

[18] Y. W. Law, J. Doumen, and P. Hartel,"*Survey and benchmark of block ciphers for wireless sensor networks*", ACM Transactions on Sensor Networks, vol. 2, 2006, pp.65–93.