THREE PARTY AUTHENTICATION SCHEME FOR RFID SYSTEMS IN IoT

by

Virlla Devi Soothar

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
July 2017

This work is dedicated to

My beloved parents Dr. Bhamro and Devi Soothar for their trust and love,

My Supervisor Dr. Srinivas Sampalli for his support and encouragement through out the research,

My Sibblings Rohin Roy and Shirlla Devi who believed in me

And

To all my friends who stood by my side.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Radio Frequency Identification (RFID) systems have become a significant part of Internet of Things(IoT) ever since its emergence. RFID systems have been used for identifying objects, tracking assets and remote monitoring in various application areas of IoT such as healthcare, automated homes, and smart retail management. Before being deployed in IoT, traditional RFID systems were mainly used in supply chain management where the RFID reader and server had a secure connection. Hence the reader was always considered legitimate and there was no need of authentication between the reader and server. However, in IoT, the RFID reader can be any mobile device like a cell phone or a smart object which do not necessarily have a secure connection with the server. The primary objective of this thesis is to introduce a three-party authentication scheme in which the reader and server authenticate each other followed by the mutual authentication between the tag and server via the reader. For achieving this goal, we have used a lightweight hash function and a binary tree traversal based key derivation approach. We have also considered basic security goals for RFID systems in IoT in general. We have evaluated security of our authentication scheme using protocol analyzing tool 'Scyther' and randomness of our key generation scheme is tested on Statistical Test Suite(STS) by National Institute of Standards and Technology(NIST). Our test results illustrate that the proposed authentication scheme is secure and keys generated are random.

LIST OF ABBREVIATIONS USED

IoT	Internet of Things
RFID	Radio Frequency Identification
BLE	Bluetooth Low Energy
Wi-Fi	Wireless Fidelity
Li-Fi	Light Fidelity
EPC	Electronic Product Code
PRNG	Pseudo Random Number Generator
OTP	One Time Password
WSN	Wireless Sensor Network
CoAP	Constrained Application Protocol
MQTT	Message Queuing Telemetry Transport
XMPP	Extensible Messaging and Presence Protocol
AMQP	Advance Message Queuing Protocol
6LowPAN	Internet Protocol (IPv6) and Low-power Wireless Personal Area Networks
GE	Gate Equivalent
RDF	Resource Description Framework
OWL	Web Ontology Language
FPGA	Field Programmable Gate Array
STS	Statistical Test Suite

ACKNOWLEDGEMENTS

Foremost I would like to thank my supervisor Dr. Srinivas Sampalli for his continuous support and patience during my research. Without his help and guidance, this work would not have been completed.

I would also like to express my sincere gratitude to my Research Labmates: Saurabh Dey, Sridhar Matta and Trishla Shah who helped me in times when I was stuck and needed some guidance.

I am grateful to my family: my parent Dr. Bhamro and Devi Soothar for their spiritual support in all walks of my life, My siblings Rohin Roy and Shirlla Devi for their love and trust.

I would also like to thank my friends Harman Khara, Malik Nabeel, Laraib Shafi and Shahtaj Jawed who were always there for cheering me up and stood with me in my good and bad times.

Finally, I would like to appreciate all my great teachers and mentors throughout my academic career without whom I would not have achieved this milestone.

CHAPTER 1 INTRODUCTION

We are moving towards an era of the smart world where we don't explicitly have to perform certain tasks as per the current situation. Instead, the environment around us and things we own execute necessary tasks for us on their own. This has become possible with the advent of IoT. IoT is one of the most prevalent technologies that has potential to impact our lives. As the name suggests IoT is about making a network of things in such a way that things are capable to interact with each other using various communication mediums and technologies. One of the basic technologies being used in IoT is RFID. In IoT, RFID systems are not only used for identifying objects but have more critical uses like remote monitoring and tracking. With each communication technology, there are certain challenges associated with it which should be addressed to provide secure and reliable communication. This thesis is an effort to address issues related to the usage of RFID technology in IoT.

Before getting into further detail we first provide a brief description of terms used in the thesis followed by the proposed approach and outline of the thesis.

1.1. Brief introduction of terms and concepts

1.1.1. Internet of Things(IoT)

IoT is an evaluation of internet of humans in which not only humans but also things can communicate with each other. It is about connecting everyone and everything. IoT consists of sensor devices, objects, communication infrastructure, computational and processing unit. These elements of IoT that are stored on the cloud, decision making and action invoking systems [1]. Enabling technologies for IoT include Bluetooth Low Energy(BLE), RFID, Wi-Fi, Li-Fi, ZigBee, Z-Wave, Long Term Evaluation(LTE), IP6 over Low Power Wireless Personal Area Network(6LoPWAN) etc.

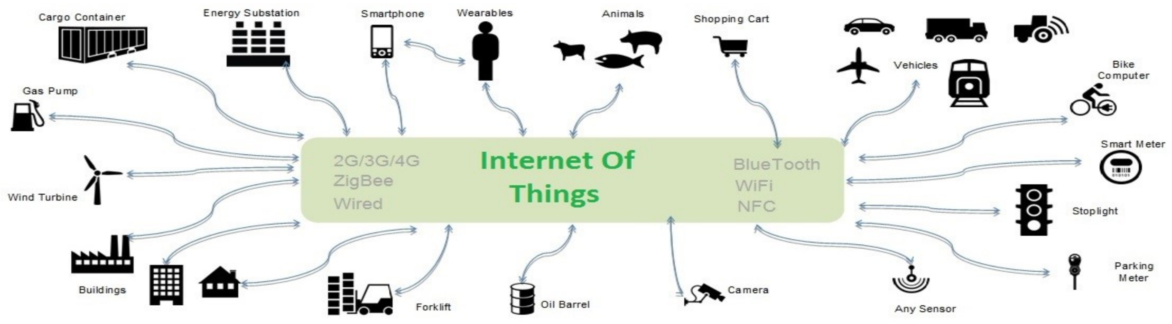


Figure 1 Basic IoT Structure [1]

1.1.2. RFID System

RFID is one of the enabling technologies for IoT. RFID systems consist of a RFID reader, backend server and RFID tag. It first appeared in 1945, as an espionage tool for the Soviet Union, which re-transmitted incident radio waves with audio information [2]. Similarly, the Identification Friend or Foe(IFF) transponder developed in the United Kingdom was used by the allies in World War II to identify aircraft as friend or foe.

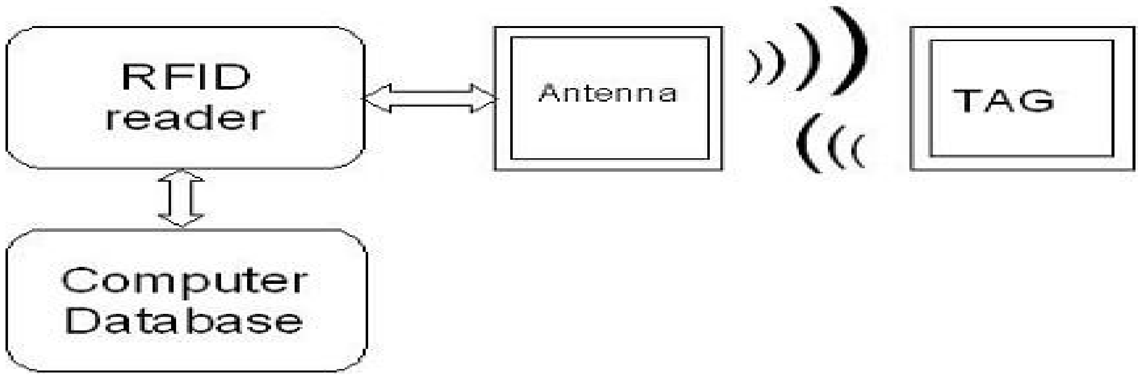


Figure 2 Basic architecture of RFID System [2]

1.1.3. Pseudo-Random Number Generator

All the EPCglobal Gen2 RFID tags are equipped with an on-chip 16-bit Pseudo-Random Number Generator (PRNG) which generates a sequence of random numbers. PRNG-generated sequence is not truly random because it is completely determined by a relatively small set of initial values, called seed values. PRNG plays a significant role in various security protocols for RFID systems because of its randomness and the limited computation capabilities of RFID tags.

1.2. Brief Introduction to proposed approach

This thesis focuses on improving the security of RFID systems in IoT. RFID technology is an automatic data capturing technology which uses radio frequency to identify objects. Previously its application range was limited to management systems to identify the product, object or person and some payment systems. However, with the emergence of the smart world and IoT, the application areas of RFID technology have been spread widely. Now it is considered as one of the strong candidates for automating environment. With the increase in the use of RFID technology, reliability and security of the communication have become one of hot research topics. There have been many authentication protocols proposed for the secure communication in RFID systems. However, most of the approaches were presented when RFID tags were purely used in supply chain management as replacement of the bar-code. So, these protocols assume that the communication medium between the server and reader is always secure. In IoT-based RFID systems such assumption is not always valid. In the smart world, every person has smart-phone or smart devices which also come with mobile RFID readers. These mobile readers can use various communication mediums to interact with the server which are not always secure. In this thesis, we have considered this factor and have proposed a three-party authentication protocol for RFID systems in IoT. In the proposed scheme, the reader is first authenticated by the server before it queries the tag. Once the reader is authenticated only then is the authentication between the tag and server carried out via the legitimate reader. In order to make the proposed scheme more secure binary tree traversal concept is used to derive random keys.

The proposed approach has three phases: registration, reader-server authentication and tag-server authentication via reader. In the first phase, the reader is registered at server's end and the encryption key is generated using the Diffie-Hellman Key Exchange approach. In the second phase, the reader and server authenticate each other after exchanging some information. Once the reader and server authenticate each other, the server generates a one-time-password(OTP). The OTP is associated with the reader and specific tag for the current session. The final phase is the tag-server authentication, where the tag and server authenticate each other. Since the tag and server have no direct connection, they communicate via an authenticated reader.

To authenticate each other in the 2nd and 3rd phases each party must generate a signature and send it to the other entity who verifies the signature. A new signature is generated and sent in each message to avoid the replay attack. In order to generate the signature both parties use the common information each of them already have. Once the signature is generated, it is sent to another party along with the additional information required to generate the next signature. The receiving party also generates a signature to verify the signature sent by the other entity. If both signatures match, then the signature and message both are considered as authentic.

In the proposed approach, we have used keyed hash functions to generate the signature. Due to limited computational capabilities of the RFID tags, we have used a lightweight hash function that is spongent hash. The key for keyed hash function is generated using our proposed binary tree traversal key derivation method. In binary tree traversal key derivation method, random values are generated using PRNG and are stored in form of tree data structure. Based on some random values and tree traversing techniques a new key is extracted each time for hashing.

1.3. Outline of Thesis

The organization of the rest of the thesis is as follows: an overview of the IoT, RFID technology, along with security goals and challenges are provided in Chapter 2. We have presented literature review of IoT, RFID Systems, Lightweight Hash Functions and Key Derivation schemes using Binary Tree in Chapter 3. Chapter 4 discusses the proposed three-party authentication scheme in detail which is followed by the explanation of the implementation in Chapter 5. We have presented the in-depth description of experiments we have conducted and evaluated results supporting our proposed approach are in Chapter 6. Chapter 7 discusses the conclusion of the thesis along with limitations and possible enhancements.

CHAPTER 2 BACKGROUND

In this chapter, we have presented a brief overview of terms and concepts that have been used throughout this thesis to have a better understanding of the proposed approach. Later we have also discussed security requirements and major security threats for RFID systems.

2.1. Internet of Things (IoT)

The term IoT was first coined by Kavin Austin from the MIT Auto-ID Center [3]. He used the term IoT to refer the binding of RFID information to the internet. After Austin's introduction to the idea of IoT, the concept of interconnected objects gained interest of IT companies and government organizations.

The term IoT is now days used widely in the era of the smart world. In the literature many definitions and concept can be found related to IoT. In [4] authors Huang and Li explain semantic meanings of IoT as: "the Internet relating to information of things". Authors of [5] define IoT as: "a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols". In [6] the concept of IoT is given a wide meaning; it describes IoT as a technology which enables physical objects to talk to each other, share information and knowledge with each other, think, see, hear and coordinate for decision making.

2.1.1. Enabling technologies for IoT

There have been various enabling technologies for IoT which have made the transition of traditional physical objects into smart objects possible. In this section, we have briefly discussed attributes of each technology such as wireless sensor networks(WSN), RFID, 6LoWPAN, ZigBee, Z-wave etc.

Wireless Sensor Networks (WSN): Wireless sensor networks consist of one or more base stations and tons of sensor nodes. Sensor nodes can sense physical information from the environment, communicate wirelessly, perform crude processing on information and report them to the base station [7]. Each node in WSN has various parts including a radio transceiver, an antenna, a microcontroller, an electronic circuit and a battery. Topologies used by the WSN varies from application to application; it can be a simple star network or an advance multi-hop mesh network.

Radio Frequency Identification(RFID): RFID is used to identify the objects without contact. It consists of a RFID tag, a RFID reader and a server. The RFID tag is a microchip that is attached to an object and has a unique identification code. The RFID reader can identify an object based on that code and get corresponding information regarding the tag from the server. RFID systems use radio signals over a short distance for data exchange between the reader and tag.

IPv6 over Low-power wireless personal area networks: One of the challenges in IoT is to integrate low-cost devices into the network and have a secure and reliable communication. 6LoWPAN [8] is one of the key components to achieve this goal. It has been designed by combining Low-power Wireless Personal Area Networks (LoWPAN) and IPv6 so that it can be used in resource-constrained and low-cost devices. In 6LoWPAN the packet size is smaller and it uses lower bandwidth. 6LoWPAN also provides connectivity and compatibility with the legacy architecture.

ZigBee: ZigBee is the wireless communication standard based on IEEE 802.15.4 for implementing low-cost, and short-range wireless networks [9]. ZigBee operates in the frequency band of 868 MHz, 915 MHz and 2.4 GHz. It is usually used in battery powered applications to achieve low energy consumption, low complexity, low data rate, security and reliability. It also supports multiple network topologies like star, mesh and tree topologies. Wireless devices that use ZigBee remain in the power-saving mode most of the time and become active only when they need to perform some tasks hence the battery lives of such devices are fairly long. [9]

Z-Wave: Like ZigBee, Z-wave is also used for short-term wireless communication because of having benefits like low cost and low energy consumption. It can have up to 232 nodes in the network, each of which is a slave node and is controlled using a controller. The frequency band in Z-wave is less than 1GHz (908.42MHz ~ 868.42MHz). [10]

Addressing scheme: Uniform Resource Name (URN) system is one of the fundamental aspects of IoT used to uniquely identify objects. One of the other options that are considered in IoT for addressing things is IPv6 which also allows accessing resources remotely and uniquely. However, considering memory and resources constraints of things (sensor devices and tags) in IoT, the IPv6 doesn't seem a good fit. To solve the addressing issue for resource constrained and low-cost devices IPv6 over Low-power Wireless Personal Area Networks (6LoWPAN) [8] is used.

Along with technologies mentioned above, there are some more technologies for IoT such as Bluetooth, BLE, Wi-Fi, Li-Fi, LTE, cellular, Thread, Near-Field Communication(NFC), SigFox, Neul and LoRaWAN.

2.1.2 Elements of IoT

Researchers in [11] have illustrated six main elements that are needed to deliver functionalities in IoT. These elements are as follows: Identification, Sensing, Communication, Computation, Services and Semantics. Below is the description of each element.

Identification: Identification is the basic element that allows naming objects uniquely so that they can be addressed in a network. In IoT ubiquitous codes (uCode) or Electronic Product Codes (EPC) are used to identify an object along with the IPv4, IPv6 and 6LoWPAN.

Sensing: IoT network contains smart objects that can gather data from surrounding and communicate with each other. Sensors enable the collection of data and transfer it to the data-centric units like the database or cloud from where it can be accessed. Sensors may include smart-sensors, actuators or wearable devices equipped with sensors.

Communication: This element includes all the communication technologies being used in a heterogeneous network of smart objects. IoT nodes have lossy and noisy communication links and use various communication technologies like WiFi, Bluetooth, IEEE 802.15.4, Z-wave, LTE, RFID and NFC.

Computation: Various hardware platforms such as Arduino, UDOO, FriendlyARM, Intel Galileo and Raspberry PI are available to run IoT applications. Along with hardware platforms, there are some software platforms such as Operating systems or Real Time Operating Systems that are required to provide IoT functionalities. Contiki, Tiny OS, LiteOS and RioT OS are some examples of operating systems for IoT-based devices. Cloud platforms allow IoT to store the data sent by smart objects and process it for extracting knowledge using big data concepts.

Services: IoT provides various services for identifying and addressing objects, collecting raw data from the real world, using data for decision making and ubiquitous services.

Semantics: Authors in [11] describes semantics as the brain of IoT. Semantics assists in providing required services like knowledge extraction, resource discovery, resource utilization, data analysis and decision making. For performing these tasks, it uses the Resource Description Framework (RDF) and Web Ontology Language (OWL).

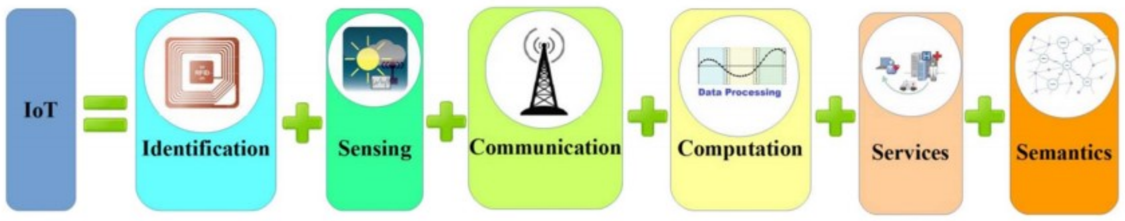


Figure 3 IoT Elements [11]

2.2. RFID Systems

When RFID was first developed as an identification system, it was aimed to replace the traditional bar-code system, but the development of the smart world and IoT has made it much more than just an identification scheme. RFID systems consist of a Radio Frequency (RF) tag (transponder), RF reader (interrogator) and server (Responder). These Radio frequencies have a wavelength value between 0.1 cm to 1000 km and frequency range varying from 30 Hz to 300 GHz [12].

2.2.1. RFID Tags

RFID tags are equipped with integrated circuits connected to an antenna [13]. Antennas in RFID tags collect and store energy to activate tags. Tags can perform various functions ranging from the simple read/write to the complex encryption and access control depending on its class. These tags are chipped into objects for identification, monitoring and tracking. The cost of a RFID tag depends upon the computational capabilities of the tag. Each tag has a unique identification number usually the electronic product code (EPC) based on which the object owning the tag is identified.

Types of RFID Tags:

RFID tags can be categorized into three main categories based on frequency ranges in which they can communicate with the reader [12]. Table 1 illustrates each type of tags along with their frequency ranges and application areas in which they can be used.

Type	**Frequency Range**	**Read Range**	**Standards**	**Application**
Low-Frequency RFID Tags	30 KHz to 300 KHz	Up to 10cm	ISO 14223 and ISO/IEC 18000-2	Access control Livestock tracking Identification
High-Frequency RFID Tags	3 MHz to 30 MHz	10cm to 1 m	ISO 15693 ECMA-340, ISO/IEC 18092 ISO/IEC 14443A ISO/IEC 14443	Electronic ticketing Electronic payment
Ultra-High Frequency RFID Tags	300 MHz and 3 GHz	12 m	EPCglobalGen2/ ISO 18000-6C	Inventory tracking

Table 1 Types of RFID Tags [12]

Tags can also be classified into three other categories based on how they communicate with the reader. That are as follows:

i) **Passive tags:** In passive RFID tags when a reader antenna sends a signal to the tag, the signal provides power to the tag. Using this power the tag wakes up, performs desired operations and reflects energy back to the reader. Passive RFID tags are cheaper and smaller in size.

ii) **Active Tags:** Active tags have a built-in power source (battery), often equipped with some sensors that can be used to measure and transmit environmental factors such as temperature or humidity. Read ranges of these types of tags are higher than that of passive tags, but they are more expensive and larger in size.

iii) **Semi-Passive Tags:** These are a hybrid type of tags that are mainly passive tags equipped with a battery. Batteries in tags make sure that they store maximum power received from the reader's signal and the read range of semi-passive RFID systems is better than passive RFID systems.

2.2.2. RFID Reader

The RFID reader which is also called the interrogator is the device that is used to read the information from RFID tags. Some readers not only just read the tag but also have the permission to access the memory of the tag and rewrite the content of the memory. A RFID reader can be either stationary or mobile depending on the type of application.

2.2.3 RFID Backend Server

The RFID Backend Server simply holds the information about all RFID tags within an application scenario. A RFID reader forwards an identification number to the backend server to get corresponding information after getting an identification number from the tag.

2.3 RFID Systems for IoT

RFID based IoT system contains three major parts [14]:

i) **RFID System**: It consists of one or more interrogators and RFID tags. The reader queries the tags to get an identification code that can be EPC.

ii) **Middleware:** EPC is sent to middleware system in IoT.

iii) **Internet System:** RFID system is connected to the internet using computer systems and network servers.

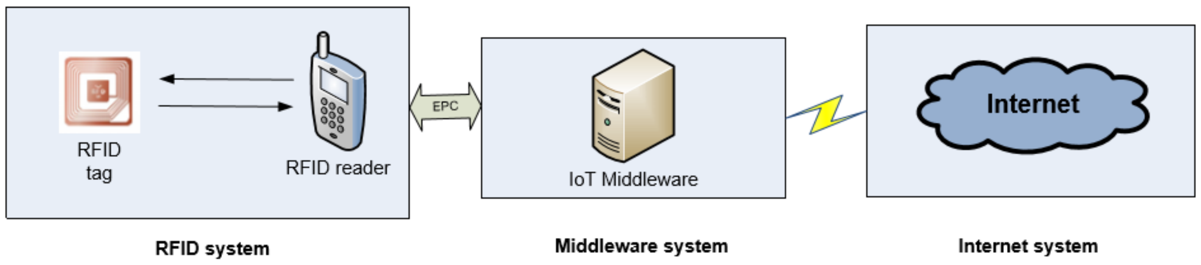


Figure 4 RFID-based IoT [14]

2.4. Computational capabilities and security risks

RFID technology has made the idea of automating the world possible. In the smart world, everything you own can be chipped with a RFID tag so that it can communicate with other objects in its surroundings and perform certain tasks as per scenarios and situations. One of the major issues with this idea is to make objects smart without influencing the cost of the object. In order to do so low-cost RFID tags are used. Class 0 and Class 1 tags are considered low-cost RFID tags.

As per EPCglobal standards, a passive tag (Class1 Gen2V2) can have up to 7-15K Gate Equivalent (GE) out of which only upto 2-5K [15] can be used for the security purpose.

These passive RFID tags have minimum computational power and cannot perform highly complex tasks, which raise security and privacy issues.

2.5. Security Requirements

Before proceeding further into details of the proposed approach and related work it is necessary to analyze security requirements of any RFID system. In this section, we would analyze the basic security needs of RFID systems.

Three main pillars of network security are confidentiality, integrity, and availability also known as CIA triad described as follow:

Confidentiality: Information should only be accessible to those who have authorized access. Any information given away by parties involved in the communication should not compromise user's privacy.

Integrity: Integrity can be described as the assurance that any message which is sent or received is not modified during the transmission. This can be achieved by using message authentication schemes such that the receiver can make sure that received message is originated by the sender.

Availability: It means how often the system is available to the desired user. One of the major threats to the availability of any system is Denial of Service (DoS) attacks.

Other major security goals that should be considered during the network security research are Authentication, Anonymity, Non-Repudiation, Forward Security, Backward Security, and Access Control. Authentication is the process of how an entity proves who it is as who it says it is, Anonymity means that the information about the sending party should remain anonymous to an eavesdropper who has been capturing all packets. Non-Repudiation means that the received message contains some attributes which prove that the message has been sent by the sender so that the sender cannot deny it later. Forward Security means that if a hacker or illegitimate person gets the knowledge of previously used keys in the communication it should not be able to infer keys that might be used for future communication. Backward Security implies that if an illegitimate user gets the current encryption keys it should not be able to deduce any previous key information from it. Access control is about who can access what and when.

2.6. Major Security Threats in RFID systems

Major Security threats faced by RFID systems, explained by authors of [17] are highlighted below:

Cloning: When a reader queries a RFID tag, the tag is supposed to return some information so that server can identify the tag. That information can be an identification number such as the electronic product code (EPC). One of the major issues with this is that an attacker can clone a tag by simply querying tag as a reader and save its EPC. Once an adversary gets the EPC it can act as a RFID tag to legitimate readers.

Tracking and privacy invasion: This thread is also related to the static EPC emitted by the RFID tag when it is queried by the RFID reader. Since EPC is a unique code and each tag has an EPC if the tag always replies with the same unique code than it is easy to track the tag (and user who owns the tag) with the help of fixed stationary reader that are located in different locations. This issue is not restricted to person's location tracking but it can also reveal certain information related to one's lifestyle and preferences. Lets say a person carries different DVDs tracking information about DVDs may reveal one's political or religious views which one might not be willing to share.

Eavesdropping and Replay Attack: An unauthorized reader can listen to the conversation among reader, server, and tag and sniff some information (messages) which it can later use to perform the replay attack.

Denial of Service(DoS) Attack: As the name suggests DoS attacks are carried out by interrupting a legitimate user(device) to access system which means it affects the availability of the system. In RFID system DoS can be carried out by simply cloning a large number of fake tags for the reader to read. Along with that jamming RFID reader signal can also prevent a legitimate tag to be read by the reader. Desynchronization attack is also a form of DoS attack in RFID scenario where the server is not able to identify a legitimate tag because of the corrupted database.

Conclusion

In this chapter, we have provided an overview of the basic terms and concepts important to understand the research work that has been done in this thesis. Along with that, we have also discussed major security challenges that need to be addressed followed by security goals that should be achieved to have a secure and reliable communication in a RFID system.

CHAPTER 3 RELATED WORK

In this chapter, we have presented the previous research work that has been done in the same stream as my thesis and has directly or indirectly influenced my research work. Literature survey of this thesis can be divided into four subparts that are IoT, RFID systems, lightweight hash functions and key generation protocols using tree data structure. This chapter is organized as follow: Section 3.1 throws light on the literature of Internet of Things followed by section 3.2 in which we have discussed research work that has been done in the stream of RFID Systems. Literature survey on lightweight hash functions is given in section 3.3 whereas section 3.4 presents the literature of the key establishment protocols using the tree data structure. In section 3.5 we have presented the motivation and research objective of my thesis.

3.1 Literature Survey on Internet of Things

IoT can transform traditional objects into smart objects by using technologies like ubiquitous computing, communication protocols, embedded devices etc. It can have a remarkable impact on various domains to improve the quality of our lives. There has been a great amount of research work done to adopt the idea of IoT in the practical world. In this section, we would have an overview of the architecture, requirement for the communication stack and protocols used in IoT.

3.1.1. The architecture of IoT

Various researchers and authors have proposed architectures [18], [19], [20], [21] for IoT. In this section, we have discussed these architectural designs.

Furness [18] proposed a five-layer architecture which allows objects to connect and exchange information among themselves. Tan and Wang [19] pointed out an underlying issue in Furness's architecture which is an exponential increase in network traffic and storage. They highlighted the fact that only objects that share similar application domain should communicate with each other to avoid network traffic and storage overhead. Authors also proposed an architecture where an IoT system consists of a backbone and several IoT applications as shown in Figure 5.

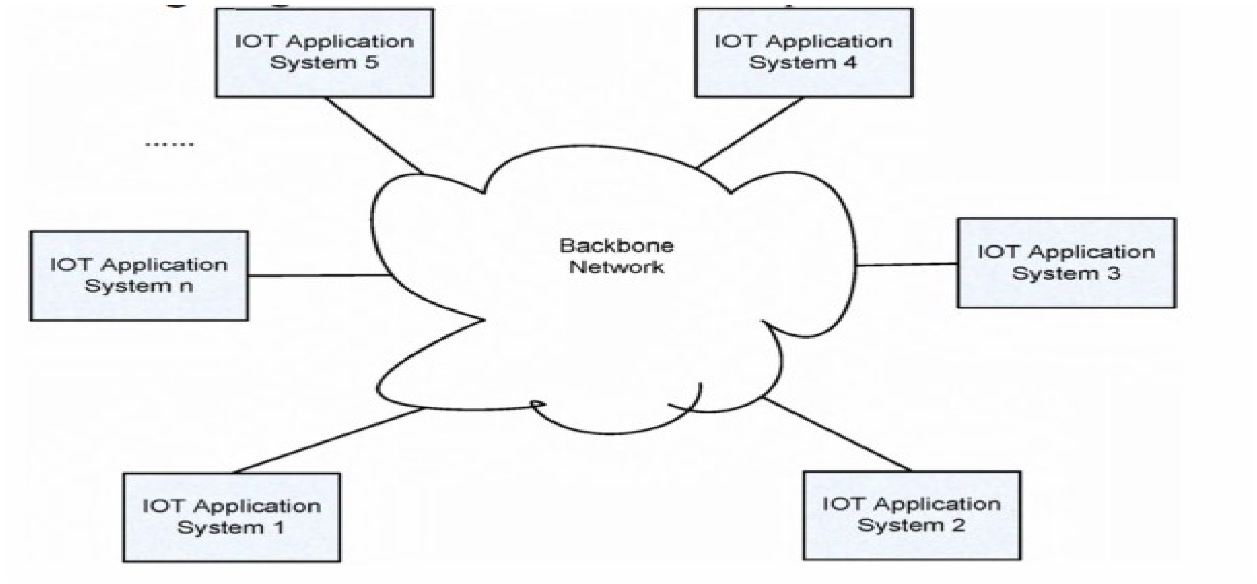


Figure 5 Internet of Things Applications and Backbone [19]

Each IoT application system works on their own but in the smart world, it is necessary to connect all application systems together. Various independent IoT applications may be using same or different communication standards. To enable the communication among independent IoT applications authors in [19] introduced a Co-ordination layer into the IoT architectural design. This layer is responsible for processing response packages from each application system. This layer also re-assembles response packages from various applications into a unified structure so that it can be processed by all application systems. Figure 6 shows the proposed architecture.

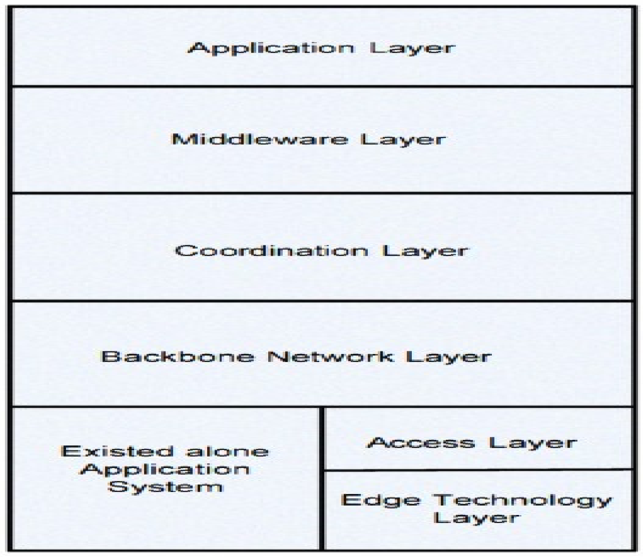


Figure 6 Tan and Wang 's IoT architecture [19]

Wu et al. [20] integrated architecture of the Internet (OSI model), the structure of Telecommunications Management Network and specific features of Internet of Things to introduce a new five-layer architecture for IoT. It is shown in Figure 7.

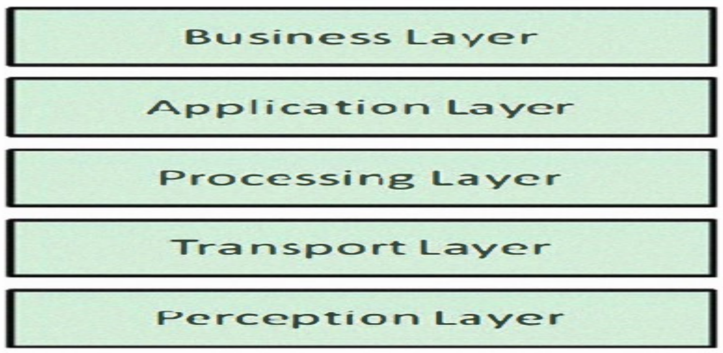


Figure 7 Architecture for IoT presented by Wu et al. [20]

Khan et al. [21] have summarized various architectures [18], [19], [20] into a general five-layer architecture. Below is the description of each layer along with the functionalities each layer provides.

- **Perception Layer:** It contains physical objects and sensors. The main task of this layer is to perceive physical objects by various sensors and transfer this information after converting it into digital signals. Technologies used in this layer includes RFID, 2-D barcodes, GPS etc.
- **Network Layer:** This layer is also called transport layer that is responsible for receiving data from the perception layer. It transmits the received data to the processing center via various networks that can be wired or wireless. Technologies used to perform these tasks include 3G, Wi-Fi, Bluetooth, ZigBee, infrared etc. Protocols like IPv4 or IPv6 are also used in this layer to address the objects.
- **Middleware Layer:** The main purpose of this layer is to store, analyze and process the data and information received from upper layers. It uses techniques like the database, intelligent processing, cloud computing etc. to provide services.
- **Application Layer:** Application layer is responsible for providing application-based functionalities. It uses the data processed by the middleware to provide services like logistic management, location based services, and safety. It plays a significant role in pushing IoT to a larger scale.
- **Business Layer:** This layer manages all the applications along with the business models. It is also responsible for performing in-depth analysis on the various

business models, construct flow charts and graphs to assist the business in decision making.

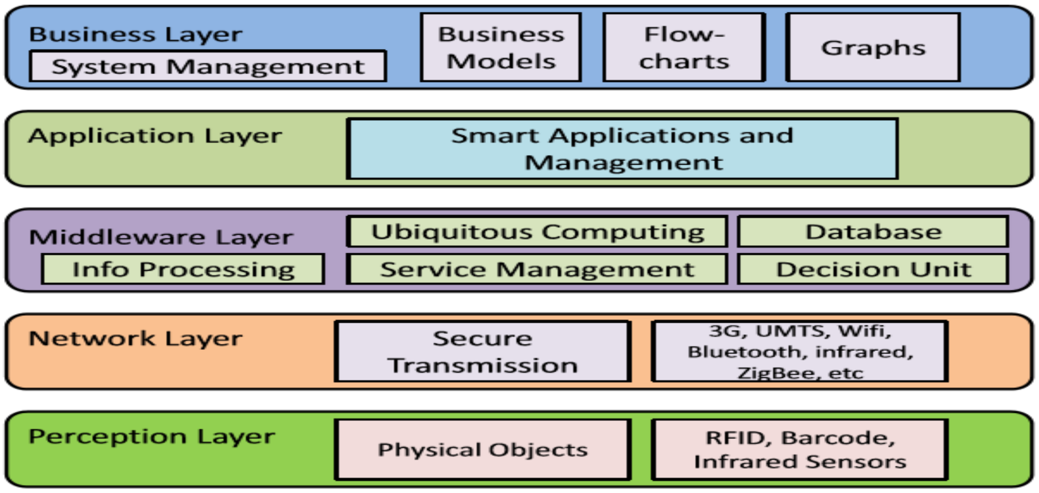


Figure 8 Generic Architecture of IoT [21]

3.1.2. Communication Stack and Standards for IoT

One of the major challenges in IoT is that the objects in IoT should have low power consumptions so that it can use the Internet while using power resources like batteries. Using protocols that are being used in the traditional internet can result in a waste of energy in the transmission of unnecessary data, protocol overheads and non-optimized communications [22]. For interconnecting objects and enable them to communicate, authors of [22] have enlisted three core requirements for the communication stack in IoT:

i) **A Low Power Communication Stack**: Most of the objects in the IoT gets their power from batteries hence such objects have a limited resource of power. This means it is difficult to communicate using a high-power communication stack as changing batteries of millions of objects on daily basis seems infeasible and expensive. Hence the communication stack and standards for IoT should be designed to meet this need.

ii) **A Highly Reliable Communication Stack:** A reliable end to end communication is an important element of all types of networking protocols. In IoT, the key challenge is to achieve the highest reliability in an efficient way.

iii) Internet-Enabled Communication Stack: Since the basic idea behind IoT is to allow the things to talk, hear and respond, which means a bidirectional communication medium is needed to facilitate it. One of the available medium is the internet that uses IP (Internet Protocol) to carry out the bidirectional communication. Since the standards and protocols being used in the traditional internet are not feasible for the constrained resource devices it is necessary for the IoT to have an IP enabled communication stack that satisfies the need of resource constrained devices.

Starting from 2003 standard governing organizations like World Wide Web Consortium (W3C), Internet Engineering Task Force (IETF), EPCglobal, Institute of Electrical and Electronics Engineers (IEEE) and the European Telecommunications Standards Institute (ETSI) have sat together to provide standards and protocols for IoT. These protocols can be categorized into application protocols, routing protocols, network layer protocols, physical layer protocols. Table 3 illustrates protocols in each category:

Category	**Protocols**
Application Protocols	Constrained Application Protocol (CoAP) [23] [24] Message Queue Telemetry Transport(MQTT) [25] Extensible Messaging and Presence Protocol (XMPP) [26] [27] Advanced Message Queuing Protocol (AMQP) [28] Data Distribution Service (DDS) [29]
Routing Protocols	Routing Protocol for Low Power and Lossy Networks(RPL) [30]
Network Protocols	6LowPAN [31] IPv4 IPv6
Link/Device Layer Protocols	IEEE 802.15. Bluetooth Low-Energy (BLE) EPCglobal
Service Discovery Protocols	DNS Service Discovery (DNS-SD) [32] Multicast DNS (mDNS) [33]

Table 2 Standards for IoT

3.2 Literature Survey on RFID Systems

RFID systems were first commercially used in the railroad industry for tracking and managing the assets like cargo containers and rail cars in the 1980s. Later it was adapted by supply chain management and retail industry as the replacement of the barcode. The RFID has an ease over the barcode since RFID tag does not require a direct line of sight to the reader. In RFID systems, multiple tags can be scanned simultaneously due to which it can operate efficiently in harsh environments like moist or dusty atmosphere, unlike barcodes. Ever since the RFID has been used in supply chain management great amount of research work has done in various areas of RFID technology that includes standard governing, collision detection and mutual authentication. Since our research work is related to the mutual authentication we have carried an in-depth literature review of the mutual authentication protocols that have been proposed for RFID systems along with the standards for RFID systems.

3.2.1. Standards for RFID Systems

Standard governing organization EPCglobal [34] is working on the development of standards for use of RFID technology in the paradigm of IoT. It introduced the use of Electronic Product Code (EPC) for the RFID tags and has also categorized the tags into various classes which are summarized in Table 2 [35]. EPCglobal has also released documents regarding specification of tags for each class, it has released the second generation of class 1(referred as EPCglobal Class 1 Gen2V2) tag along with its specification document [36] in which adoption of the cryptographic suite (optional) is made possible for the class 1 tags. According to that specification document, a RFID tag can generate a response of maximum 32 Kbits and can have a 16 bit RNG (PRNG) for inventory and password related operations.

Class	Type	Features
0-Read Only Tag	Passive	Read only Memory.
1-Identity Tag	Passive	Read only Memory.
2-Higher Functionality Tag	Passive	Read and write memory up to 64KB.
3-Semi-Passive Tags	Semi-Passive	Read and write up to 64KB. Built in battery to improve read range.
4-Active Tags	Active	Built in battery to improve read range. Allows tags to be networked with each other.
5-Active Tags	Active	Built in battery to improve read range. Allows tag to tag communication (class 4 and class 5 tags only).

Table 3 EPCglobal Tag classes

3.2.2. Mutual Authentication Protocols in RFID systems

There have been many papers that have tried to address the security concerned raised by low-cost RFID tags and have proposed various mutual authentication protocols to solve these issues. All the proposed protocols can be classified into various classes based on their computation cost. First class is "Full-Fledged Class" in which protocols that need the support of conventional cryptographic functions like symmetric key encryption, hash functions, and even public key algorithms are included. The second class is the "simple" class that includes those protocols which use random number generators and one-way hash functions on tags. Another class is “lightweight” and “ultralightweight” which includes protocols that only use bitwise operations along with the Random Number Generator (RNG) and Cyclic Redundancy Check (CRC) functions.

Further, in this section, we will discuss protocols in each class along with their limitations.

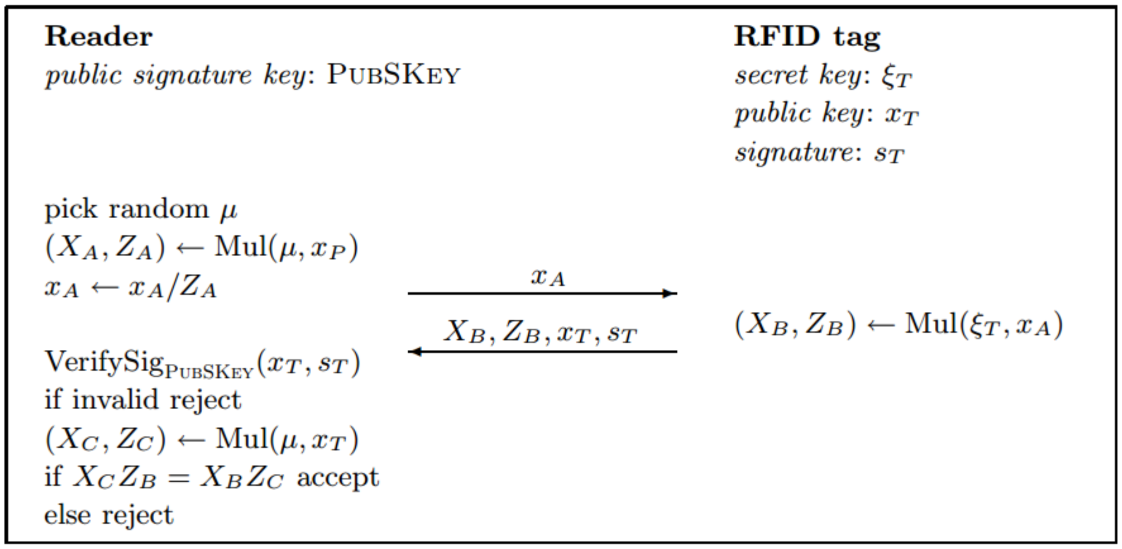
Full-Fledged Authentication protocols:

In this section, we will look into Full-Fledged Authentication protocols that have been proposed and how these protocols can be adopted in IoT and what are major risks associated with each protocol. Most of the protocols in this class are only feasible in RFID tags with high computational capabilities like active tags and are not feasible for low-cost passive RFID tags.

First protocol in this category was proposed by Kinoshita et al. [37] in 2005. The main focus of this paper was to eliminate threats related to the tag tracking and user privacy, in order to mitigate these risks an idea of the anonymous ID was proposed. Whenever a tag is queried by the reader it instead of replying with a fixed ID (usually EPC) tag outputs a temporary ID called Anonymous-ID which is different each time. Secure server uses this Anonymous-ID to identify the tag. Hence if eavesdropper captures the packet that is sent by a tag containing Anonymous-ID it is not able to identify the tag because of its randomness.

Bock et al. [38] proposed high-level tag authentication protocol which is a challenge-response procedure based on an Elliptic Curve Cryptosystem (ECC) and Diffie-Hellman key exchange. Tags using this protocol need to be initialized with a random private key 0

$< \xi T < q$ and a certificate (xT , sT) which contains its corresponding public key(xT) that is x-coordinate of point $T = \xi T \cdot P$ and also uses a signature generation algorithm. The reader also contains a public key(pubKey). Using ECC knowledge reader picks up a random ephemeral key $0 < \mu < q$, calculates the x-coordinate xA of the point $A = \mu \cdot P$ and sends xA to the tag. On receiving the challenge the tag computes (XB, ZB) $\leftarrow$ Mul(ξT, xA) which is x-coordinate XB/ZB of the point $B = \xi T \cdot (\mu \cdot P)$ and transmits it to the reader along with its signature. Reader verifies the signature and calculates the projective coordinates itself. If coordinates calculated by the reader are same as sent by the tag then the tag is accepted otherwise it is rejected. The major drawback of this approach is the assumption that the reader and server are same entity which does not seem right in IoT because of which this approach cannot be used in RFID-based IoT.



Reader
public signature key: PubSKey
RFID tag
secret key: ξ_T
public key: x_T
signature: s_T
pick random μ
$(X_A, Z_A) \leftarrow \mathrm{Mul}(\mu, x_P)$
$x_A \leftarrow x_A / Z_A$
x_A
X_B, Z_B, x_T, s_T
$(X_B, Z_B) \leftarrow \mathrm{Mul}(\xi_T, x_A)$
$\mathrm{VerifySig}_{\text{PubSKey}}(x_T, s_T)$
if invalid reject
$(X_C, Z_C) \leftarrow \mathrm{Mul}(\mu, x_T)$
if $X_C Z_B = X_B Z_C$ accept
else reject

Figure 9 ECC based high-level authentication protocol [38]

Liao and Hsiao [39] presented an ECC based scheme for mutual authentication of the tag and server via a reader in IoT. Zhao [40] proved how this approach is vulnerable to impersonate attack. Along with [38] and [39],there are other similar ECC based authentications schemes [41], [42], [43], [44] and [45] have been proposed to improve but most of are not able to withstand the replay attack.

Gross et al. [46] proposed an authenticated key exchange (AKE) protocol for RFID technology which made one of the initial steps to explain how RFID systems can be adopted in IoT. In this paper, authors focused on implementing IPsec on an EPC Gen2 Class1 tag to allow tags and the things associated with tags to become part of the internet of things. In order to use IPsec, it used the Advanced Encryption Standard (AES), and the Diffie-Hellman key exchange over an Elliptic Curve (ECP/ECCDH). The communication between an initiator sends an IP/UDP packet that contains IKE Security Association (SA) message to the reader. The reader upon reception of that message sends a challenge to tag which contains IKE SA_INIT message and a Diffie-Hellman value (ECC point), the initiator nonce and proposals for crypto suite. If any of the proposal in the crypto suite is acceptable for the tag it creates an initial Security Association (SA). Tag also generates its own Diffie-Hellman value, nonce and key material (seven keys) from received Diffie-Hellman value and the nonce. The first key from the derived seven keys is used to create new key material for future child SAs and moreover for each direction of communication a symmetric encryption and decryption key is generated.

Tag in response replies with accepted SA proposal and generated Diffie-Hellman value and the nonce. Tag also calculates an Auth-Value by signing IKE_SA_INIT response, initiator nonce and a unique ID (IPv6 address of tag) and stores it into ResponseBuffer. The reader either receives challenge response along with the EPC or ResponseBuffer of the tag is read by a reader which further encapsulate the response into an IP/UDP packet and forward it to the initiator who performs same calculation to obtain SA. For further steps, IKE_Auth payload is used along with SA to perform authentication between the tag and initiator. After that authentication key updating step is used to update keys and create child SA.

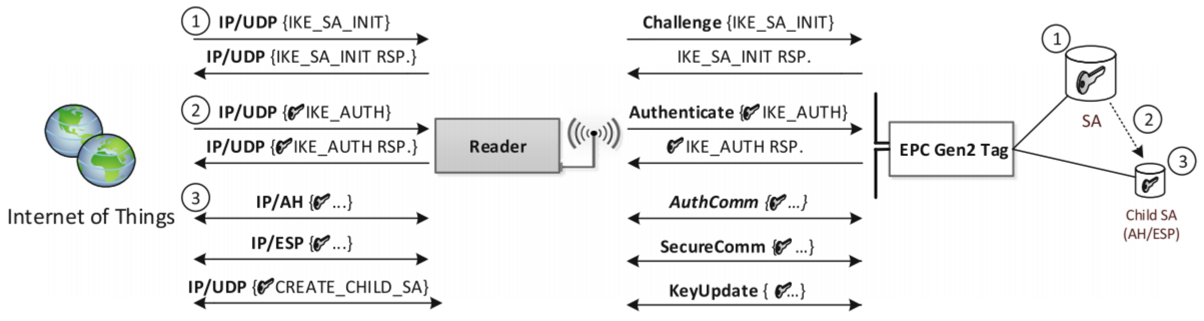


Figure 10 IPv6 based RFID Authentication [46]

This approach seems a perfect fit in terms of using RFID systems in IoT but considering the computation cost used in this protocol makes it infeasible for low-cost RFID tags. Although authors claim this protocols to be fully complimented with RFID EPC Gen2 Class1 tags but it requires about 52k GE along with all components to establish a secure IPsec tunnel between an Internet client and a RFID tag which makes it more complex. Another similar protocol that tried to use IPsec type key exchange scheme for the mutual authentication was presented in [47] but major flaw of that scheme is the assumption that reader and backend server are the same entity.

Zhu [48] authors proposed Authentic Key Exchange based mobile RFID (AKE-MRFID) protocol which considers the fact that that there can be multiple readers that can be dishonest or compromised. The backend server generates a pair of public and private key (PKDB, SKDB) that is used for the encryption and decryption. Each reader has an identification RID and a secret key (Kr). Each tag is also associated with an identification and secret key TID and Kt respectively. This scheme focuses on the access-type-based authentication of the reader. A reader sends a hello to the tag with a random value Nr, in response tag replies with a random value Nti, TIDi and C1 = Hash (TIDi, Nt, Nr, Kti). The reader appends this message with the Nr, Access-type, and RID and sends it to the server. The server checks for the tag and look into the access list for the tag and authenticates the reader only if the access-type it provides is correct. If the reader is authenticated, server computes the session key sk = Hash (NR, NT , KTi), C3 = Hash(TIDi, KTi) $\oplus$ TID_{new} I, C4 = Hash (TID_{new} i , sk) , C5 = EKRj (RIDj, NR, TIDi, sk, C3, C4) and sends C5 to the reader. Reader upon reception of the C5 decrypts it and extracts the information to authenticate the server if TID and Nr are correct. It forwards the C3 and C4 to the tag if the server is authenticated. The tag computes Hash (TID_{new} i, sk) to verify C4. If the equality holds, then it accepts the reader, the back-end system, and the session key, and then updates TIDi and KTi with TIDnew i and K new Ti = Hash (NT, KTi) respectively. Authors of [49] have illustrated the scenarios in which various attacks like replay attack, desynchronization attack, and reader impersonation attack can be launched on this scheme.

Simple Authentication Protocols

There is a wide range of protocols that have been proposed which can be categorized as simple authentication protocols because they do not use complex cryptographic functions like ECC or public key infrastructure instead they rely on the random number generator (RNG) and one-way hash functions to carry out authentication among communicating parties. Further, in this section, we will look into these protocols and their limitations.

Hash-based authentication protocols are popular because of their one-way nature. One of the earliest hash-based schemes for RFID system was proposed in [50] and [51]. Weis et al. [50] presented a protocol in which when a tag is queried by a reader it replies with a message (r, hash (ID, r)), where r is a random number. Upon receiving tag's response, the reader that contains a list of IDs, calculates hash of r with each of the ID in the list to find out the tag. Weis et al. [50] also presented another version of the protocol where each tag has a shared key for the communication with the server (reader) assuming that reader and server are one entity or reader is always legitimate. Chien's proposed approach in [51] tried to extend the approach used in [50] by introducing one master key for backend server to reduce the load on the server and tag response involves the current date stamp. The major drawback in Chein's approach is the lack of mutual authentication among the reader and tag along with the assumption of reader always being an honest participant. Another assumption made in this approach is that a passive tag can generate date stamp which seems infeasible to the date as a RFID tag do not have the functionality to compute current date or time.

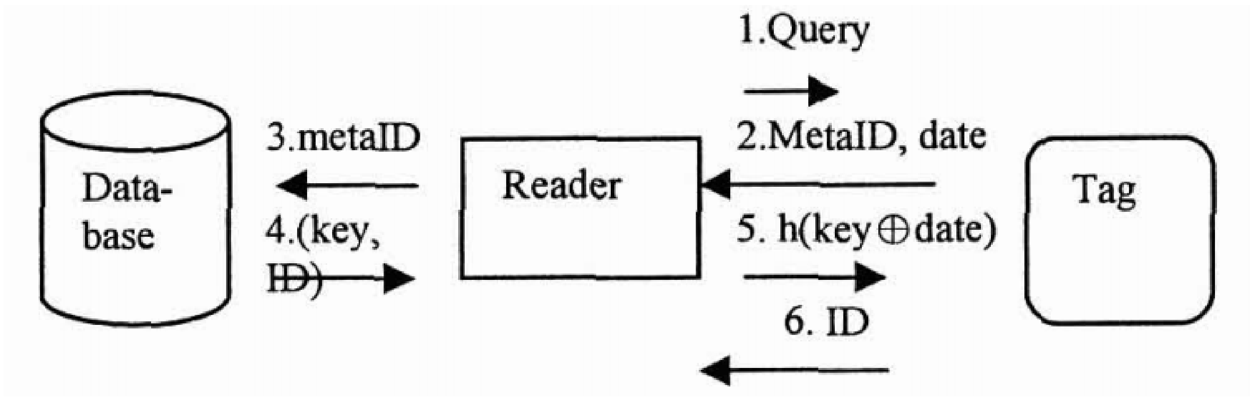
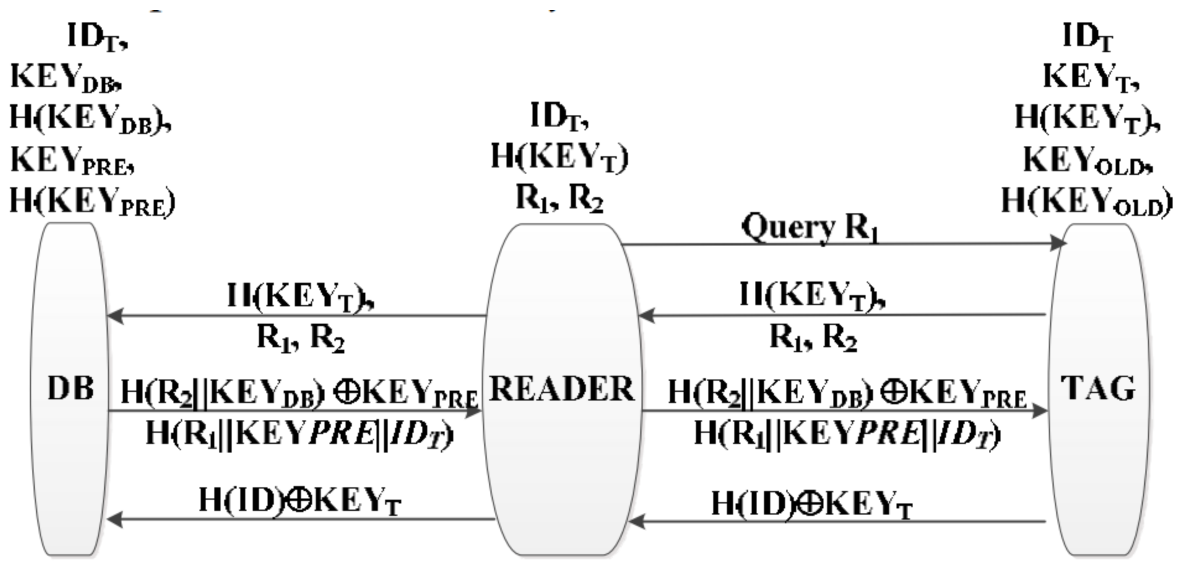


Figure 11 Hash-based scheme for RFID system [51]

In [52] both the tag and the backend server use Keccak algorithm to perform hash. The RFID tag contains IDT and an authentication key(KeyT) along with the PRNG. The backend server stores the ID, authentication key along with the hash of the authentication key for all tags. The basic flow of the proposed protocol is explained in Figure 12. The reader initiates the communication by querying tag with a random number R1. The tag upon reception of the query generates a random number R2 and performs a hash of KeyT and sends it to the reader along with random values R1 and R2. The reader forwards the message to the server and server checks for the tag, if the tag is found hash values of KeyT (KeyDB) and keyPre are stored along with keys. The tag gets KeyPre by computing Hash(R2||KEYDB), KEYPRE, Hash(R2||KEYT) and verifies it and update its keys.



ID_T, KEY_DB, H(KEY_DB), KEY_PRE, H(KEY_PRE)
ID_T, H(KEY_T) R_1, R_2
ID_T KEY_T, H(KEY_T), KEY_OLD, H(KEY_OLD)
Query R_1
H(KEY_T), R_1, R_2
H(KEY_T), R_1, R_2
DB
READER
TAG
H(R_2||KEY_DB) ⊕KEY_PRE H(R_1||KEYPRE||ID_T)
H(R_2||KEY_DB) ⊕KEY_PRE H(R_1||KEYPRE||ID_T)
H(ID)⊕KEY_T
H(ID)⊕KEY_T

Figure 12 Mutual authentication protocol based on Keccak Algorithm

[52]

Other hash based mutual authentication protocols were proposed by Dehkordi and Farzaneh [53], Khedr [54]and Habibi [55]. Each of these protocols are subjected to traceability attacks [56]. Authors of [56] have performed in-depth traceability analysis on these three protocols which indicates weaknesses of each of the above approaches. Successful forward and backward traceability attacks were made on Dehkordi's protocol whereas Khedr and Habibi's approaches proved good against backward traceability attack but failed against forward traceability attack.

Cho et al. [57] also presented a hash based authentication scheme and claim it to be secure against all types of privacy and security issues. For each session in this authentication scheme, both the reader and tag use nonce values R_r and R_t respectively and two fresh values are generated from each of the nonces that are RID_{ti} and RID_{ri}. For authentication tag generates α = hash ($ID_k \oplus R_t \oplus R \oplus RID_{ti}$) and sends it to the reader along with the nonce R_t. The reader passes the information to the backend server which uses R_t to generate α' and checks if $\alpha = \alpha'$ to authenticate tag. If the tag is authenticated by the server, it generates response message $DATA_k \| Hash(RID_{ri} \oplus RID_{ti}) \| RID_{ri} \oplus s_{ti+1} \| RID_{ti} \oplus s_{ri+1}$ and forward this message to the reader. The reader transmits the message to tag who verifies server's response by generating its own RID_{ri} as $RID_{ri} = F_{s_{ri}}(R_r) = (R_r - R_r \bmod s_{r+1}) \| (R_t + s_r - R_r \bmod s_r)$ to authenticate backend server. After each successful session, current session secret values are updated to the new one.

Although Cho et al. [57] claim this approach to be safe against the most common type of security attacks in RFID system like desynchronization and tag impersonation attack, Peris-Lopez et al. [58] have falsified these claims by performing cryptanalysis. In [58] authors have presented cases in which successful desynchronization attack can be made by eavesdropping one session and altering the last message sent by the reader to the tag. The paper [58] also discussed scenarios to launch impersonation attacks onto the tag and reader.

Song and Mitchell [59] proposed a hash-based mutual authentication approach in 2008 which has been analyzed by Cai et al. [60] and is found vulnerable to the desynchronization and reader impersonation attacks. Cai et al [60] also provided suggestions to improve Song and Mitchell's approach. Tsudik [61] presented a single round protocol for authentication that uses monotonically increasing timestamp for authentication. The reader first sends the current timestamp to the tag and tag checks the validity of the timestamp by comparing it with the previously received timestamp. If the timestamp is newer than the previous timestamp, the tag records the timestamp and compute the hash of it with a secret key. This approach cannot resist tag impersonation attack as an adversary can send a large number of future timestamps to the tag and record tag's responses so that if any request comes in future with the same timestamp it can impersonate as the tag and sends the response to the reader.

There has been a bunch of other similar hash-based approaches to providing security against attacks on RFID systems but none of them has been proved to be completely secure. J.H. Ha and S.J. Moon's protocol [62] tried to provide security against forward tracing attack but failed to make it as secure as claimed[63]. Authors of [63] pointed out the cases in which the tag tracking is made possible by simply observing previous unsuccessful sessions. Similarly, G-I protocol proposed in [64] which is a hash based low-cost RFID mutual authentication approach. In [64] tags have secret keys k1 and k2 that is shared secret value between the tag and reader/backend server. Along with secret keys of tags the backend server also have their hashes stored. The backend server/reader and the tag can implement the Hash function and pseudorandom number generating operation. Despite all the claims made by authors, this protocol is vulnerable to tracking and desynchronization attack. An attacker can act as a legitimate server and sends a query to tag and in response tag replies with Hash(k1) and stops the authentication process further. Now, whenever next time tag is queried by the server it response with the same Hash(k1) which makes it vulnerable to the tracking and cloning as an attacker can also act as the tag since it knows tag's response already.

Lightweight and Ultra Lightweight Authentication Protocols:

There have been a plethora of protocols proposed to the date that can be categorized as lightweight or ultra-lightweight in terms of the computation cost. Most of the protocols in this category use pseudonyms (an authorized changing of IDs with the help of PRNG or other techniques) along with lightweight bitwise operations such as XOR, OR, AND. In this section, we have considered protocols that have been proposed so far in these categories along with their security analysis and weaknesses.

There is a family of three Ultra-Light Weight Mutual Authentication Protocols (UMAP) that includes EMAP [65], M2AP [66] and LMAP [67] proposed in 2006. In LMAP the RFID tag has a static ID that is tag ID and a dynamic pseudonym along with four subkeys (K = k1||k2||k3||k4). First, the reader sends a hello message to the tag to which the tag replies with the IDS. Upon reception of the IDS, the reader sends some values A, B, C (A = IDS $\oplus$k1 $\oplus$n1, B = (IDS$\vee$k2) + n2, C = IDS + k3 + n2 where n1 and n2 are random number picked by the reader) to the tag. The tag authenticates the reader based on these values and if the reader is authenticated, tag updates its IDS and sub keys (K) with new values and replies the reader with nonce value D (D = (IDS+ID) $\oplus$n1$\oplus$n2) so that reader can validate the response and authenticate the tag. M2AP also has similar structure as LMAP, only difference is that value of B (i.e. (IDS$\wedge$k2) + n2) is computed differently and after authenticating the reader tag responses with different values that are D and E (i.e. D = (IDS$\vee$K4) $\wedge$ n2, E = (IDS +ID) $\oplus$ n1). Last protocol in this family is EMAP which is also like two previously mentioned protocol. Figure 13 shows each of the three protocols. The basic flaw in this approach is that they are vulnerable to desynchronization attack as when tag authenticates the reader it immediately updates its IDS and keys regardless of the fact whether the reader has also authenticated it or not. Along with that protocols in this family are also vulnerable to the traceability attack, if an adversary sends a Hello message to the tag and do not proceed with the rest of the steps, tag does not change its IDS and hence will reply with same the IDS when queried next time.

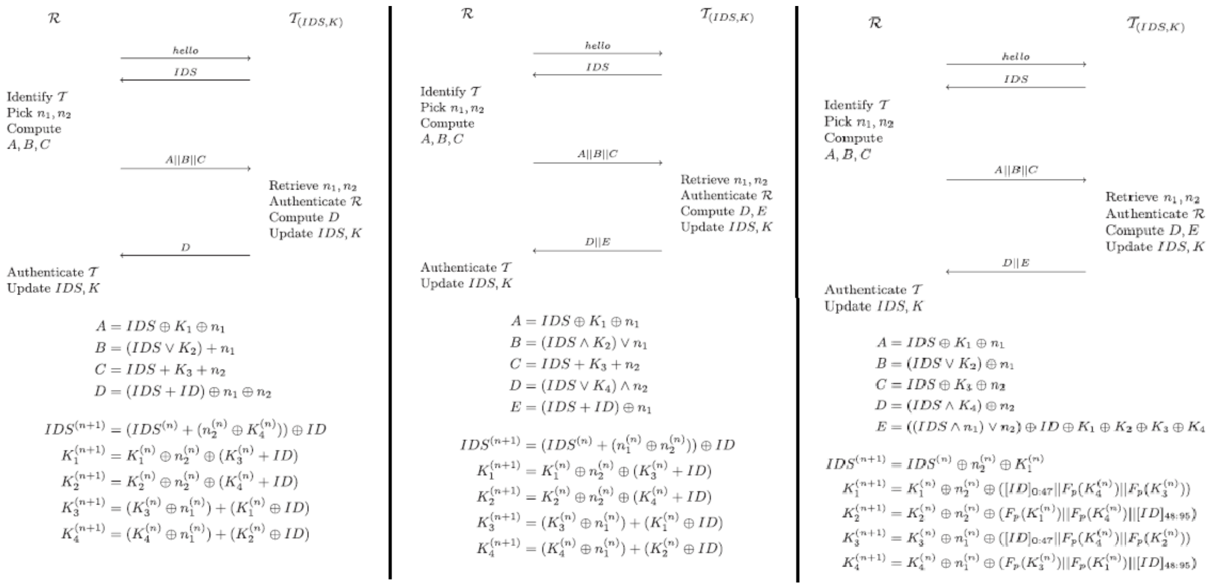

$\mathcal{R}$
$\mathcal{T}_{(IDS,K)}$
hello
IDS
Identify $\mathcal{T}$
Pick n_1, n_2
Compute
A, B, C
$A||B||C$
Retrieve n_1, n_2
Authenticate $\mathcal{R}$
Compute D
Update IDS, K
D
Authenticate $\mathcal{T}$
Update IDS, K
$A = IDS \oplus K_1 \oplus n_1$
$B = (IDS \vee K_2) + n_1$
$C = IDS + K_3 + n_2$
$D = (IDS + ID) \oplus n_1 \oplus n_2$
$IDS^{(n+1)} = (IDS^{(n)} + (n_2^{(n)} \oplus K_4^{(n)})) \oplus ID$
$K_1^{(n+1)} = K_1^{(n)} \oplus n_2^{(n)} \oplus (K_3^{(n)} + ID)$
$K_2^{(n+1)} = K_2^{(n)} \oplus n_2^{(n)} \oplus (K_4^{(n)} + ID)$
$K_3^{(n+1)} = (K_3^{(n)} \oplus n_1^{(n)}) + (K_1^{(n)} \oplus ID)$
$K_4^{(n+1)} = (K_4^{(n)} \oplus n_1^{(n)}) + (K_2^{(n)} \oplus ID)$
$\mathcal{R}$
$\mathcal{T}_{(IDS,K)}$
hello
IDS
Identify $\mathcal{T}$
Pick n_1, n_2
Compute
A, B, C
$A||B||C$
Retrieve n_1, n_2
Authenticate $\mathcal{R}$
Compute D, E
Update IDS, K
$D||E$
Authenticate $\mathcal{T}$
Update IDS, K
$A = IDS \oplus K_1 \oplus n_1$
$B = (IDS \wedge K_2) \vee n_1$
$C = IDS + K_3 + n_2$
$D = (IDS \vee K_4) \wedge n_2$
$E = (IDS + ID) \oplus n_1$
$IDS^{(n+1)} = (IDS^{(n)} + (n_1^{(n)} \oplus n_2^{(n)})) \oplus ID$
$K_1^{(n+1)} = K_1^{(n)} \oplus n_2^{(n)} \oplus (K_3^{(n)} + ID)$
$K_2^{(n+1)} = K_2^{(n)} \oplus n_2^{(n)} \oplus (K_4^{(n)} + ID)$
$K_3^{(n+1)} = (K_3^{(n)} \oplus n_1^{(n)}) + (K_1^{(n)} \oplus ID)$
$K_4^{(n+1)} = (K_4^{(n)} \oplus n_1^{(n)}) + (K_2^{(n)} \oplus ID)$
$\mathcal{R}$
$\mathcal{T}_{(IDS,K)}$
hello
IDS
Identify $\mathcal{T}$
Pick n_1, n_2
Compute
A, B, C
$A||B||C$
Retrieve n_1, n_2
Authenticate $\mathcal{R}$
Compute D, E
Update IDS, K
$D||E$
Authenticate $\mathcal{T}$
Update IDS, K
$A = IDS \oplus K_1 \oplus n_1$
$B = (IDS \vee K_2) \oplus n_1$
$C = IDS \oplus K_3 \oplus n_2$
$D = (IDS \wedge K_4) \oplus n_2$
$E = ((IDS \wedge n_1) \vee n_2) \oplus ID \oplus K_1 \oplus K_2 \oplus K_3 \oplus K_4$
$IDS^{(n+1)} = IDS^{(n)} \oplus n_2^{(n)} \oplus K_1^{(n)}$
$K_1^{(n+1)} = K_1^{(n)} \oplus n_2^{(n)} \oplus ([ID]_{0:47} || F_p(K_4^{(n)}) || F_p(K_3^{(n)}))$
$K_2^{(n+1)} = K_2^{(n)} \oplus n_2^{(n)} \oplus (F_p(K_1^{(n)}) || F_p(K_4^{(n)}) || [ID]_{48:95})$
$K_3^{(n+1)} = K_3^{(n)} \oplus n_1^{(n)} \oplus ([ID]_{0:47} || F_p(K_4^{(n)}) || F_p(K_2^{(n)}))$
$K_4^{(n+1)} = K_4^{(n)} \oplus n_1^{(n)} \oplus (F_p(K_3^{(n)}) || F_p(K_1^{(n)}) || [ID]_{48:95})$

Figure 13 The LMAP, M2AP and EMAP protocol from The UMAP Family.

There are other ultra-lightweight protocols SASI [68], KMAP [69], RCIA [70] and SLAP [71], all of these protocols have similar design as UMAP family and have been proved to be vulnerable to desynchronization attack by authors of [72] and since all these approaches have similar structures as UMAP family traceability attacks can also be made to these protocols.

One of the recent lightweight authentication protocol has been presented by Chen et al. [73] which is the dynamic token-based authentication scheme where the tag upon being queried by the reader sends a token value to the reader and update its key. The reader requests the central server for the tag token and matches the received token. If token sent by the tag matches the original token of the tag in server's database, it identifies tag as authentic and transmits a newly generated token to authenticate itself. The major drawback of this approach is that it is not resistant to desynchronization and replay attack. The tag updates its key soon after transmitting token to the reader and an unauthorized reader can desynchronize the tag and server by sending the hello request and not proceeding ahead. An unauthorized reader can also use tag's response to perform replay attack for later sessions. Authors of [73] have also presented an improved version of this protocol in [74] to avoid these attacks.

Sampangi et al. [75] proposed a secure authentication scheme that is secure against desynchronization attack as there is no key updation included in this scheme. In this scheme, the reader registers itself to the server and upon registration, the reader queries the server for the seed value of the tag. Once the seed value is issued, the reader use this value to generate different nonce values for carrying out the authentication process. This scheme is simple and lightweight as it does not make use of any complex operations or does not even lightweight bitwise operations, it only uses the PRNG to generate nonce values. The major drawback of this approach is that when the reader is registered to the server, it is issued a seed value of the tag, in case if that reader is compromised in future then it can impersonate as a tag since it has all the information (i.e. ID_t and seed value) which is required by a tag to carry out an authentication process.

3.3. Lightweight Hash Functions For RFID

One of the major challenges in using low-cost RFID in IoT is the security of the communication channel with limited computation capabilities of low-cost RFID tags (passive tags). Many protocols have been proposed for the secure communication assuming that traditional hash functions like MD5 and SHA-128 and SHA-256 can be computed by the tag. However, that is not the case, a RFID tag has only 2-5K GE available for security [15] which makes it computationally infeasible for the tag to perform these complex hash operations. To solve this issues researchers have come up with the idea of lightweight hash functions that are suitable the constraint devices like the passive RFID tag. In this section, we will discuss the existing lightweight hash functions and compare their performance and efficiency.

3.3.1. SPONGE Construction

Most of the existing lightweight hash functions [76] [77] and [78]are based on sponge construction. It was first proposed in [79] to build hash functions in a new way by using fixed permutation. It consists of three major components that are; state memory (S), a fixed length permutation function (f) and a padding function (p). The state memory is composed of c-bit capacity and r-bit bitrate and some fixed initialization value.

The sponge function has two main phases:

i) **Absorbing**: Firstly the message is padded such that total bits of the message become multiple of r, then it is divided into n-blocks of r-bit. Each message block is iteratively XORed with the bitrate part of the internal state and then the permutation function of fixed length (c+r) bits is applied.

ii) **Squeezing**: Once all blocks of the message are processed by the absorbing phase, r bits are extracted from the bitrate part of the internal state and the permutation p is applied onto it to generate a fixed r bit hash value.

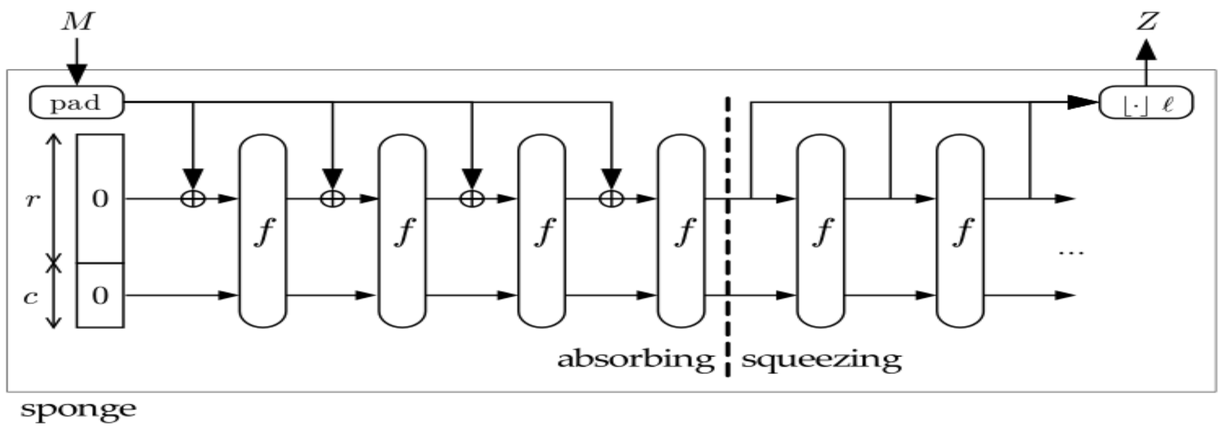


Figure 14 Sponge Function [23]

3.3.2. QUARK Hash Family

In [76] authors proposed three versions of QUARK hash functions based on sponge construction: U-QUARK (128-bit preimage resistance and at least 64-bit security against all other attacks), D-QUARK (160-bit preimage resistance and at least 80-bit security against all other attacks) and S-QUARK (24-bit preimage resistance and at least 112-bit security against all other attacks). U-QUARK is lightest among three. Like the sponge function, these hash functions contain the same absorbing and squeezing phase to extract a r-bit hash value. The permutation function P in QUARK is inspired by the stream cipher Grain and by the block cipher KATAN [24].

3.3.3. PHOTON Hash Family

Photon Hash Family [77] consist of five different variants of lightweight hash functions based on sponge construction. It defined an AES-like function to be a fixed key permutation P which contains Nr rounds and each round contains has four layers:

AddConstants (AC): Adds fixed values to the cells of internal state

SubCells (SC): For applying s-bits S-box to each of the cell.

ShiftRows (ShR): For rotating positions of cells in each row.

MixColumnsSerial (MCS): For linearly mixing all columns independently.

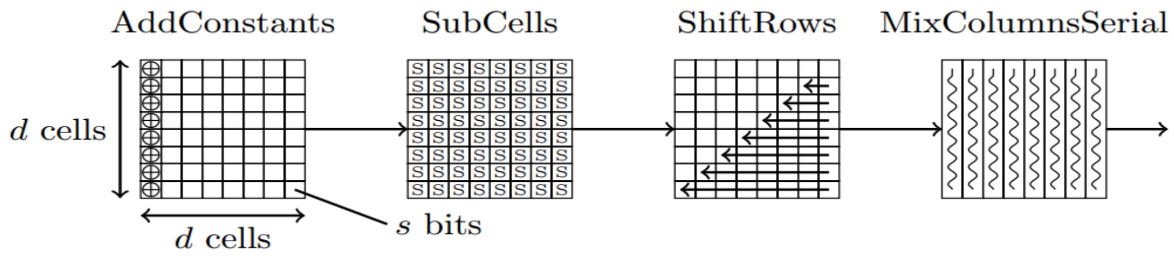


Figure 15 One round of photon permutation [77]

3.3.4. Spongent Hash Family

Spongent Hash Family [78] also have 5 different variations [spongent-88 spongent-128 spongent-160 spongent-224 spongent-256] each of them is constructed using the sponge function. It uses Present-80 like permutation P that consists of the S-BoxLayer, P-Layer and ICounter.

S-BoxLayer: This denotes the use of a 4-bit to 4-bit S-box S: F4 2 → F4 2 which is applied b/4 times in parallel.

P-Layer: This is an extension of the (inverse) present bit-permutation.

ICounter: This is one of the three [log2 R]-bit LFSRs. The LFSR is clocked once every time its state has been used and its final value is all ones.

Other lightweight hash functions available to the date are DM-PRESENT family [80], KECCAK-f family [81] , SHA-1 [82] and BLAKE [83].

3.3.5. Comparison

Out of all the existing lightweight hash functions, spongent hash functions is lightest one and suits best for RFID tags. Table 4 illustrate a comparison of different versions of each the lightweight function in terms of security and computation cost.

Hash Function	**Security bits**			**Hash length**	**Cycles**	**Area(GE)**
	Preimage	**Collision**	**PreImage2**			
SPONGENT-88	80	40	40	88	990 45	738 1127
SPONGENT-128	120	64	64	128	2380 70	1060 1687
SPONGENT-160	144	80	80	144	3960 90	1329 2190
SPONGENT-224	208	112	112	208	7200 120	1728 2903
SPONGENT-256	240	128	128	240	9520 140	1950 3281
Photon 80	64	40	40	80	708 132	865 1168
Photon 128	112	64	64	128	996 156	1122 1708
Photon 160	124	80	80	160	1332 180	1396 2117

Hash Function	Security Bits			Hash length	Cycles	Area(GE)
	Preimage	**Collision**	**PreImage2**			
Photon 224	192	112	112	224	1716 204	1736 2786
Photon 256	224	128	128	256	996 156	2177 4362
U-QUARK	120	64	64	128	544 68	1379 2392
D-QUARK	144	80	80	160	704 88	1702 2819
S-QUARK	192	112	112	224	1024 64	2296 4640
KECACAK-f [400]	160	80	160	160	1000 20	5090 10560
KECCAK-f [200]	128	64	128	128	900 18	2520 4900
SHA-1	160	80	160	160	450 490	6812 8588
DM-PRESENT-80	64	32	64	64	547 33	1600 2212

Hash Function	Security Bits			Hash length	Cycles	Area(GE)
	Preimage	Collision	PreImage2			
DM-PRESENT-128	64	32	64	64	559 33	1886 2530
H-PRESENT-128	128	64	64	128	559 32	2330 4256
C-PRESENT-192	192	96	192	192	3338 108	4600 8048

Table 4 Performance Comparison of lightweight hash functions.

3.4. Key establishment protocols using tree data structure

In the proposed approach, we tried to generate a random key from the random values stored in binary tree data structure. This work is inspired by key establishment approaches presented in [84] and [85] that uses tree data structure for generating the key.

In [84] authors presented a Hierarchical Identity-Based Encryption (HIBE) that has 4 phases: Setup, KeyGen, Encrypt and Decrypt. In the setup phase the master key is generated that is stored at depth 0 of the tree, the second phase involves key generation for other nodes of the system, it takes input from all the nodes at depth k and the private key of the parent node at depth k-1 and outputs private key for the node. Other two phases are related to encryption and decryption of the text using master key and the private key of the node. Another approach presented in [85] is Binary Tree Encryption that is a relaxed version of HIBE [84]. It also has same four phases but the method of the driving the key is different. In the first phase, it takes security parameters as input and returns an initial secret key and a public key of the current node. In the second phase, it takes to input the node and its associated secret key and outputs two secret keys for two child nodes (there can be only two child nodes as binary tree data structure is used). Along with above two key generations schemes there is also a similar key derivation algorithm A8 [100] used in GSM security which makes use of secrete key and random number to drive session key.

3.5. Motivation and Research Objective

In this chapter, we have presented the literature survey for IoT and RFID systems along with an in-depth analysis of the existing mutual authentication protocols. Based on the literature survey performed in this chapter we can conclude that most of the protocols were designed with an assumption that the reader cannot be dishonest or always have a secure communication channel with the server. Since RFID systems are being used in IoT which tends to make objects smart this assumption is not always true. There can be scenarios when the reader is not directly connected to a server or an attacker can impersonate as a reader. Along with this major assumption, most of the protocols have failed to provide protection against basic attacks in RFID systems like desynchronization attack, replay attack, and tag tracking. We also considered the lightweight hash schemes suitable for RFID tags. The main focus of this research work is to design an authentication protocol which is in compliance with industry standards. EPCglobal Class1 Gen2 standard [36] have specified the number of gates that a tag can use for security operation that is up to 5k. Hence using approaches like ECC that needs up to 8.2-

15 K logic gates [86], RSA that needs upto 10K logic gates [87] or other lightweight hash functions like SHA-1 or MD5 which needs 15 -20 K gates [88] does not seems practically feasible. Along with that we also need to design a solution that can eliminate the assumption made by previous researchers regarding always having a secure communication channel between the reader and server and provide a mutual authentication protocol that can protect against major security attacks.

Chapter 4: PROPOSED AUTHENTIATION SCHEME

4.1. Brief Overview of Proposed Approach

Most of the traditional RFID authentication protocols makes assumption that the RFID reader and server always have a secure communication channel or reader is always honest. With the advent of IoT this assumption is no more valid. In IoT, we have network of things that are smart enough to communicate with each other, any smart object can be equipped with an RFID reader which does not necessarily have a secure communication channel with server or whose security can be compromised in future. In our proposed approach, we have eliminated the assumption that reader is always honest by introducing reader-server authentication.

In the proposed approach, we used three party authentication scheme for RFID systems in IoT. The proposed scheme is designed in such a way that the RFID tag can only respond to a legitimate RFID reader so that if an attacker tries to impersonate as a reader, RFID tag will not identify it as a valid reader. Along with that we also used the lightweight hash function: Spongent [78] which is computationally feasible for resource constrained devices. Spongent lightweight hash is used to generate authentication signatures. In order to make signatures more secure, we have used the sandwich-keying methodology [89] . The key for hashing is generated using our newly proposed binary tree traversal key derivation approach. In the binary tree traversal key derivation approach, we have stored ‘n’ number of nonce values in a binary tree data structure and derived the key from it after applying bitwise operations and tree traversal approaches.

From here onwards, in order to make the approach easy to understand we have used 'reader' for RFID reader and 'tag' for RFID tag. Our proposed scheme has following three phases:

i) **Registration**: In registration phase, the reader first registers its self to the server, it sends a hello message to the backend server. Both reader and server use the Diffie-Hellman key exhange method [90] for generating the encryption key to encrypt messages in the communication.

ii) **Reader- Server Authentication**: After registering the reader, server shares a seed value with the reader. Both the server and reader can use seed value to generate a set of random values which are stored in form of a binary tree data structure. This binary tree is used to derive keys for generating signatures using keyed-hash mechanism. Once the reader and server have generated the same binary tree from which keys for generating signatures can be derived. The first the reader sends two random values p_{r1}, q_{r1} based on which server drives the key k1 and uses this key to perform keyed-hash on RID ,X_1 = Hash(RID,k1) . The server sends X_1 to the reader along with two random numbers p_{r2}, q_{r2}. The reader computes its own key k1 and generates X_1`. If $X_1$` = X_1 reader authenticates the server. The reader then generates X_2 using the same approach that is driving k2 from p_{r2} and q_{r2} and perform keyed hash to generate X_2. The reader sends the signature X_2 = Hash(X_1,k2) along with new random values p_{r3} and q_{r3} to the server. Server upon verifying signature creates a secret value Z and sends it to the reader along with the new signature X_3.

Reader upon receiving the message generates X_3` and validate the server's signature.

iii) **Server-Reader-Tag Authentication:** Once the server verifies and authenticate the reader as a legitimate entity it issues some secret value Y to the reader. The reader queries the tag with the value of Y. The tag analysis Y and based on common information it has with the server and checks for validity of the Y. The tag responses to the reader only if the value of Y is valid which means the reader is legitimate. If the reader is legitimate, the tag and server mutually authenticate each other using the same authentication mechanism via the reader.

In order to drive the key for hashing, we have used a binary tree traversing approach for making keys more random and unpredictable.

4.2. Architecture of the proposed approach

Our RFID system contains basic three components the RFID tag (EPCglobal Class1 Gen2V2), RFID reader (that can be resource constrained) and a backend server. The reader and server do not have any secure channel of communication. EPCglobal Class1 Gen2 V2 includes ultra-high frequency passive RFID tags that can have up-to 512 KB of the user memory block which can be re write by the reader which has permission to write on the tag.

The reader can be attached to any mobile device like a cellular phone or smart key chain or smart car. It can communicate with the server using any available wireless technology like Wi-Fi or LTE or Bluetooth. In order to initiate the communication with the tag, the reader needs to register itself with a backend server. Upon the registration, the backend server issues a secret value to the reader with the help of that value the tag can identify whether the reader is legitimate or not. The RFID reader has a 96-bit ID referred as RID from now onwards. The backend-server contains a 96-bit server ID along with a database that holds information about all tags. The database contains 96-bit tag IDs of each tag associated with the server, binary tree data structure to store nonce values for the key derivation for each tag (along with the latest table it holds two recent copies of tables to avoid desynchronization attack). The server also holds the information about the reader that tries to access some tag and the one-time password issued for the reader to read any tag. A RFID tag also has the binary tree data structure to store nonce values for the key derivation along with the 96-bit tag ID referred as TID and server ID.

Table 5 shows the information each entity in the RFID system contains along with the notation used to refer them.

Entity	**Information**	**Notation**
Reader	Reader ID (96-bits)	RID
Server, Tag	Server ID (96-bits)	SID
Tag, Server	Tag ID (96-bits)	TID
Reader, Server	Session Key for encryption (96-256 bits)	Kr
Tag, Server	Nonce Values to drive Key for hash (16-bits)	M1, M2, M3,…..Mi
Reader, Server	Nonce Values to drive Key for hash (16-bits)	N1, N2, N3……Ni

Entity	Information	Notation
Tag, Server	Current Key for hashing (16-bits)	$K_{ts}i$
Reader, Server	Current Key for hashing (16bits)	$K_{rs}i$
Reader, Server	Random values to drive key for hash	Pr(2-bit), Qr(4-bit)
Server, Tag	Random values to drive key for hash	Pr(2-bit), Qr(4-bit)

Table 5 Information table for each entity in RFID system

4.3. Detailed Description of proposed approach

In this section, we will analyze the proposed approach message by message and look into small details to have a better understanding of the flow of the authentication protocol. Before going into the authentication protocol, we will first look into the key derivation approach.

4.3.1. Binary Key Traversal Approach for Key Derivation

A binary tree is the tree data structure in which each parent has exactly 2 children. To traverse the tree there are two main approaches that are breadth first and depth first. We have considered a depth-first approach which further has three methods.

I) Pre-order: In pre-order approach, the parent node is visited first followed by left and then right child node.

II) Post-order: Post order approach is that first the left child with the highest depth is visited followed by right child including all the depths and finally the parent is visited.

III) In-order: In order,in this approach the order of visiting node is as a left child, parent, and right child.

InOrder(root) visits nodes in the following order:
4, 10, 12, 15, 18, 22, 24, 25, 31, 35, 44, 50, 66, 70, 90

A Pre-order traversal visits nodes in the following order:
25, 15, 10, 4, 12, 22, 18, 24, 50, 35, 31, 44, 70, 66, 90

A Post-order traversal visits nodes in the following order:
4, 12, 10, 18, 24, 22, 15, 31, 44, 35, 66, 90, 70, 50, 25

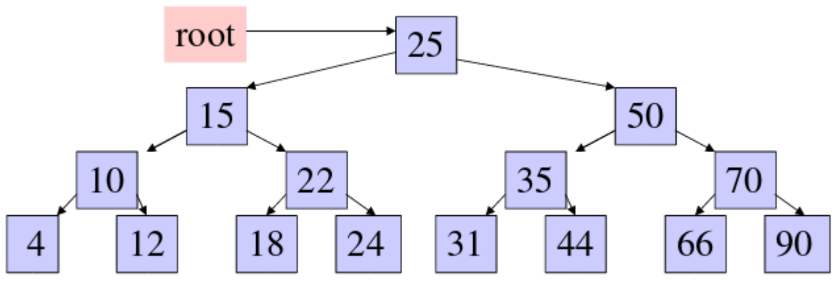


Figure 16 Binary Tree Traversing [91]

Each tag can have a 16-bit PRNG that can be used to drive random values of 16 bits, we have used this PRNG along with some data structure knowledge to keep keys more unpredictable.

This logic is not only used by the tag but also the reader and server which makes it more generalized. Two involve parties have PRNG and a shared seed value to derive the same set same random values from it. Both parties in setup mode use PRNG to derive ‘n’ random value (in our case we have chosen n as 16 considering the constrained resource devices like tag). The value of ‘n’ should be in power of 2 such that $n=2^m$. Once n random values are derived from the PRNG it is stored in form of binary tree data structure along with their corresponding next nodes index for each traversing approach, as shown in Table 6.

index	**Node**	**NextIndex(Pre-Order)**	**Next(Post-order)**	**Next(in-order)**
0	N0	1	7	11
1	N1	3	11	9
2	N2	5	0	13
3	N3	7	9	8
.	.			
7	N7	11	4	4
.				
11	N11	0	2	7
15	N15	0	6	7

Table 6 Nonce values along with the next node index.

Once both involved parties have the common nonce value table, it can generate same keys from this table, based on values of p and q, which is sent to the other party to derive the key where p is the random value that point to the index of the table $p \in \{0,1, 2,.15\}$ and q indicates the type of traversing technique to be used, $q \in \{0,1,2\}$. Upon receiving p and q, we can go the index p of the table and get values of m-1 nodes based on the value of q and XOR them together.

P||q
Node A
Node B

Figure 17 Node A and Node B

Node B take p= p mod(n -1) and q = q mod (3-1), here ‘n’ is the size of the table. The value of q decides which traversal technique to choose. It goes on index p and finds next m-1 values of N. Now the derived key let say K_{new} will be:

$K_{new} = Ni \oplus N_{next1} \oplus N_{next2} \oplus \ldots \oplus N_{next\ m-2}$

Replace Ni with K_{new} and use K_{new} as key for hashing.

Example:

Let say p=1 and q=1 so we choose post order traverse (0 = pre-order, 1 = post-order and 2 = in-order), for value of p we go to the index 1 and take N1 and next m-2 (4-2 = 2) nodes that are N3 and N7.

$$K_{new} = N1 \oplus N3 \oplus N7$$

In the table, we replace N1 by K_{new} for the next time and use K_{new} for keying hash function.

Parameters to use Binary tree key derivation in RFID system

RFID tag has 4 memory banks: EPC, User, Reserved, Tag ID. We can only use the User memory bank which is only up to 512-bits, so in RFID ideal size for n (i.e. table size) is 16 which is 2^4 and since n = 2^4, m will be 4. Instead of using all three traversing approaches in option we use only two of them to minimize the storage area needed to store the binary tree data.

Table size = 16*(16 bit nodes + 4-bit index + 2*(4bit next Indexes))

= 16*(12+16)

=448 bits

4.3.2. Mutual Authentication Protocol

Proposed mutual authentication approach contains three phases, each phase is discussed in detail below:

Registration Phase:

To access the information about any tag or query any tag the reader first needs to register itself to the server. A reader sends a Hello message to the server along with its 96-bits RID. In response to that, the server and reader establish the Diffie-Hellman key exchange algorithm to generate a session key for the encryption.

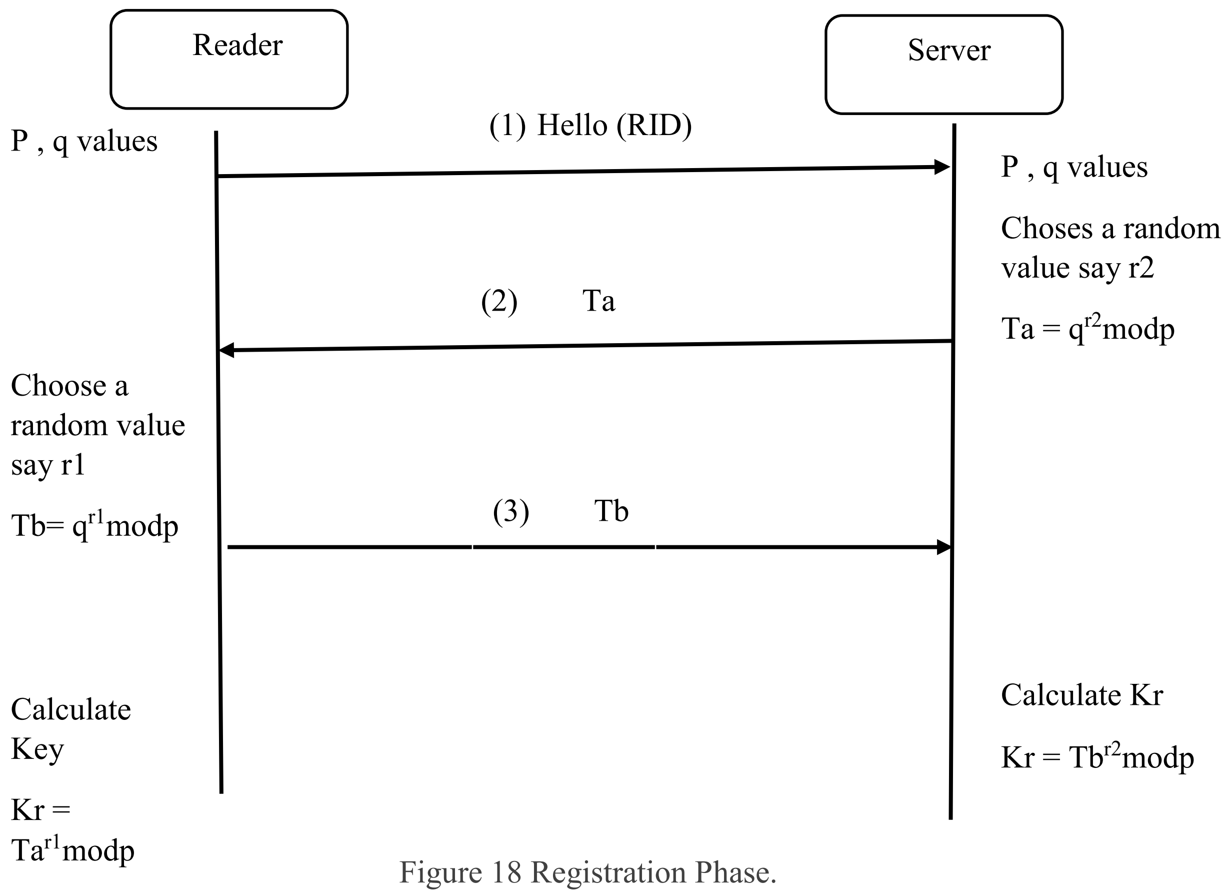


Figure 18 Registration Phase.

- In message (1) reader sends a hello message with its RID to the server.
- The server in response chooses a random number r_2 and generates Ta for the Diffie-Hellman key exchange and sends it to the reader in (2).
- Reader upon receiving (2) chooses a random number r_1 to generate Tb and sends it to a server in the message (3).

After an exchange of first three messages now the reader and server can generate an encryption key Kr for the further communication using the Diffie-Hellman approach.

Reader-Server Authentication:

Once the reader is registered with the server and both have an encryption key, the server sends the reader a seed value Seedr for PRNG. Both the reader and server use Seedr to generate a common set of nonce values $\{N_0, N_1, N_2 \ldots\ldots N_{15}\}$and store them into binary tree data structure along with their corresponding next node indexes. After that the reader sends p_{r1} and q_{r1} to the server, using which the server drives a key from the binary tree. This key is used to generate a signature X_1 = hash (RID, K_{rs1}). The server sends that to the reader along with new values for p_{r1} and qr_1. The reader generates its own X_1' and matches it with the server's response if X_1=X_1' the reader authenticates the server. In the next message, the reader sends its signature X_2 = hash (K_{rs2}, X_1) to the server along with new random values p_{r2} and q_{r2} where K_{rs2} is new key generated using values of p_{r2} and q_{r2}. The server upon receiving that signature from the reader generates its own X_2`and verifies the reader response by comparing the reader sent signature $X_2$ with $X_2$`. If both signature matches than the server consider the reader as authentic.

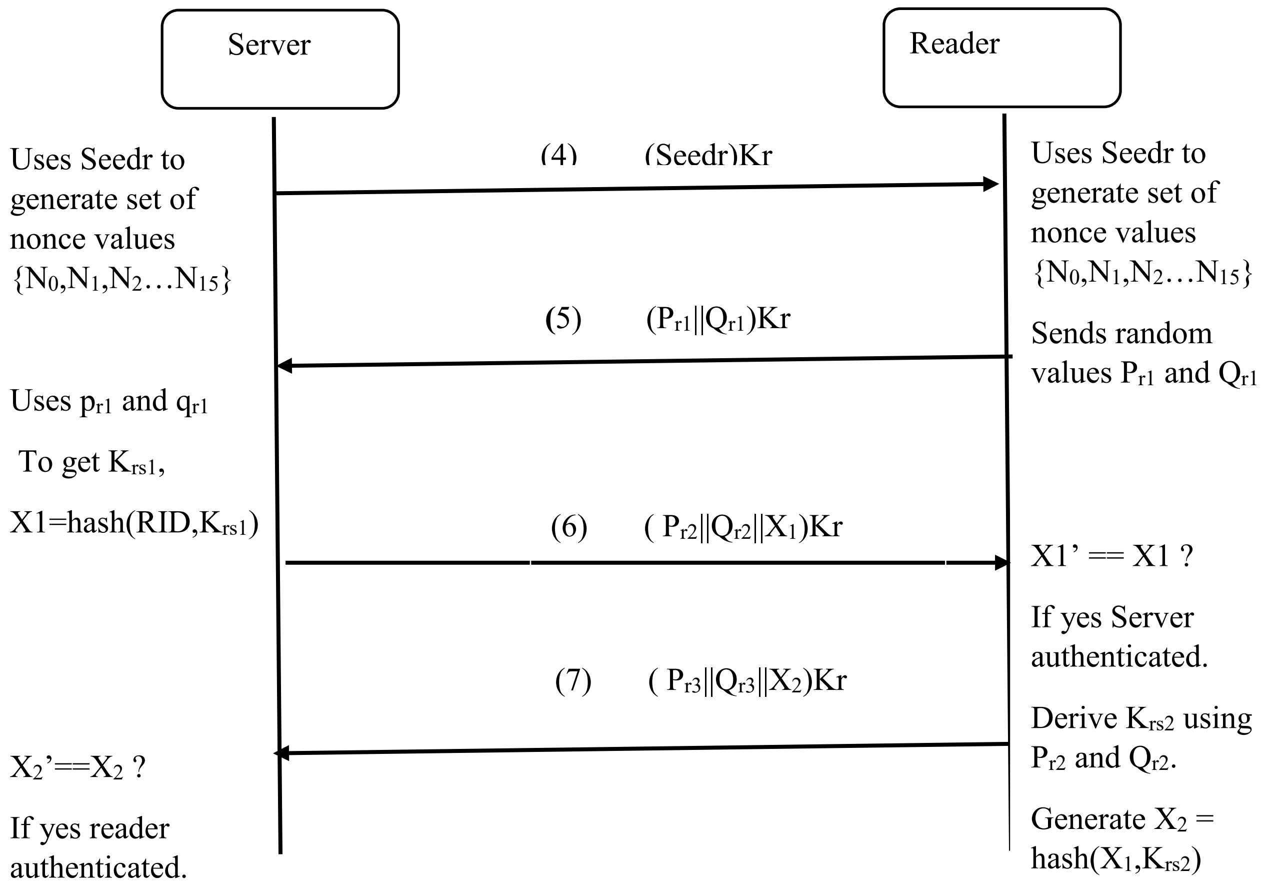

Server
Reader
Uses Seedr to generate set of nonce values $\{N_0,N_1,N_2\ldots N_{15}\}$
(4) (Seedr)Kr
Uses Seedr to generate set of nonce values $\{N_0,N_1,N_2\ldots N_{15}\}$
(5) $(P_{r1}||Q_{r1})Kr$
Sends random values P_{r1} and Q_{r1}
Uses p_{r1} and q_{r1}
To get K_{rs1},
$X1=hash(RID,K_{rs1})$
(6) $(P_{r2}||Q_{r2}||X_1)Kr$
X1' == X1 ?
If yes Server authenticated.
(7) $(P_{r3}||Q_{r3}||X_2)Kr$
Derive K_{rs2} using P_{r2} and Q_{r2}.
$X_2'==X_2$?
If yes reader authenticated.
Generate $X_2 = hash(X_1,K_{rs2})$

Figure 19 Reader-Server Authentication Phase

- In (4) after establishing an encryption key Kr the server the sends encrypted seed value Seedr to the reader. Along with that it also generates a set of nonce $\{N_0, N_1, N_2\ldots N_{15}\}$ using Seedr and PRNG.
- The reader decrypts (Seedr)Kr using same key Kr and generates the same set of nonce values as the server $\{N_0, N_1, N_2\ldots. N_{15}\}$. The set of nonce values is stored in into the binary tree data structure and calculates next node indexes using pre-order and post-order traversing techniques. Now reader generates two random values P_{r1}

and Q_{r1} for deriving the key from the set of nonce values. The reader sends p_{r1} and q_{r1} to the server in encrypted form in the message (5).

- The server upon receiving (5), decrypts it and gets p_{r1} and q_{r1} values from it to generate the key K_{rs1} as explained in binary tree traversal key derivation approach in the previous section. Using the key K_{rs1} it generates the keyed hash of RID that is referred as X_1. X_1 is used as a signature to authenticate the server at the reader's end. Along with X_1 the server also generates p_{r2} and q_{r2}. The server then encrypts $X_1||p_{r2}||q_{r2}$ with Kr and sends it to the reader in (6).
- The reader decrypts (6) to extract X_1, P_{r2}, and Q_{r2}. Based on the prior knowledge of P_{r1} and Q_{r1} the reader derives the key K_{rs1} and generates $X_1` = \text{Hash}(RID, K_{rs1})$. If $X_1` = X_1$ the server is authenticated. The reader uses p_{r2} and q_{r2} to generate the key K_{rs2} and computes $X_2 = \text{Hash}(X_1, K_{rs2})$ along with p_{r3} and q_{r3}. The reader will encrypt the next message $X_2||P_{r3}||Q_{r3}$ and sends it to the server in the message (7).
- Server upon reception of (7) decrypts it and computes $X_2' = \text{Hash}(X_1, K_{rs2})$. If $X_2' = X_2$ the server authenticates the reader.

Tag -Server Authentication Via Reader:

This phase includes two main parts, in the first part, the server issues a secret value to the reader based on which the tag can find out whether the reader is legitimate or not. In the second part, the server identifies which tag the reader is trying to read and issues a one-time-password(OTP) to the reader. Using OTP and the previously used signature based authentication mechanism along the tag and server authenticate each other via reader. Below is the description of each message exchanged during this phase.

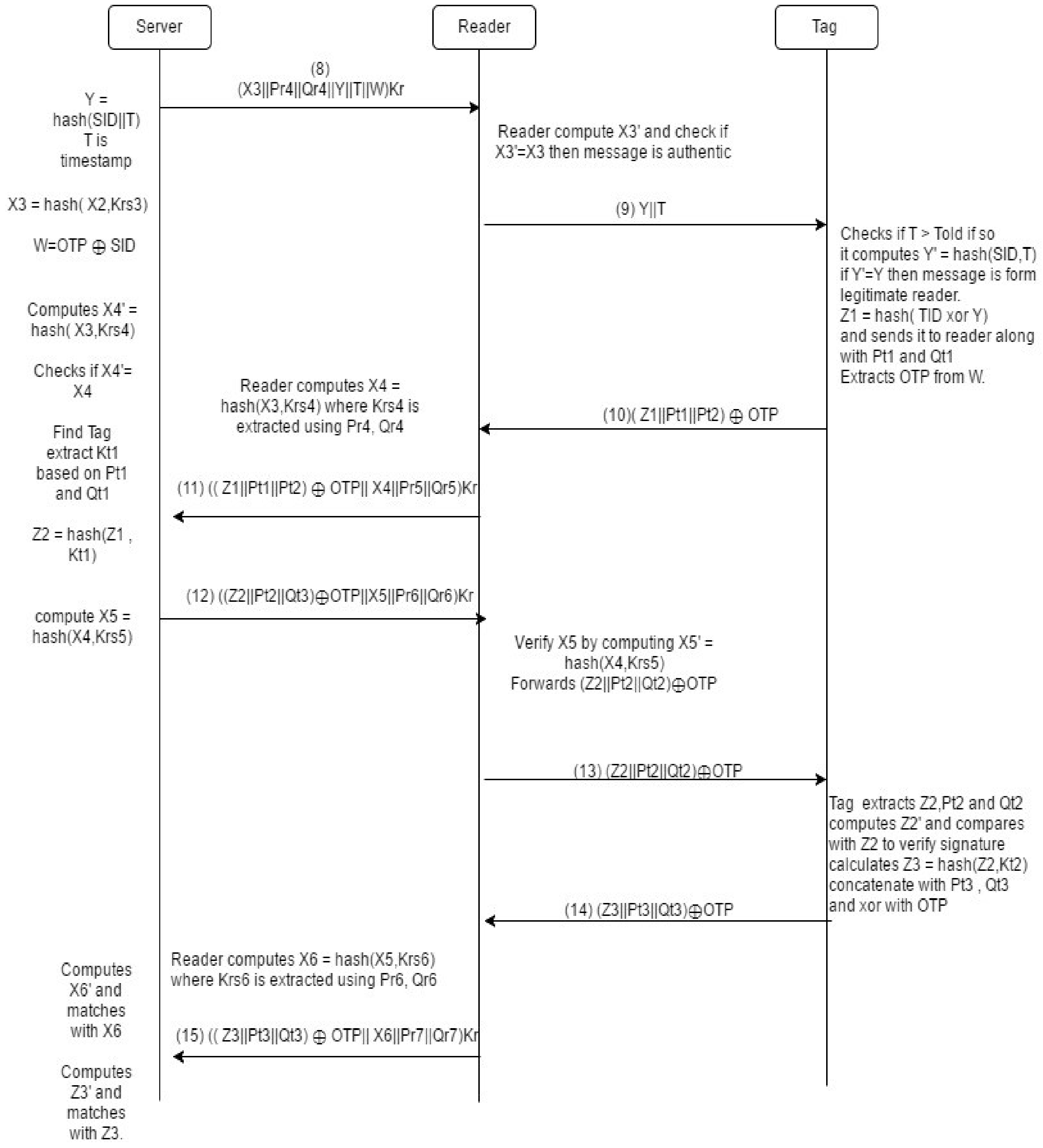

Server
Reader
Tag
(8) (X3||Pr4||Qr4||Y||T||W)Kr
Y = hash(SID||T) T is timestamp
X3 = hash(X2,Krs3)
W=OTP ⊕ SID
Reader compute X3' and check if X3'=X3 then message is authentic
(9) Y||T
Checks if T > Told if so it computes Y' = hash(SID,T) if Y'=Y then message is form legitimate reader. Z1 = hash(TID xor Y) and sends it to reader along with Pt1 and Qt1 Extracts OTP from W.
Computes X4' = hash(X3,Krs4)
Checks if X4'= X4
Reader computes X4 = hash(X3,Krs4) where Krs4 is extracted using Pr4, Qr4
(10)(Z1||Pt1||Pt2) ⊕ OTP
Find Tag extract Kt1 based on Pt1 and Qt1
(11) ((Z1||Pt1||Pt2) ⊕ OTP|| X4||Pr5||Qr5)Kr
Z2 = hash(Z1 , Kt1)
(12) ((Z2||Pt2||Qt3)⊕OTP||X5||Pr6||Qr6)Kr
compute X5 = hash(X4,Krs5)
Verify X5 by computing X5' = hash(X4,Krs5) Forwards (Z2||Pt2||Qt2)⊕OTP
(13) (Z2||Pt2||Qt2)⊕OTP
Tag extracts Z2,Pt2 and Qt2 computes Z2' and compares with Z2 to verify signature calculates Z3 = hash(Z2,Kt2) concatenate with Pt3 , Qt3 and xor with OTP
(14) (Z3||Pt3||Qt3)⊕OTP
Computes X6' and matches with X6
Reader computes X6 = hash(X5,Krs6) where Krs6 is extracted using Pr6, Qr6
(15) ((Z3||Pt3||Qt3) ⊕ OTP|| X6||Pr7||Qr7)Kr
Computes Z3' and matches with Z3.

Figure 20 Tag-Server Authentication via Reader

- After authenticating the reader as valid, the server generates a secret nonce Y based on which a reader can be considered as legitimate by the tag. Y = Hash (SID, T) where T is the current timestamp. Server sends (X_3||P_{r4}||Q_{r4}||Y||T) Kr to the reader as message (8).

- Upon receiving the message (8), the reader checks whether the response is from the legitimate server or not by computing X_3' = Hash (X_2, K_{rs3}) where K_{rs3} is generated using p_{r3} and q_{r3}. If $X_3'= X_3$ then it uses Y||T to interrogate the tag. The reader sends Y||T in (9) to tag.
- As the tag is queried by the reader it first checks the timestamp T it received which should be greater than the previous timestamp that the tag had in its memory. If $T_{old} < T$ it computes Y`= Hash (SID, T) where SID is the server ID that the tag already has. If Y' = Y then the tag can conclude that the interrogating reader is a legitimate reader and is registered by the server. The tag uses Y and XORs it with TID to generate Z_1 and sends it to the reader along with p_{t1} and q_{t1} in the message (10).
- Upon reception of the message (10), the reader concatenates its signature X_4 = Hash (X_3, K_{rs4}) along with P_{r5} and Q_{r5} and encrypts the message to transmit it to the server
- The server receives the message (11) and validates the authenticity of the message by computing its own X_4`. It compares $X_4$` with the received value of X_4. If both signatures match, the message is authentic. The server further checks for the tag that the reader is interrogating by hashing the Y with the tag IDs it has in the database. When it finds out the tag it extracts its corresponding nonce values set {M_0, M_1, M_2,.. M_n} to derive the key K_{t1} from it by using p_{t1} and q_{t2}. After deriving the key K_{t1} it uses this key to compute Z_2 = Hash (Z_1, K_{t1}) and send it to the reader along with P_{r6}, Q_{r6} and X_5 = Hash (X_4, K_{rs5}) where K_{rs5} is extracted using p_{r5} and

q_{r5}. The server also sends a 96-bits OTP XORed with its SID that is W = OTP$\oplus$SID to the reader.

- The reader decrypts (12) and checks if the message is authentic or not by computing signature X_5` and matching it with the server's X_5. If signatures match, it forwards the ($Z_2||P_{t2}||Q_{t2)}$ $\oplus$OTP and W to tag.
- The tag upon receiving (13) extracts Z_2 and compares it with Z_2' = Hash (Z1, K_{t1}). If Z_2`=Z_2 it authenticates the server. The tag then generates Z_3 = Hash (Z_3, K_{t2}) where K_{t2} is derived from the set of nonce values {M_0, M_1,…M_n} using P_{t2} and Q_{t2}. Tag now sends $Z_3||P_{t3}||Q_{t3}$ to the reader in the message (14) after XORing it with OTP.
- The reader after receiving the tag's response (14), appends its new signature X_6 = Hash (X_5, K_{rs6}) and encrypts the message with Kr to transmit it to the server.
- The server receives reader's response (15) and checks for the authenticity of the message by computing X_6' = Hash (X_5, K_{rs6}) and matching it with X_6. If both are equal then it extracts tag's response and validates it again in the same way by calculating Z_3` = Hash (Z_2, K_{t3}) and matching it with received Z_3.

After the last message (15) we can say that all three involved parties have authenticated each other using signatures and nonce values. With each message a new signature is generated which is different from the previous signature which makes this approach safe against the replay attack. An illegitimate reader cannot get a tag's response because of timestamp-based secret value generated by the server that is used by the tag to authenticate a reader. The proposed approach is also successful in avoiding tracking as every time tag's response is generated from the hashed value it received from a reader which should be different from the previous signature. To avoid desynchronization the server keeps a copy of two recent tables from which the key is generated which is the only thing that is changing. In order to make protocol more secure the tag ID is never exposed in plain text only the hash of tag ID XORed with a secret nonce (i.e. Z_1) is used in the message (10).

The main focus of the proposed approach is to design a mutual authentication protocol that does not allow a dishonest reader to read the tag in future and access the information regarding the tag. Bellow we have discussed the scenarios how the dishonest reader is dealt in each phase.

Reader Dishonest in Registration Phase : If reader is dishonest from the start, then the proposed scheme cannot detect the trustworthiness of the reader. But in order to make success full attach reader should also have access to tag's internal memory which is not possible until tag is tampered. The dishonest reader after successfully being registered by the server. It establishes an authenticated communication with the server.

Dishonest Reader in Authentication Phase: Even dishonest reader is authenticated by the server the proposed approach is designed in such a way that a reader is never issued a tag specific information it only gets the secret value Y which has no meaning for reader as value of Y is used by the tag to verify whether reader is registered and authenticated by the server or not.

Dishonest Reader in Tag-Server Authentication via Reader Phase: The following can be two possible scenarios:

i) In final phase if registered and authenticated reader is not honest even then the reader cannot get any meaning full information from the tag or server as all the information between tag and server is encrypted using OTP. And shared information only contains the signature values for tag and server to authenticate each other. The dishonest reader cannot extract any information from tag or server to launch successful attacks like replay attack or desynchronization attack or tag impersonation attacks.

ii) If a dishonest reader that is not registered and authenticated by the tag it can never get a tag's response as it needs a valid value of Y which is only issued to the authenticated reader.

We can conclude that although the proposed approach allows the dishonest reader to register and authenticate itself with the server but the approach is designed in such a way that even if the reader is fraud from start it cannot launch any attacks which compromise the security of the tag or server.

Chapter 5: IMPLEMENTATION

5.1 DEVELOPMENT ENVIRONMENT

The proposed approach is implemented using Java platform on a Windows operating system. Table 7 shows the details about the version of Java used along with the IDE and operating system version.

Programming Language	Java 1.8.0_65
Operation System	Windows 10.1
IDE	Eclipse Luna Version (4.0.4)

Table 7 Development Environment

5.1.1 Program Architecture:

In order to simulate RFID systems in IoT, we have used the client-server architecture. There are three parties involved in the communication that are the RFID tag, RFID reader, and backend server. The reader and server use one port for the communication whereas the reader and tag use another. There is no direct communication between the server and tag hence they can only communicate via reader.

5.1.2 Java Environment and Libraries used:

Java is the platform independent and easy to execute programming language which makes it one of the popular choices of programmers. For the implementation of the proposed approach, we have used many built-in libraries and packages such as java.security, java.crypto and java.net. In this section, we will discuss these libraries and their usage in our protocol.

i) **java.security:** This package includes a wide range of classes that provides security services and security providers such as MD5, SHA-256 and AES etc. It allows you to choose security options of your choice as per need [92]. In our implementation, the basic purpose of the java.security package is to provide the secure random number generator.

ii) **java.crypto:** This package provides cryptographic serveries including the message digest, key generations, digital signatures, symmetric and asymmetric encryption. We have included this package in our code to use the DES symmetric encryption and decryption to secure the communication channel between the reader and server.

iii) **java.net:** This package provides interfaces and classes to implement the network communication processes which involve the data transfer from one channel to another channel or from one machine to another machine. We have used some classes of this package to enable the reader, tag, and backend-server communication.

iv) **ServerSocket:** This class allows to accept the incoming connection requests from other sockets and data can be transferred using it. It needs a port number to open a socket for the communication. Once a socket is open, it listens to all the requests that are made to that port number and can send a response to the requesting party. To send and receive the data it uses various methods like dataOutputStream (), getInputStream etc.

Other packages used in the implementation are java.util and java.io. Java.util package provides interfaces for collections like Vector, ArrayList, TreeSet etc. whereas java.io is used to perform file read and write operations.

5.2. Implementation Details of the phases in proposed approach

In this section, we would discuss the implementation details of three major phases of proposed protocol. In chapter 4 we have presented message by message description of each, in this section, we would look how each message is generated and sent. We would first look into the major classes and their functionalities.

i) **RFIDReader**: As the name suggests this class is the reader class that provides the reader object to perform all major functions for the reader. It initiates a session with the server by sending a hello message to the server port. It can also interrogate a RFID tag. It has two open channels to listen, one with the server and other with the tag. It listens to both sockets and responses with appropriate replies.

ii) **RFIDServer**: This class acts as a backend server of a RFID system. It can communicate to any reader that sends requests to it. The major purpose of this server class is to listen to the socket channel and authenticate the requesting reader, register it and issues a One-Time-Password to it. It also has methods to validate reader's responses and tag's signature.

iii) **RFIDTag**: This class depicts all the functionalities that a RFID tag has. It can listen to the reader, checks if the reader is legitimate or not. It replies a legitimate reader with a signature so that a server can authenticate and identify a tag.

iv) **BinaryTreeTraversing**: The major purpose of this class is to provide functionalities needed to implement the binary tree traversal key derivation approach explained in section 4.3.1. The reader, server and tag class can access this class using public method GetKeys which takes the file name that holds the set of nonce values along with P and Q values and returns the key for hashing.

v) **Diffie-Hellman**; This class is used to provide the reader and server with methods for generating the key for encryption and decryption using the Diffie-Hellman approach, along with that it also includes some general functions needed by all other classes like getHexString (String str) to convert string into the hexadecimal, getStringFromHex (String Hex) to convert the hexadecimal string into the simple string, encrypt and decrypt method to perform symmetric encryption and decryption respectively. For the simplicity, we have used DES algorithm for encryption and decryption between reader and server.

vi) **Spongent**: This class is designed to compute the lightweight hash proposed in [78]. The main reason to design this class was that there is no built-in library in Java that for the lightweight hash function based on the sponge construction. The code for some methods of this class can be found on Google sites [93] which was written in the programming language C++ so we have rewrite whole code into Java. This class is accessible to the classes like the reader, server and tag via public function getHashValue (String str) that takes string value of any length as input and returns fixed 88 bit Hash value. Other methods of this class that is used in computing hash are Absorb (), Permute (), PLayer (), ICounter (), Sponge (), but these are private methods.

These major classes are linked with one another in terms of functionality. Objects of the RFIDReader, RFIDTag and RFIDServer act as parties involved in communication. Spongent, Diffie-Hellman and BinaryTreeKeyTraversing assist these three entities with functionalities that are needed to perform important tasks. Along with these classes there are some main methods used by objects of reader, server and tag classes to perform certain operations;

i) **XOR operations**: XOR is one of the basic bitwise operation used various time in the proposed approach. It is a lightweight operation that is used in most of the protocols for resource constrained devices like RFID tags. In the proposed approach, we have used XOR operation to minimize computation overhead at the tag's end and to secure the information. XOR operations are used in the message (8) to securely transfer OTP to the tag via the reader. OTP is XORed with server ID which is not known to reader hence reader cannot extract OTP but since the tag has the server ID it can derive OTP. OTP is later used to XOR the tag and server's signature values sent in (12) and (14) via reader. To compute XOR of two string we have converted strings into a character array and performed bitwise XOR on each character. A snippet of the code is given bellow:

```
Result_String [pos] = (char) (Operand1[pos] ^ Operand2[pos]);
```

ii) **Signature Generation and Validation**: Since this is an authentication protocol our main focus throughout this approach is to come up with a secure solution that allows all involved parties to authenticate each other and verify messages received by each other. To achieve this goal, we used signature based authentication scheme in which each message has a signature that is generated using a lightweight keyed hash function. A new key is generated for each signature using the common information shared by involved entities. With every message the receiving party receives a signature value (X_n). Since this signature is generated based on some commonly shared information the receiving party can generate the signature (X_n`) and compare it with the received signature. If the signature is validated, then the message is considered authentic.

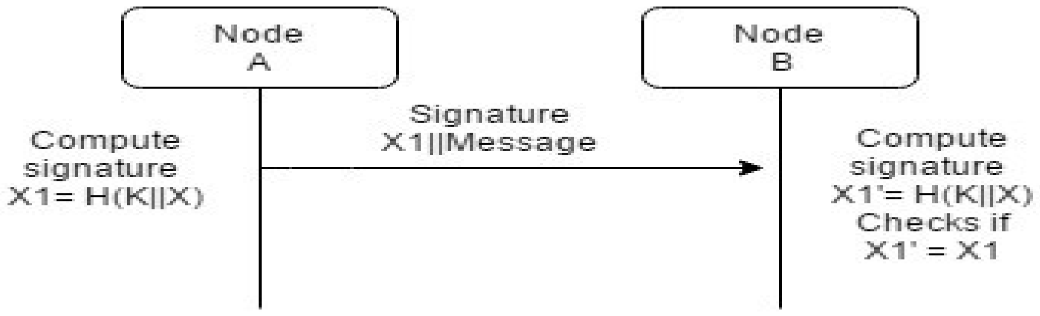

Node A
Node B
Compute signature X1= H(K||X)
Signature X1||Message
Compute signature X1'= H(K||X) Checks if X1' = X1

Figure 21 Signature generation and validation

iii) **Generating Random Number:** In this approach, all three parties: the reader, server and tag needs a random number generator to get a random sequence. For this purpose, we have used the SecureRandom that is a subclass of the Random class. SecureRandom can generate random bit streams in a secure manner. Random numbers generated by this class are not only cryptographically strong but also satisfies tests specified in FIPS 140-2, Security Requirements for Cryptographic Module [94]. Below is the code snippet used to generate a random number:

```
SecureRandom prng_generator = new SecureRandom(seed_Value);
 int rand = prng_generator.nextInt():
```

iv) **Socket Connections:** We have used sockets to establish two connections, one between the reader and server and second between the reader and tag. Each party listens to their corresponding sockets; validates the requests it receives and responds with appropriate replies. To perform all these tasks, we have used java.net package. For the communication between the reader and server, a socket object is created at port 19 whereas for the communication between the reader and tag a socket object is created at port 20.

Code snippet to create socket object is below:

```
ServerSocket tag_Socket = new ServerSocket(20);
```

Code snippet to listen to the channel for upcoming requests is:

```
Socket channel_Socket = tag_Socket.accept();
```

Once the connection is established successfully, the BufferedReader class is used to receive the incoming message. The BufferedReader class takes an object of the InputStreamReader class as an input which further needs an instance of the getInputStream class as an argument.

```
BufferedReader tag_receiver = new BufferedReader (new InputStreamReader
(tag_socket. getInputStream(),"UTF-8"));
```

To send data over the socket channel we need an ObjectOutputStream class which takes a getOutputStream function as an argument.

```
DataOutputStream sender = new
DataOutputStream(reader_Socket.getOutputStream());
```

Summary

We have used Java platform to simulate the communication flow of the proposed authentication scheme for RFID systems in IoT. We have created our own classes and have also used some built-in libraries and packages to implement the proposed approach.

Chapter 6: EXPERIMENTAL RESULTS AND ANALYSIS

In this chapter, we have discussed the experimental setup and results along with the comparison of the proposed approach with existing protocols. The major focus of the proposed approach is to provide a secure authentication protocol for RFID systems such that it can be adopted in IoT. In order to achieve this goal, we have used three party authentication scheme in which each party authenticate other party. This chapter is organized as follow: In section 6.1 we have discussed the experimental setup, how data is being collected and stored in each entity involved in the authentication process. Later in section 6.2 we have presented evaluation results achieved by performing traffic analyses via Wireshark. Section 6.3 deals with the security evaluation performed using the Scyther tool. We have also discussed the randomness testing performed on our key generation approach using NIST in section 6.4, which is followed by section 6.5 that presents security analysis of the proposed scheme. Section 6.6 presents the computation analysis on the proposed scheme and section 6.7 deals with the comparison of the proposed scheme with existing protocols.

6.1. Experimental Setup and Data Management

RFID system is simulated using the client-server architecture of Java programming language in a personal laptop with the following specification:

i) Operating System: Windows OS 10

ii) Processor: Intel® Core™ i7-4510U- CPU @ 2.00 GHz 2.6 GHz

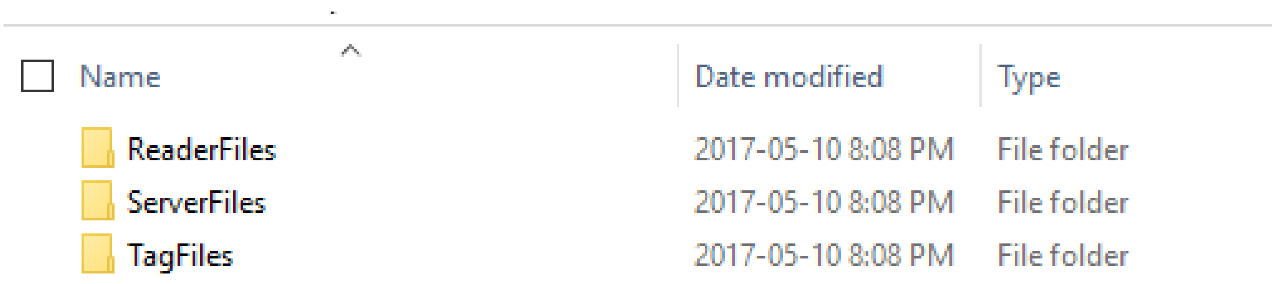
iii) Internal Memory(RAM): 8.00 GB

iv) System Type: 64-bit Operating System, x64-Jbased processor

v) Java Version: 1.8.0_65

vi) Hard Disk: 1 TB

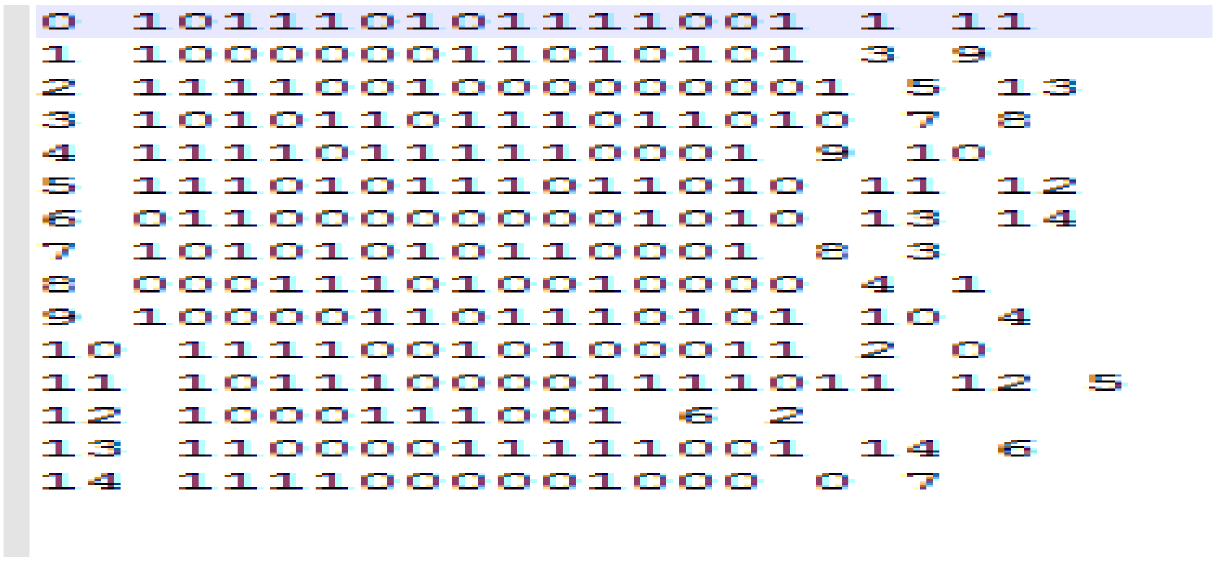
In order to store data each entity have used the internal space of a personal laptop. For organizing the data stored and used by each node, we have made separate folders for each instance in our Java project.



Name	Date modified	Type
ReaderFiles	2017-05-10 8:08 PM	File folder
ServerFiles	2017-05-10 8:08 PM	File folder
TagFiles	2017-05-10 8:08 PM	File folder

Figure 22 Data Storage folders for each entity.

Data for the reader involves the reader nonce value table used by our binary tree traversal key derivation approach along with the corresponding index of the next node for each traversing approach. Figure 23 illustrates an example of piece of data. Each record is separated by a new line and in each record, values are separated by a space. The first column represents the index value; the second column contains the nonce value whereas third and fourth column are next node indexes for the pre-order and post order traversing respectively. Similar files are also stored at the tag's end for their keys, whereas server holds the information of both the reader and tag key tables. For avoiding the desynchronization attack the server holds two recent copies of the tag's key table along with the latest copy. The tag also stores the data about the TagID, ServerID along with the last timestamp it received from the reader. The server on other hand has various data, it stores the information about the registered reader, the OTP issued for the reader, tag information table where it stores all the tag IDs for the tags with the nonce value table of each tag.



```
0  101110101111001  1  11
1  100000011010101  3  9
2  1111001000000001  5  13
3  1010110111011010  7  8
4  11110111110001  9  10
5  111010111011010  11  12
6  011000000001010  13  14
7  10101010110001  8  3
8  000111010010000  4  1
9  100001101110101  10  4
10  11110010100011  2  0
11  1011100001111011  12  5
12  1000111001  6  2
13  11000011111001  14  6
14  1111000001000  0  7
```

Figure 23 Nonce values table along with next node index for traversing.

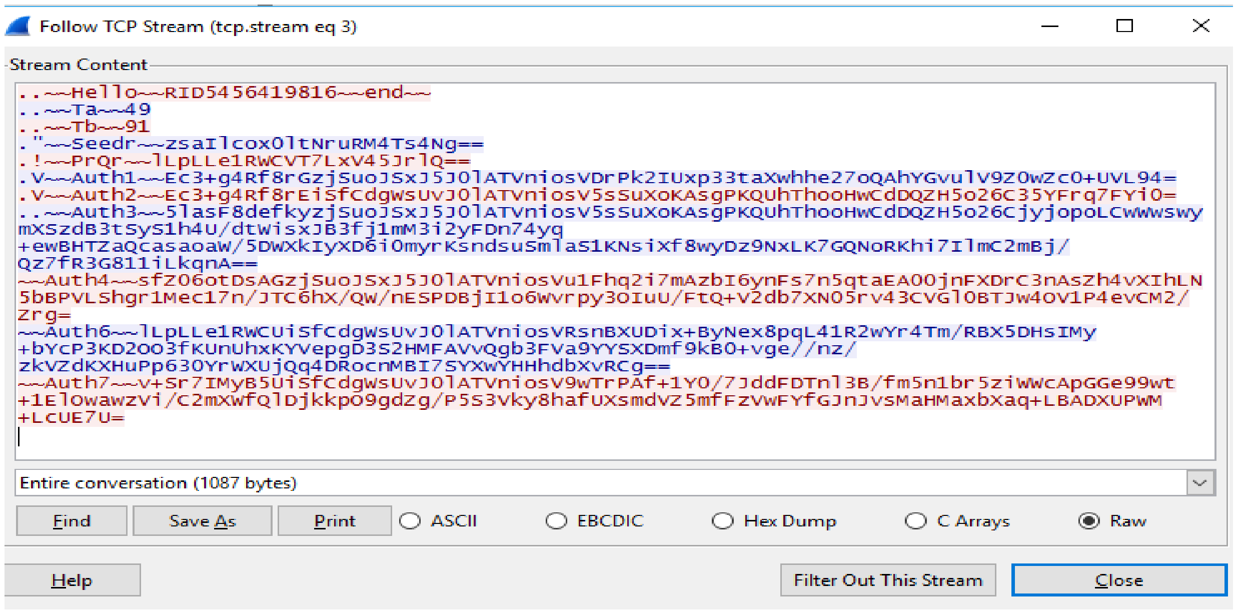
To perform the randomness test on keys generated using the binary tree traversal key derivation scheme we have stored all keys generated by it into a separate file. This file is later provided as an input to the NIST-STS, results obtained are discussed in section 6.4. All messages sent over from one party to another party over the network are encrypted. Data packets sent or received on the channel between reader and server are encrypted using DES (just for example, one can use any other symmetric key encryption) except for the registration phase where messages are not encrypted. Messages exchanged between the reader and tag are encrypted using one-time password (OTP). In order to perform traffic analyses, we have used the RawCap version 0.1.5.0 [95] that captures loopback packets generated by the code. Later we have analyzed captured packets using the Wireshark version 1.12.11 [96].

Security of the proposed approach is tested using the protocol verification tool Scyther. We have used version 1.1.3 of Scyther [97].

6.2. Evaluation using WireShark:

In the proposed approach, messages exchanged among involved parties are encrypted throughout the protocol except for the registration phase between the reader and server, where the reader first registers itself with the server and uses Diffie-Hellman approach to establish the encryption key. In order to differentiate messages sent at different points in the protocol we have appended a plain text string at the start of each message between the reader and server, rest of the message is encrypted.

We will analyze packets captured during each phase of the protocol in this section.



```
..~~Hello~~RID5456419816~~end~~
..~~Ta~~49
..~~Tb~~91
."~~Seedr~~zsaIlcox0ltNruRM4Ts4Ng==
.!~~PrQr~~lLpLLe1RWCVT7LxV45JrlQ==
.V~~Auth1~~Ec3+g4Rf8rGzjSuoJSxJ5J0lATVniosVDrPk2IUxp33taXwhhe27oQAhYGvulV9Z0wZc0+UVL94=
.V~~Auth2~~Ec3+g4Rf8rEiSfCdgWsUvJ0lATVniosV5sSuXoKAsgPKQUhThooHwCdDQZH5o26C35YFrq7FYi0=
..~~Auth3~~5lasF8defkyzjSuoJSxJ5J0lATVniosV5sSuXoKAsgPKQUhThooHwCdDQZH5o26CjyjopoLCWWswy
mXSzdB3tSys1h4U/dtWisxJB3fj1mM3i2yFDn74yq
+ewBHTZaQcasaoaW/5DWXkIyXD6iOmyrKsndsuSmlaS1KNsiXf8wyDz9NxLK7GQNoRKhi7IlmC2mBj/
Qz7fR3G811iLkqnA==
~~Auth4~~sfZ06otDsAGzjSuoJSxJ5J0lATVniosVu1Fhq2i7mAzbI6ynFs7n5qtaEA00jnFXDrC3nAsZh4vXIhLN
5bBPVLShgr1Mec17n/JTC6hX/QW/nESPDBjI1o6Wvrpy3OIuU/FtQ+V2db7XN05rv43CVGl0BTJw4OV1P4evCM2/
Zrg=
~~Auth6~~lLpLLe1RWCUiSfCdgWsUvJ0lATVniosVRsnBXUDix+ByNex8pqL41R2wYr4Tm/RBX5DHsIMy
+bYcP3KD2OO3fKUnUhxKYVepgD3S2HMFAVvQgb3FVa9YYSXDmf9kB0+vge//nz/
zkVZdKXHuPp63OYrWXUjQq4DRocnMBI7SYXwYHHhdbXvRCg==
~~Auth7~~v+Sr7IMyB5UiSfCdgWsUvJ0lATVniosV9wTrPAf+1YO/7JddFDTnl3B/fm5n1br5ziWWcApGGe99wt
+1ElOwawzvi/C2mXWfQlDjkkpO9gdzg/P5s3vky8hafUXsmdvz5mfFzVWFYfGJnJvsMaHMaxbXaq+LBADXUPWM
+LcUE7U=
```

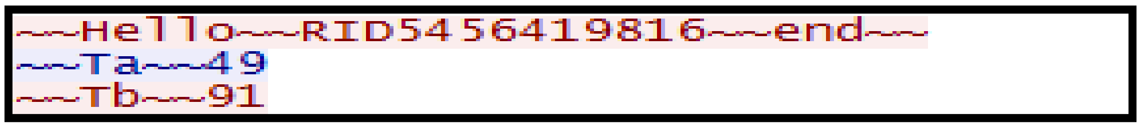
Entire conversation (1087 bytes)

( ) ASCII ( ) EBCDIC ( ) Hex Dump ( ) C Arrays [✓ Raw]

Figure 24 Data packages exchanged between Reader and Server.

6.2.1. Registration Phase

This phase involves two parties, that are the reader and server. The reader sends a hello message to the server and in response, both the reader and server use the Diffie-Hellman approach to generate the key. For using Diffie-Hellman they exchange a couple of messages that can be seen in the screenshot of packets captured via WireShark.

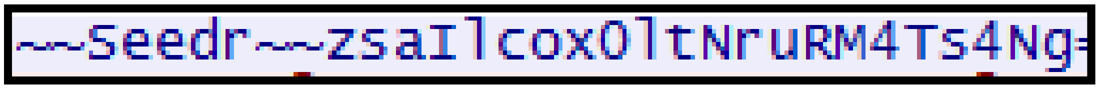


```
~~Hello~~RID5456419816~~end~~
~~Ta~~49
~~Tb~~91
```

Figure 25 Registration Phase messages send in plain text.

6.2.2 Reader-Server Authentication

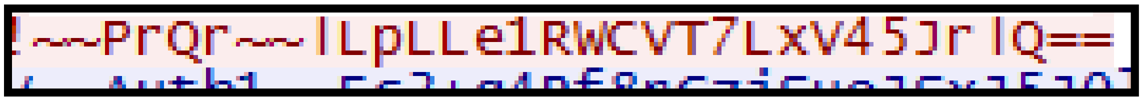
In this phase, the reader and server have already established a symmetric encryption key so now onwards data transferred between the reader and server is encrypted. We will go message by message and see the content of the messages visible for traffic analysis. As discussed above, in order to differentiate message from one another we have appended an identifying string in the start of each message. This plane text string is not part of the information (data) that need to be transferred in the actual protocol.



```
~~Seedr~~zsaI1coxO1tNruRM4Ts4Ng=
```

Figure 26 Message (4) Server sent encrypted seed value.

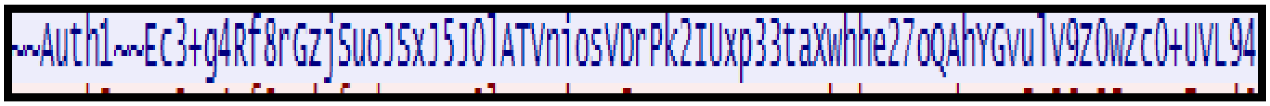
In message (4), the server has sent a seed value to the reader, the plain text '~~Seedr~~' is added just to identify the message.



```
!~~PrQr~~lLpLLe1RWCVT7LxV45Jr lQ==
```

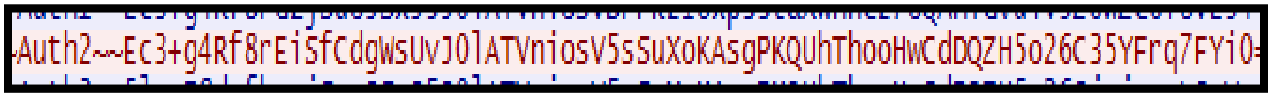
Figure 27 Reader reply with Pr and Qr value in the encrypted form.

Figure 27 is a snapshot of the message (5) sent by the reader to the server which includes Pr and Qr values needed to extract the hashing key. Figure 28 and 29 below display messages exchanged between the reader and the server to carry out the mutual authentication. We can analyze the packet content that all the data is in encrypted form and no third party can extract information from data without knowledge of the encryption key.



```
~~Auth1~~Ec3+g4Rf8rGzjSuoJSxJ5J0lATVniosVDrPk2IUxp33taXwhhe27oQAhYGvulV9Z0wZc0+UVL94
```

Figure 28 Message (6) Signature generated by server along with new Pr and Qr values

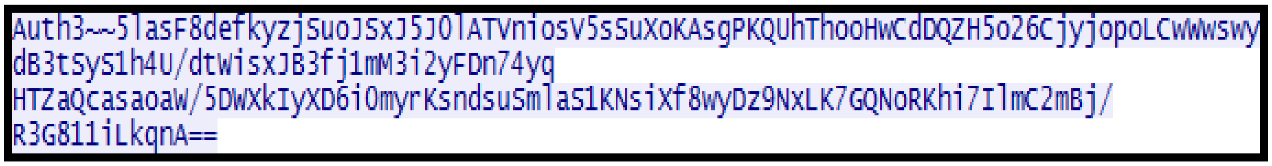


```
Auth2~~Ec3+g4Rf8rEiSfCdgWsUvJ0lATVniosV5sSuXoKAsgPKQUhThooHwCdDQZH5o26C35YFrq7FYi0=
```

Figure 29 Message (7) sent from reader to server containing a signature, Pr and Qr.

6.2.3 Tag-Server Authentication via Reader

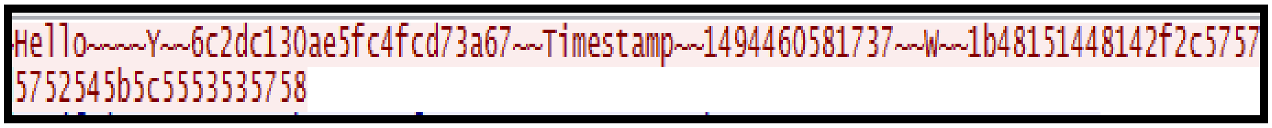
In this phase the tag and server authenticate each other after server identifies the tag with which the reader is interacting. This authentication between the tag and server is carried out via the legitimate reader. Once the server and reader mutually authenticate each other server sends some secret value to the reader that would be forwarded to the tag based on which the tag identifies if the reader is legitimate or not. This phase involves message-exchange between three parties as the tag and server cannot communicate directly so they exchange messages via the reader. Figure 30 shows the data packet sent by the server to the reader after mutual authentication. This packet contains a signature to verify the authenticity of the message along with some secret value that the server issues for the tag to identify the reader as legitimate and the OTP XORed with the server ID. As we can see in the snapshot of the packet no information is in plain text all the data in encrypted.



```
Auth3~~5lasF8defkyzjSuoJSxJ5J0lATVniosV5sSuXoKAsgPKQUhThooHwCdDQZH5o26CjyjopoLCwWwswy
dB3tSyS1h4U/dtWisxJB3fj1mM3i2yFDn74yq
HTZaQcasaoaW/5DWXkIyXD6i0myrKsndsuSmlaS1KNsiXf8wyDz9NxLK7GQNoRKhi7IlmC2mBj/
R3G811iLkqnA==
```

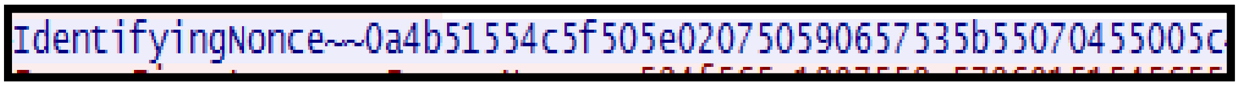
Figure 30 Message (8) Server sent signature and secret value.

Once the reader receives the message (8) it decrypts the message and checks the signature sent by the server to verify the authenticity of the message. If the message is authentic it then extracts the information sent to the reader for the tag and sends a hello message to the tag which is shown in Figure 31. The reader appends the secret value sent by server that is indicated as Y in the Figure 26 along with the timestamp and value of W, where W is XOR of the server ID and OTP. We can see in the Figure 31 that this message is not encrypted because the reader and tag do not have any key. One important fact to be considered here is that all the information sent in this message have no impact on the security of the protocol because the value of Y is exposed along with the timestamp but in order to compute a new value of Y, any eavesdropper will need the server ID which is not known. Other values exposed in this message is the value of W which is not in plaintext as it is XOR of serverID and OTP. Only the entity that has the serverID can extract OTP from it.



```
Hello~~~~Y~~6c2dc130ae5fc4fcd73a67~~Timestamp~~1494460581737~~W~~1b48151448142f2c5757
5752545b5c5553535758
```

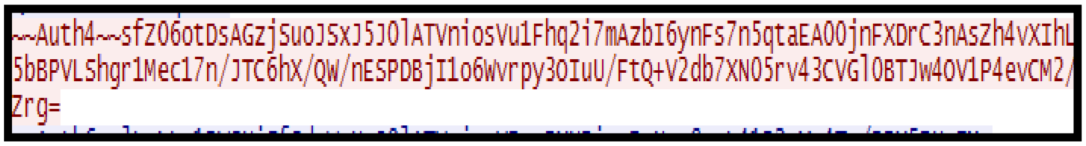
Figure 31 Message (9) from reader to tag.



```
IdentifyingNonce~~0a4b51554c5f505e020750590657535b55070455005c
```

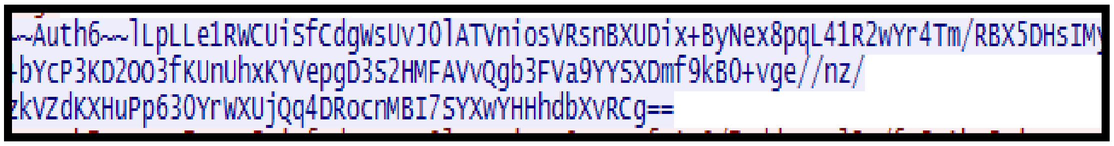
Figure 32 Message (10) tag sends an identification nonce to server via reader.

Figure 32 shows the message (10) that the tag sends to the reader only if it identifies it as a legitimate reader. In the message (10) tag has generated an identification secret value and encrypted it using OTP before sending it to the reader. The reader upon receiving this message will forward it to the server along with a new signature. Figure 33 represents the message sent to the server by reader and Figure 34 shows server's response after identifying the tag.



```
~~Auth4~~sfZ06otDsAGzjSuoJSxJ5J0lATVniosVu1Fhq2i7mAzbI6ynFs7n5qtaEA00jnFXDrC3nAsZh4vXIhL
5bBPVLShgr1Mec17n/JTC6hX/QW/nESPDBjI1o6Wvrpy3OIuU/FtQ+V2db7XN05rv43CVGl0BTJw4OV1P4evCM2/
Zrg=
```

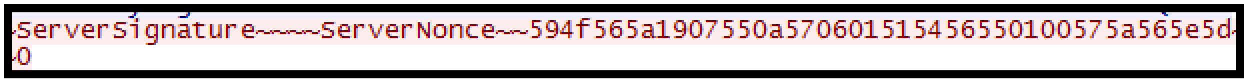
Figure 33 Message (11) sent by the reader to the server after interacting tag.



```
~~Auth6~~lLpLLe1RWCUiSfCdgWsUvJ0lATVniosVRsnBXUDix+ByNex8pqL41R2wYr4Tm/RBX5DHsIMy
-bYcP3KD2O03fKUnUhxKYVepgD3S2HMFAVvQgb3FVa9YYSXDmf9kB0+vge//nz/
zkVZdKXHuPp630YrWXUjQq4DRocnMBI7SYXWYHHhdbXvRCg==
```

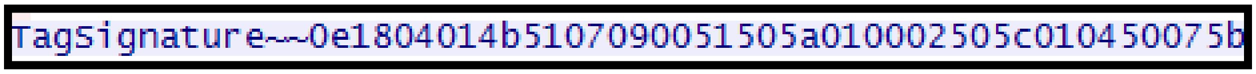
Figure 34 Message (12) sent by the server to the reader after identifying the tag.

After verifying the authenticity of message (12), the reader extracts the tag's part that is the server's signature which is already encrypted using OTP and send it to the tag in the message (13) shown in Figure 35. Tag upon receiving the message verifies the signature sent by the server via the reader and generates its own signature using Pt and Qt values sent by the server which is sent to the reader in the message (14). The reader forwards that message to the server along with its signature. The server first verifies the reader's signature to make sure that the message is received from a valid reader and then checks for tag's signature. If tag's signature is authentic it authenticates the tag as legitimate.



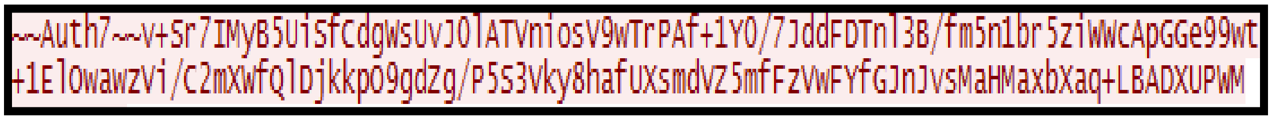
```
~ServerSignature~~~~~ServerNonce~~594f565a1907550a57060151545655010057 5a565e5d~
~0
```

Figure 35 Message (13) Reader forwards server's signature to tag.



```
TagSignature~~~0e1804014b5107090051505a010002505c010450075b
```

Figure 36 Message (14) Tag sends the signature to the reader.



```
~~Auth7~~V+Sr7IMyB5UisfCdgWsUvJ0lATVniosV9wTrPAf+1YO/7JddFDTnl3B/fm5n1br5ziWWcApGGe99wt
+1ElOwawzVi/C2mXWfQlDjkkpO9gdZg/P5S3Vky8hafUXsmdVZ5mfFzVWFYfGJnJvsMaHMaxbXaq+LBADXUPWM
```

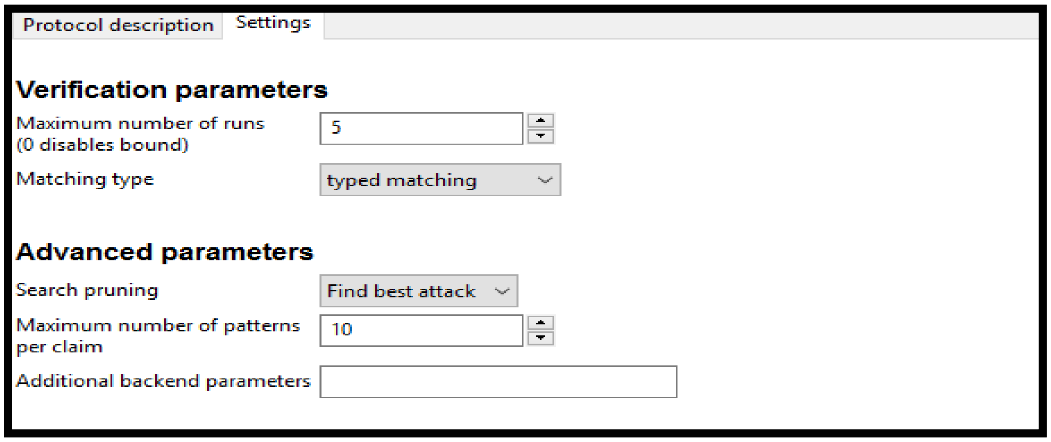
Figure 37 Message (15) Reader sends tag's signature with its own signature to server.

6.3 Security Evaluation Using Scyther

Scyther is a protocol verification tool, which analysis the security of the protocol and test is against most of the possible attacks that can be launched on it. Three main tasks performed by Scyther are as follow: First, it guarantees to prove the correctness of the protocol in an unbound number of sessions and provides the option of using a proof tree. Secondly, it provides classes of the protocol behavior to analyze the protocol. And finally, it also allows multi-protocol analysis that is in- parallel analysis of two sub-protocols [97].

For testing a protocol Scyther assumes that an adversary has full access to the contents of the communication channel. In order to test the proposed approach, we used following parameter for Scyther testing environment:

- Maximum number of runs: 5
- Matching Type: typed matching
- Search Pruning: Find all attacks
- Maximum number of patterns per claim: 10



Verification parameters

Maximum number of runs (0 disables bound): 5

Matching type: typed matching

Advanced parameters

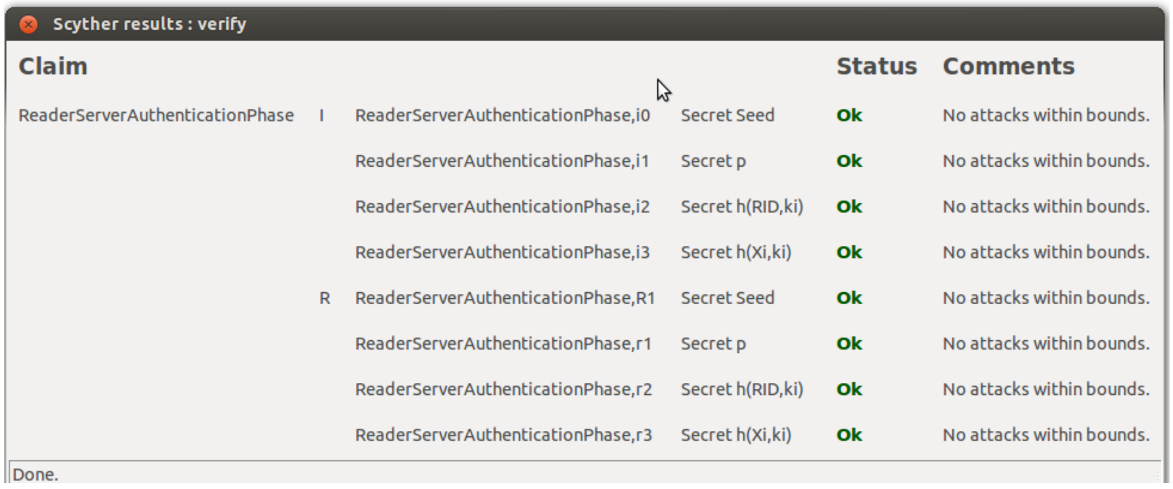
Search pruning: Find best attack

Maximum number of patterns per claim: 10

Additional backend parameters:

Figure 38 Scyther Testing Environment

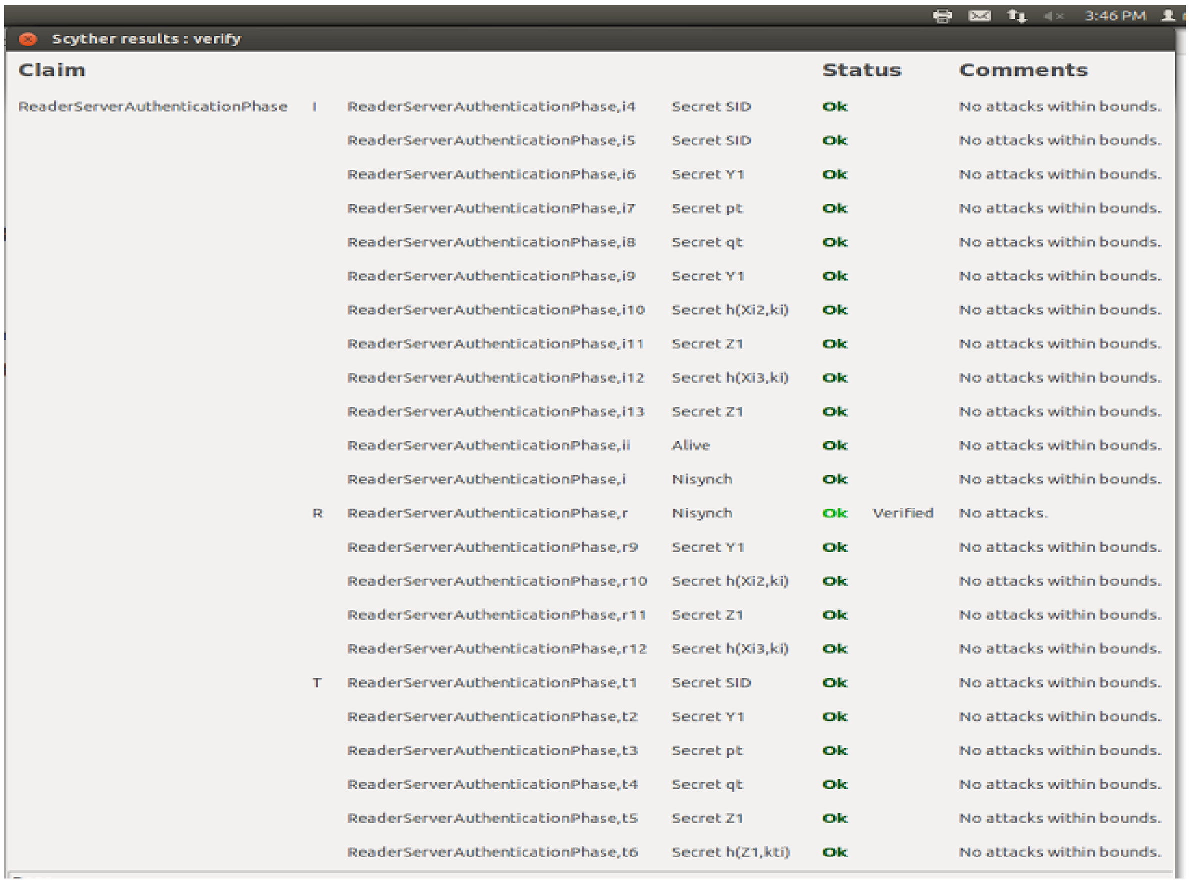
We have tested two out of three phases of the proposed approach on the Scyther because the first phase is using the Diffie-Hellman approach for establishing a key and registering the reader. The second phase is a major part of the protocol where the reader and server authenticate each other. We have used various claims for the secrecy of the information shared during message exchange along with claims of the aliveness and synchronization. Results achieved from the Scyther analysis for the phase two are shown in Figure 39. The Scyther results show that all the information shared during this phase is secure from all types of attacks. For the tag-server authentication via reader results of the Scyther are pretty match same. It has verified the security of the protocol against all possible attacks based on all claims made in the protocol, Figure 40 shows the Scyther results for the third phase.



Claim				Status	Comments
ReaderServerAuthenticationPhase	I	ReaderServerAuthenticationPhase,i0	Secret Seed	Ok	No attacks within bounds.
		ReaderServerAuthenticationPhase,i1	Secret p	Ok	No attacks within bounds.
		ReaderServerAuthenticationPhase,i2	Secret h(RID,ki)	Ok	No attacks within bounds.
		ReaderServerAuthenticationPhase,i3	Secret h(Xi,ki)	Ok	No attacks within bounds.
	R	ReaderServerAuthenticationPhase,R1	Secret Seed	Ok	No attacks within bounds.
		ReaderServerAuthenticationPhase,r1	Secret p	Ok	No attacks within bounds.
		ReaderServerAuthenticationPhase,r2	Secret h(RID,ki)	Ok	No attacks within bounds.
		ReaderServerAuthenticationPhase,r3	Secret h(Xi,ki)	Ok	No attacks within bounds.

Done.

Figure 39 Scyther Results for Phase Two Reader-Server Authentication



Claim				Status		Comments
ReaderServerAuthenticationPhase	I	ReaderServerAuthenticationPhase,i4	Secret SID	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,i5	Secret SID	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,i6	Secret Y1	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,i7	Secret pt	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,i8	Secret qt	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,i9	Secret Y1	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,i10	Secret h(Xi2,ki)	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,i11	Secret Z1	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,i12	Secret h(Xi3,ki)	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,i13	Secret Z1	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,ii	Alive	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,i	Nisynch	Ok		No attacks within bounds.
	R	ReaderServerAuthenticationPhase,r	Nisynch	Ok	Verified	No attacks.
		ReaderServerAuthenticationPhase,r9	Secret Y1	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,r10	Secret h(Xi2,ki)	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,r11	Secret Z1	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,r12	Secret h(Xi3,ki)	Ok		No attacks within bounds.
	T	ReaderServerAuthenticationPhase,t1	Secret SID	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,t2	Secret Y1	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,t3	Secret pt	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,t4	Secret qt	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,t5	Secret Z1	Ok		No attacks within bounds.
		ReaderServerAuthenticationPhase,t6	Secret h(Z1,kti)	Ok		No attacks within bounds.

Figure 40 Scyther Results for Phase Three Tag-Server Authentication Via Reader.

6.4. Randomness Testing Using NIST

In the proposed approach, we have used a binary tree traversal key derivation approach to extract keys for the hashing. The main goal of using this method is to derive random and unpredicted keys. In order to evaluate the randomness and unpredictability of keys generated we have used the Statistical Test Suite (STS) for random and pseudorandom number generator for cryptographic applications specified by National Institute of Standards and Technology(NIST) [98].

To perform the randomness testing via STS we have generated a ten thousand 16-bit keys and saved it in form of continues binary stream that is 160000 bits. Since in our case the key size is only 16 bits and STS do not support that smaller size of bit stream we have used the block size 256 for testing. The test suite assesses the randomness of each bit using a variety of statistical tests. For each test, the STS computes a p_value which is a measure of the probability that the sequence being tested is more random than that generated by a perfect PRNG. The p-value typically ranges from 0 to 1, higher the p_value more random the sequence is. In order to pass a test the p_value should be greater than 0.01. The set of sequences are considered random if the minimum number that is 98.7 % of sequences passes the test. Below is the description of randomness test that we have performed over the binary sequence.

Frequency (Monobit) Test:

This test computes the proportion of 1s and 0s in the input sequence, the purpose of this test is to find out whether the entire sequence has an equal number of 1s and 0s. If the number of 1s and 0s are almost equal the resulting p_value is high.

Frequency Test within a Block:

This test checks whether the number of 1s and 0s within an M-bit blocks in the sequence are approximately equal or not. If the proportion of 1s and 0s are approximately same, then the sequence is random.

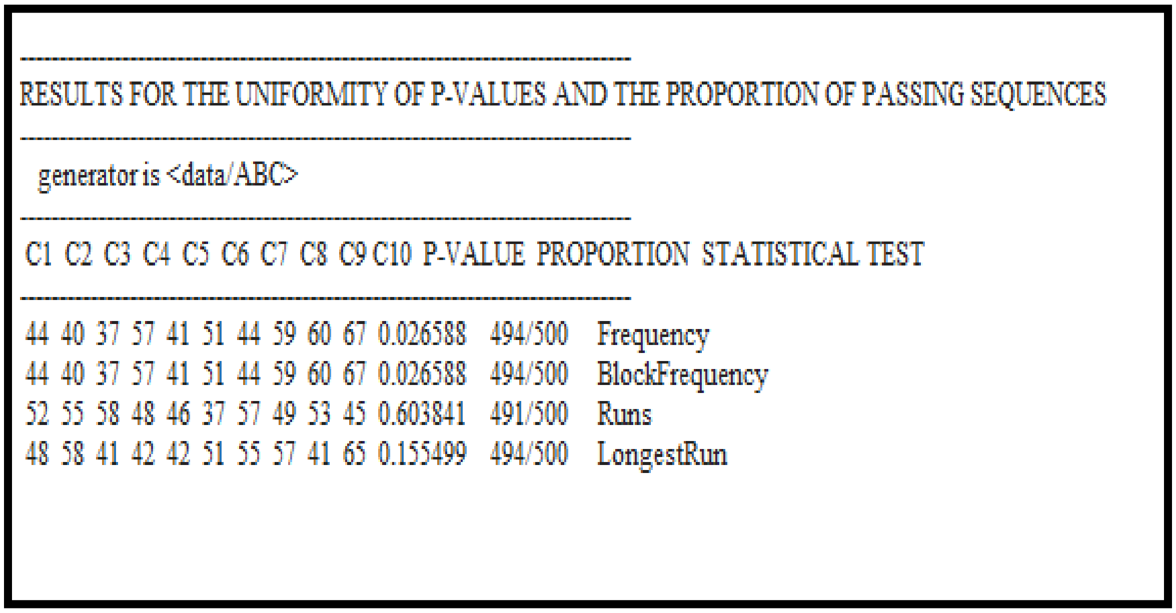
Runs Test:

This test checks for a total number of the uninterrupted sequence of identical bits (run) in a sequence, which determines whether oscillation between zeros and ones is too slow or too fast.

Longest Run of One's Test:

This test is carried out on the sequence to determine whether the longest run of ones in an M-bit block of the sequence is consistent with the length of the longest run of one expected in a random sequence. Any irregularity in the longest length of ones will result in irregularity in the length of longest run of zeros too.

Results obtained running statistical tests are given in Figure 41 with the sequence 1600000 bits, block size of 256-bit and total 500 sequences. Since we have 500 binary sequences minimum pass rate for each statistical test is 488. The sequence generated by proposed scheme passes all four tests.



```
------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------
  generator is <data/ABC>
------------------------------------------------------------------------
C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------
44 40 37 57 41 51 44 59 60 67 0.026588  494/500   Frequency
44 40 37 57 41 51 44 59 60 67 0.026588  494/500   BlockFrequency
52 55 58 48 46 37 57 49 53 45 0.603841  491/500   Runs
48 58 41 42 42 51 55 57 41 65 0.155499  494/500   LongestRun
```

Figure 41 NIST STS results.

We have run statistical tests using different parameters like the greater size of sequence and various block sizes, results attained after running those tests are given in the appendix. Most of the tests are passed in terms of the proportion of binary sequences passing the test but p_values are not uniformly distributed hence are less than passing criteria of 0.01.

6.5. Security Analysis of Proposed Approach

In this section, we have performed the security analysis of the proposed approach whether it meets security requirement and resists major security attacks.

Confidentiality, Integrity, and Authentication:

The proposed approach provides the confidentiality and integrity by using the encryption. The reader and server use the asymmetric encryption technique to secure the information exchanged between them. The communication between the tag and reader is secured using OTP to encrypt the data via XOR operations. The proposed scheme's basic purpose is to authenticate involved parties, so this requirement is achieved in this approach.

Forward Secrecy:

Due to updating keys for hashing and a new OTP for each session, even if a tag is compromised in future, previous keys and secrets cannot be found by an adversary. So, the previously shared information between the tag and reader is still secure.

Non-Repudiation:

Non-Repudiation is achieved by using a signature with each message. Both the reader and server send a signature with each message even after authenticating each other which verifies the authenticity of the message. None of the parties can later deny sending that message because the signature can only be generated by the party that has a previous signature and the common key for hashing.

Tag Cloning and Tag Tracking:

Tag cloning and tag tracking attacks are avoided as each time the tag replies to user query with a different signature based on its current key state and the previous signature, since the tag ID is never exposed in plain text (only hash of the tag ID along with some nonce value is sent in the message) the tag cloning and tracking attacks can be resisted.

Replay Attack:

The replay attack is also avoided in the proposed approach as if some previous message is sent again by an adversary at some point the signature sent with the message would not be authentic hence the message will be discarded and the session would be terminated.

De-synchronization Attack:

This attack occurs when the tag and server are not synchronized which means that both have different common knowledge (keys or IDs etc.). In the proposed approach, the only thing that is changing with each message is the hashing key and the table from which that key is derived. In order to avoid the desynchronization, the server keeps two previous copies of the key table. If an adversary tries to corrupt the data and interrupts the communication channel in such a way that the server's key table is updated but the tag has some old table server can still identify the tag. In order to avoid the similar scenario where tag's table is updated and the server's table remains un-updated, the server keeps values of previously sent Pt and Qt on the base of which the tag might have updated table so that server can update its table.

The Scyther evaluation, in Section 6.4, shows that attacks such as modification and fabrication are not possible on data packets exchanged in proposed approach.

6.6. Computation Analysis

The proposed approach is designed for the EPCglobal class1 Gen2V2 RFID tags, that has limited computation capabilities. It has up to 7-15K GE out of which only from 250 to 5k can be used for the security purpose [15]. In this section, we analyze total computation cost of the proposed approach in terms of the GE. For each message, we will calculate the GE required by tag to respond to the reader. In [99] authors have presented a table that states

GE required for each bitwise operation. Table 8 shows the operations used in the approach and GE used to perform these operations.

Operation	**GE**
96-bits XOR	254.4
16-bits XOR	42.4
Spongent Hash	738[23]

Table 8 GE used by each operation

Table 9 shows the operation used in each message computed at tag's, GE used by each message and total GE used throughout the protocol.

Message	**Operations Used**	**GE for each Operation**	**Total GE**
Message ()	Hash(SID,T)	738	1942.4
	OTP =SIDxorW	254.4	
	Hash(TID,Y)	738	
	Z1=Hash(TID,y) xor OTP	212	
Message()	Z2 = Received xor OTP	254.4	2069.6
	Z2=Hash(Z1,kti)	738	
	Z3=Hash(Z2,kti)	738	
	(Z3||pt||Qt) xor OTP	254.4	
	Key extraction (2 16 bit xors)	84.8	
Total	4012.0		

Table 9 Operations in each message along with GE

Total GE used by the proposed approach is 4012 which is still feasible for the EPCglobal Class1 Gen2V2 tags. For minimizing the computation cost and making the proposed scheme more low-cost and lightweight size of the key (16 bit) and IDS (96-bits TID, SID and RID) can be reduced.

6.7. Comparison with Existing Approaches

In this section, we will consider comparing our proposed approach with various existing schemes in term of the security. There has been a plethora of RFID authentication protocol proposed so far which has been discussed in chapter 3. Since our research work is contributed towards using RFID in IoT and there are numerous existing approaches we have only chosen some of the recent RFID authentication schemes that can be potentially used for RFID systems in IoT. Table 10 illustrates existing schemes and the proposed approache along with the capability of each approach to provide security against corresponding attacks or security issues.

Existing Approaches	**Replay Attack**	**De-synchronization**	**Forward Secrecy**	**Traceability**	**Reader-Server Authentication**
AKE-MRFID	NO	NO	YES	YES	YES
SHA-3 RFID	YES	YES	YES	NO	NO
SLAP	YES	NO	YES	NO	NO
Dehkorki	YES	YES	NO	YES	NO
Khedr	YES	YES	NO	YES	NO

Existing Approaches	**Replay Attack**	**De-synchronization**	**Forward Secrecy**	**Traceability**	**Reader-Server Authentication**
Habibie	YES	YES	NO	YES	NO
R. V. Sampangi	YES	YES	YES	YES	NO
Our Protocol	YES	YES	YES	YES	YES

Table 10 Comparison with existing approaches

Summary

The evaluation done using the WireShark provides the traffic analysis of the proposed approach that shows each information shared in the protocol is encrypted except for the registration phase. However, in the registration phase, both the reader and server use the Diffie-Hellman key exchange algorithm it is not possible to derive the key as the adversary has no knowledge of p and g. Another message that is sent in plain text is the first message sent by the reader to the tag in order to query the tag, this message contains a secret value sent by the server to the tag via the reader to identify whether the reader is legitimate or not. That secrete value can only be useful for he entity that already has a SID which only tags and the server have. The Scyther evaluation is carried out to test the protocol against different attacks and results shows protocol is free from most of the major attacks. The Scyther results also prove that all the data transferred over the communication channel is secure. We have also analyzed the randomness of keys extracted using the proposed binary tree traversal key derivation approach. Finally, we have performed the security and computation cost analyze for the proposed protocol. Security analysis shows proposed approach satisfies major security requirements like confidentiality, integrity, authentication, non-repudiation and forward secrecy. It can also resist most of the major attacks that can be done on RFID system like replay attack, desynchronization attack, cloning and location tracking etc.

Chapter 7 CONCLUSION AND FUTURE WORK

The proposed three-party authentication scheme is an attempt to increase the security of RFID systems in IoT. This research work was motivated by the need of an authentication scheme that eliminates the assumption that a reader and backend-server always have a secure communication channel.

The authentication scheme presented in this paper has three main phases: the Registration phase where the reader registers itself to the server, the Reader-Server authentication phase in which the reader and server authenticate each other, the Tag-Server authentication via the reader where a server identifies the tag and both parties authenticate each other.

The proposed scheme needs a RFID reader to first register itself with the server before querying any RFID tag. It is designed in such a way that a RFID tag only replies to a legitimate reader based on the secret value that server issues to the reader. The communication between RFID reader and server is made secure by using symmetric key encryption along with that, messages exchanged between the RFID reader and the tag are also encrypted using OTP.

The proposed approach also uses the signature based message authentication to verify the authenticity of each message. With each message the involved party appends a signature based on which the receiving entity makes sure that the message is sent by the legitimate party.

The Wireshark traffic analysis proves that the data transferred do not expose key information, Tag ID or Server ID in plain text. All the information shared including signatures and parameters required to generate the key for hashing are encrypted. Thus, we can say that even if an adversary intercepts the communication channel he cannot get any meaning full information out of it without knowledge of the key.

The Scyther results show that there are no possible attacks that can compromise the security of the proposed protocol. All the information shared throughout the authentication scheme is secure.

For generating signatures, we have used the lightweight hash function Spongent [78] which is the most lightweight hashing scheme available till date. Instead of using simple hash we have used the keyed hash to make the signature more unpredictable. Keys for hashing are extracted using our proposed binary tree traversal key derivation approach. The randomness and uniqueness of keys generated using this approach are tested via STS by NIST.

7.1. Limitations

The proposed approach is tested on protocol verification tool called Scyther that checks whether the protocol is resistant to major security attacks but there is a limitation of using Scyther as it cannot test any protocol against replay and desynchronization attacks. Although in order to avoid the replay attack in the proposed approach, we have used a new signature with each message. If some adversary tries to resend the packet captured some time before in the communication channel, the signature will be outdated by that time, as the key state of the involved party would have changed. For avoiding the desynchronization attack, the server is required to keep the copies of two previously used key states along with P and Q values sent. If the server and tag (or reader) get out of sync the server can extract the possible keys that other party may have. Despite these mechanisms, we have not tested the proposed scheme with some verification tool that can test the protocol against replay and desynchronization attack.

Keys generated using the binary tree traversal approach for key derivation have been tested for the randomness and uniqueness but for the 1600000-bit sequence it passes four basic test but when the size of the sequence is increased it fails to satisfy randomness criteria of having p_value greater than 0.01.

For security purpose the proposed scheme needs tag to come with server ID associated with it which limits the application areas of the proposed scheme to personal IoT based application only where there is always only one designated server associated with each object. Examples of such applications are smart homes, smart office and smart cars where we already have list of smart objects and a dedicated server.

7.2. Future Work

For this research work, we have implemented our proposed approach using Java programming language on a personal laptop and tested its security using the protocol verification tool Scyther. We have not implemented the logic on actual integrated circuits of RFID tags using digital logic hardware implementing platforms like Field-Programmable Gate Arrays (FPGA). FPGA does not have any specific functions we can program the integrated circuits as per the application requirement to attain the desired functionality.

For the randomness testing of the proposed binary tree traversal key derivation approach, the sample of size 1600000 bits passes randomness tests but when the size of data increases the tests does not show expected p_values (which should be less than 0.01). In future, we should find out the reason of lower p_values and improve the key generation scheme.

References

[1] "IoT Basics | opentechdiary," 2017. [Online]. Available: https://opentechdiary.wordpress.com/category/iot-basics/. [Accessed 25 Marchi 2017].

[2] "Radio Frequency Identification - Ethicalhavoc.net," 2017. [Online]. Available: http://www.ethicalhavoc.net/Thread-Radio-Frequency-Identification-RFID. [Accessed 25 March 2017].

[3] "Auto-ID Labs," [Online]. Available: http://www.autoidlabs.org.. [Accessed 1 April 2017].

[4] Y. Huang and G. Li, "Descriptive Models for Internet of Things," in Proc. *International Conference on Intelligent Control anf Information Processing, ICICIP*, 2010, pp. 483-486.

[5] A. Bassi and G. Horn, "Internet of Things in 2020: A Roadmap for the Future," *European Commission: Information Society and Media.,* 2008.

[6] L. Atzori, A. Iera and G. Morabito, "The Internet of Things: A survey," *Computer Networks,* vol. 54, pp. 2787–2805, 2010.

[7] V. Potdar, A. Sharif and E. Chang, "Wireless Sensor Networks: A Survey," in Proc. *International Conference on Advanced Information Networking and Applications Workshops*, 2009, pp. 393-422.

[8] "Internet Engineering Task Force, RFC 4944," 2007. [Online]. Available: https://tools.ietf.org/html/rfc4944. [Accessed 1 April 2017].

[9] S. Farahani, ZigBee wireless networks and transceivers, newnes, 2011, pp. 1-23.

[10] B. Fouladi and S. Ghanoun, "Security Evaluation of the Z-Wave Wireless Protocol" Black hat USA , 2013, pp. 1-6.

[11] A. Al-Fuqaha, M. Guizani and Mohammed Aledhari, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications" *IEEE COMMUNICATION SURVEYS & TUTORIALS,* vol. 17, no. 4, pp. 2347-2376, 2015.

[12] "Radio Frequency Identification," En.wekipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Radio-frequency_identification. [Accessed 4 April 2017].

[13] K. Finkenzeller, RFID Handbook, John Wiley & Sons, 1999.

[14] D. Vuković, "Security Issues in Internet of Things(IoT) related passive RFID tags", Facta Universitatis, Series: Automatic Control and Robotics, vol.13, no. 2, p. 97-105, 2014.

[15] M. Chen and S. Chen, " Lightweight Anonymous RFID Authentication " in *RFID technologies for Internet of Things*, 2016, pp. 39-65.

[16] L. Soo-Young, K.-Y. Kang, Soo-Young and I.-Y. Lee., "A Study on low-cost RFID system management with mutual authentication scheme in ubiquitous," in Proc. *Managing Next Generation Networks and Services. APNOMS*, 2007, pp. 492-502.

[17] Peris-Lopez, A. Mitrokotsa, M. Beye and Pedro, "Classification of RFID Threats based on Security Principles", Security Lab, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology (TU Delft), 2011.

[18] A. Furness, "CASAGRAS and the Internet of Things" in GRIFS/CASAGRAS Workshop, 2008.

[19] L. Tan and N. Wang, "Future Internet: The Internet of Things" in Proc. *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, 2010, pp. V5-376-V5-380 .

[20] M. Wu, T. Lu, F. Ling, J. Sun and H. Du, "Research on the Architecture of Internet of Things" in Proc. *International Conference on Advanced Computer Theory and Engineering (ICACTE)*, 2010, pp. V5-484-V5-487 .

[21] R. Khan, S. U. Khan, R. Zaheer and S. Khan, "Future Internet: The Internet of Things Architecture" in Proc. *10th International Conference on Frontiers of Information Technology*, 2012, pp. 257-260.

[22] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia and Mischa Dohler, "Standardized Protocol Stack for the Internet of (Important) Things," *IEEE COMMUNICATIONS SURVEYS & TUTORIALS,* vol. 13, no. 3, pp. 1389-1405, 2013.

[23] Z. Shelby, K. Hartke, C. Bormann and B. Frank, "Constrained Application Protocol (CoAP).draft-ietf-core-coap-18" *Internet Eng. Task Force,* 2013.

[24] C. Bormann, A. P. Castellani and Z. Shelby, "CoAP: An application protocol for billions of tiny Internet nodes" *IEEE Internet Computing,* vol. 16, no. 2, p. 62-67, 2012.

[25] D. Locke, "MQ telemetry transport (MQTT) v3. 1 protocol specification" IBM DeveloperWork, 2010. [Online]. Available: http://www.Ibm.Com/Developerworks/Webservices/Library/Ws-Mqtt/Index.html. [Accessed 12 12 2016].

[26] P. Saint-Andre, "Extensible messaging and presence protocol (XMPP): Core" *Internet Eng. Task Force (IETF),* 2011.

[27] M. T. Jones, "Meet the Extensible Messaging and Presence Protocol," *IBM DeveloperWorks,* 2009.

[28] "OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0,”," *Adv. Open Std. Inf. Soc. (OASIS),* 2012.

[29] "Data distribution services specification, V1.2, Object Manage. Group," April 2015. [Online]. Available: www.omg.org/spec/DDS/1.2/.

[30] J. Vasseur, "RPL: The IP routing protocol designed for low," *Internet Protocol for Smart Objects (IPSO),* 2011.

[31] N. Kushalnagar, G. Montenegro and C. Schumacher, "IPv6 over low-power wireless personal area networks (6LoWPANs): overview, assumptions, problem statement, and goals," *No. RFC 4919,* 2007.

[32] M. Krochmal and S. Cheshire, "DNS-based service discovery," *International Eng. Task Force (IETF),* 2013.

[33] S. Cheshire and M. Krochmal, "Multicast DNS," *Internet Eng. Task Force (IETF),* 2013.

[34] "[EPCglobal | GS1," Gs1.org, 2017. [Online]. Available: http://www.gs1.org/epcglobal. [Accessed 4 April 2017].

[35] EPCglobal, "EPCglobal Class Definations," 2007.

[36] EPCglobal, "EPC Compliant Class-1 Generation-2 UHF," 2015.

[37] S. Kinoshita, M. Ohkubo, F. Hoshino, G. Morohashi, O. Shionoiri and A. Kanai, "*Privacy Enhanced Active RFID Tag",* Cognitive Science Research Paper-University of Sussex, 2005.

[38] H. Bock, M. B. M. Dichtl, E. Hess, J. Heyszl, W. Kargl, H. Koroschetz, B. Meyer and H-. Seuschek, "*A milestone towards RFID products offering asymmetric authentication based on elliptic curve cryptography",* Invited talk at RFIDsec, 2008.

[39] Y.-P. Liao and C.-M. Hsiao, "A secure ECC-based RFID authentication scheme", *Ad Hoc Networks,* vol. 18, pp. 133-146, 2014.

[40] Z. Zhao, "A Secure RFID Authentication Protocol for Healthcare," *Journal of Medical Systems,* 2014.

[41] Y. K. Lee, L. Batina and I. Verbauwhede, "EC-RAC (ECDLP Based Randomized Access Control): Provably Secure RFID authentication protocol" in Proc. *IEEE International Conference on RFID*, 2008, pp. 97-104.

[42] J. Bringer, T. Icart and H. Chabanne, "Cryptanalysis of EC-RAC, a RFID Identification Protocol." in *Cryptology and Network Security. CANS* , pp. 149-161, 2008.

[43] Y. K. Lee, L. Batina and I. Verbauwhede, "Untraceable RFID authentication protocols: Revision of EC-RAC," in Proc. *IEEE International Conference on RFID*, 2009, pp. 178-185 .

[44] M. Sandhya and T. R. Rangaswamy, "A Combined Approach of Elliptic Curve and Zero Knowledge base Forward Security Protocol," World Academy of Science, Engineering and Technology , p. 847-852, 2009.

[45] S. Martínez, M. Valls, C. Roig, J. M. Miret and F. Giné., "A secure elliptic curve-based RFID protocol," *Journal of Computer Science and Technology,* vol. 24, p. 309-318, 2009.

[46] H. Gross, E. Wenger, H. Martín and M. Hutter, "PIONEER—a prototype for the internet of things based on an extendable EPC Gen2 RFID tag," in Proc. *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, 2014, pp. 54-73.

[47] H. Gross, M. Hölbl, D. Slamanig and R. Spreitzer, "Privacy-Aware Authentication in the Internet of Things," in Proc. *International Conference on Cryptology and Network Security*, 2015, pp. 32-39.

[48] W. Zhu, J. Yu and T. Wang, "A security and privacy model for mobile RFID systems in the internet of things," in Proc. *IEEE 14th International Conference on Communication Technology*, 2012., pp. 726-732.

[49] I. Erguler, "A potential weakness in RFID-based Internet-of-things systems," *Pervasive and Mobile Computing,* vol. 20, pp. 115-126, 2015.

[50] S. A. Weis, S. E. Sarma, R. L. Rivest and D. W. Engels, "Security and privacy on low-cost Radio Frequency Identification systems," *International Journal of Security and Networks,* vol. 5, no. 2-3, pp. 128-134, 2010.

[51] H.-Y. Chien, "Secure Access Control Schemes for RFID Systems with Anonymity," in Proc. *7th International Conference on Mobile Data Management, IEEE*, 2006, pp. 10-12.

[52] Q. Dong, J. Zhan and L. Wei, "A SHA-3 Based RFID Mutual Authentication," in Proc. *IEEE International Conference on Signal Processing, Communication and Computing*, 2013, pp. 1-5.

[53] M. H. Dehkordi and Y. Farzaneh, "Improvement of the Hash-Based RFID Mutual Authentication Protocol," *Wireless Personal Communications,* vol. 75, no. 1, pp. 219-232, March 2014.

[54] W. I. Khedr, "SRFID: A hash-based security scheme for low cost RFID systems," *Egyptian Informatics Journal,* vol. 14, no. 1, pp. 89-98, 2013.

[55] M. H. Habibi and M. Gardeshi, "Cryptanalysis and improvement on a new RFID mutual authentication protocol compatible with EPC standard," in *8th International ISC Conference on Information Security and Cryptology*, 2011.

[56] S. M. Alavi, K. Baghery, B. Abdolmaleki and M. R. Aref, "Traceability Analysis of Recent RFID Authentication Protocols," *Wireless Personal Communications,* vol. 83, no. 3, pp. 1663-1682, 2015.

[57] J.-S. Cho, Y.-S. Jeongb and S. O. Park, "Consideration on the brute-force attack cost and retrieval cost: A hash-based radio-frequency identification (RFID) tag mutual authentication protocol," *Computers & Mathematics with Applications,* vol. 69, no. 1, pp. 58-65, 2015.

[58] M. Safkhani, P. Peris-Lopez, J. C. Hernandez-Castro and N. Bagheri, "Cryptanalysis of the Cho et al. protocol: A hash-based RFID tag mutual authentication protocol," *Journal of Computational and Applied Mathematics,* vol. 259, pp. 571-577, 2014.

[59] B. Song and C. J. Mitchell, "RFID Authentication Protocol for Low-cost Tags," in Proc. *ACM conference on Wireless network security*, 2008, pp. 140-147.

[60] S. Cai, Y. L. Li and R. H. Deng, "Attacks and Improvements to an RFID Mutual authentication Protocol," in Proc. *ACM conference on Wireless network security*, 2009, pp.51-58.

[61] G. Tsudik, "YA-TRAP: Yet another trivial RFID authentication protocol," in Proc. *4th Annual IEEE International Conference Pervasive Computing and Communications Workshops, PerCom Workshops*, 2006, pp. 4 pp.-643.

[62] J. Ha, S. Moon, J. Zhou and J. Ha, "A New Formal Proof Model for RFID Location Privacy," in Proc. *European Symposium on Research in Computer Security*, 2008, pp. 267-281.

[63] D.-Z. Sun and J.-D. Zhong, "A Hash-Based RFID Security Protocol for Strong Privacy Protection," *IEEE Transactions on Consumer Electronics,* vol. 58, no. 4, November 2012.

[64] G. Gódor and S. Imre, "Hash-Based Mutual Authentication Protocol for Low-Cost RFID Systems," in *Meeting of the European Network of Universities and Companies in Information and Communication Engineering,* 2012, pp. 76-87.

[65] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador and A. Ribagorda, "EMAP: An Efficient Mutual-Authentication Protocol for Low-Cost RFID Tags," in Proc. *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"* 2006, pp. 352-361.

[66] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador and A. Ribagorda, "M2AP: A Minimalist Mutual-Authentication Protocol for Low-Cost RFID Tags" in Proc. *International Conference on Ubiquitous Intelligence and Computing*, 2006, pp. 919-923.

[67] P. Peris-Lopez, J. C. Hernandez-Castro and J. M. E. Tapiador, "LMAP: A Real Lightweight Mutual," in Proc. 2nd Workshop on RFID security, 2006, pp, 06.

[68] H.-Y. Chien, "SASI: A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity," *IEEE Transactions on Dependable and Secure Computing,* vol. 4, no. 4, pp. 337 - 340, November 2007.

[69] U. Mujahid, r. Najam-ul-Islam and S. Sarwar, "A New Ultralightweight RFID Authentication Protocol for Passive Low Cost Tags: KMAP," *Wireless Personal Communications,* pp. 1-20, 2016.

[70] U. Mujahid, M. Najam-ul-Islam and M. A. Shami, "RCIA: A new ultralightweight RFID authentication," *International Journal Of Distributed Sensor Networks,* January 2015.

[71] H. Luo, G. Wen, J. Su and Z. Huang., "SLAP: Succinct and lightweight authentication protocol for low-cost," *Wireless Networks*, vol. 94, no 23*,* pp. 1-10, 2016.

[72] M. Safkhani and N. Bagheri, "Generalized Desynchronization Attack on UMAP Application to RCIA, KMAP, SLAP and SASI+ protocols," IACR Cryptology ePrint Archive, 2014.

[73] M. Chen and S. Chen, "An Efficient Anonymous Authentication Protocol for RFID Systems Using Dynamic Tokens," in Proc. *IEEE 35th International Conference on Distributed Computing Systems*, 2015, pp. 756-757.

[74] M. a. S. C. Chen, "ETAP: Enable Lightweight Anonymous RFID authentication with O (1) overhead," in Proc. *IEEE 23rd International Conference on Network Protocols (ICNP)*, 2015, pp. 267-278.

[75] M. Rahman, R. V. Sampangi and S. Sampalli, "Lightweight Protocol for Anonymity and Mutual," in Proc. *IEEE 12th Consumer Communications and Networking Conference (CCNC): CCNC 2015 Workshops - IEEE CCAN*, 2015, pp. 910-915.

[76] J.-P. Aumasson, L. H. Meier and M. Naya-Plasencia, "Quark: A Lightweight Hash," *Journal of Cryptology,* vol. 26, no. 2, pp. 313-339, 2013.

[77] J. Guo, T. Peyrin and A. Poschmann, "The PHOTON Family of Lightweight Hash Functions," in Proc. *Annual Cryptology Conference*, 2011, pp. 222-239.

[78] A. Bogdanov, M. Knezevic, G. Leander, D. Toz, K. Varici and I. Verbauwhede, "SPONGENT: The Design Space of Lightweight Cryptographic Hashing," *IEEE Transactions on Computers,* vol. 62, no. 10, pp. 2041 - 2053, 2011.

[79] G. Bertoni, J. Daemen, M. Peeters and G. V. Assche, "Sponge Functions," *ECRYPT hash workshop,* 2007.

[80] A. Bogdanov, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw and Y. Seurin, "Hash Functions and RFID Tags: Mind the Gap," in Proc. *International Workshop on Cryptographic Hardware and Embedded Systems* , 2008, pp. 283-299.

[81] E. B. Kavun and T. Yalcin, "A Lightweight Implementation of Keccak Hash Function for Radio-Frequency Identification Applications," in Proc. *International Workshop on Radio Frequency Identification: Security and Privacy Issues* 2010, pp. 258-269.

[82] M. Kim and J. Ryou, "Power Efficient Hardware Architecture of SHA-1 Algorithm for Trusted Mobile Computing," in Proc. *International Conference on Information and Communications Security,* 2007, pp. 375-385.

[83] L. Henzen, J.-P. Aumasson, W. Meier and R. C.-W. Phan, "VLSI Characterization of the Cryptographic Hash Function BLAKE," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 19, no. 10, pp. 1746 - 1754, 2011.

[84] D. Boneh, X. Boyen and E.-J. Goh, "Hierarchical Identity Based Encryption with Constant Size Ciphertext," in Proc. *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2005, pp. 440-456.

[85] R. Canetti, S. Halevi and J. Katz, "A Forward-Secure Public-Key Encryption Scheme.," *Journal of Cryptology ,* vol. 20, no. 3, pp. 265-294, 2007.

[86] L. Batina, J. Guajardo and T. Kerins, *An Elliptic Curve Processor Suitable For RFID-Tags,* IACR Cryptology ePrint Archive, 2006.

[87] Y. Chen, J.-S. Chou and Hung-Min Sun, "A novel mutual authentication scheme based on quadratic residues for RFID systems," *Computer Networks,* vol. 52, no. 12, p. 2373–2380, 2008.

[88] A. Juels and S. A. Weis, "Authenticating Pervasive Devices with Human Protocols," in Proc. *Annual International Cryptology Conference*, 2005, pp. 293-308.

[89] Y. Naito, "Sandwich Construction for Keyed Sponges: Independence Between Capacity and Online Queries," in Proc. *International Conference on Cryptology and Network Security*, 2016, pp. 245-261.

[90] W. Diffie and M.- . Hellman, "Multiuser cryptographic techniques," in Proc. *National Computer Conference and Exposition*, New York, 1976, pp. 109-112.

[91] "Tree Traversals (Inorder, Preorder and Postorder)," 2017. [Online]. Available: http://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/. [Accessed 17 April 2017].

[92] "Overview of Java SE Security," Docs.oracle.com, 2017. [Online]. Available: http://docs.oracle.com/javase/7/docs/technotes/guides/security/overview/jsoverview.html. [Accessed 09 05 2017].

[93] "SPONGENT," sites.google, 2017. [Online]. Available: https://sites.google.com/site/spongenthash/. [Accessed 25 02 2017].

[94] "Oracle- Class Secure Random," Oracle, 2017. [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/security/SecureRandom.html. [Accessed 09 05 2017].

[95] "RawCap," Netresec, [Online]. Available: http://www.netresec.com/?page=RawCap.

[96] "WireShark," 2017. [Online]. Available: https://www.wireshark.org/. [Accessed 02 02 2017].

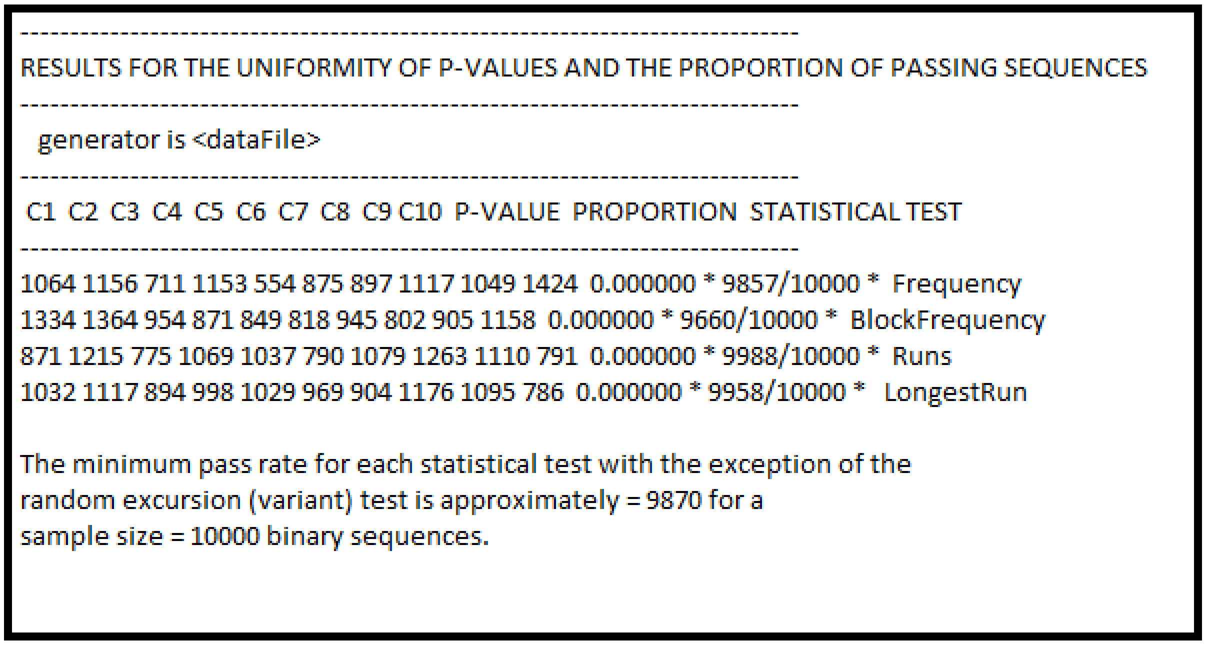
[97] C. Cremers, "The Scyther Tool: Verification, Falsification,," 2008.

[98] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh and M. Levenson, "A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications," NIST Special Publication, 2010.

[99] C. Paar, A. Poschmann and M. Robshaw, "New Designs in Lightweight Symmetric encryption," in RFID security, Springer US, 2008, pp. 349-371.

[100] M. Mouly and M.-B. Pautet, " The GSM system for mobile communications", Telecom Publishing, 1992.

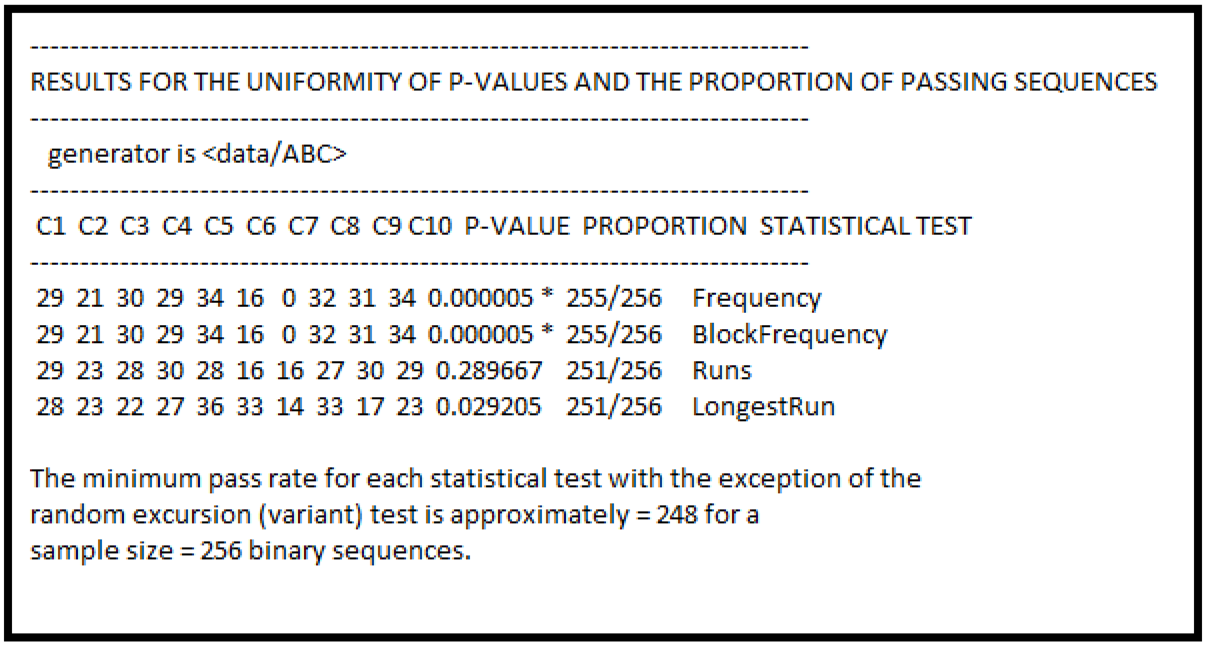
APPENDIX A



```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
  generator is <dataFile>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
1064 1156 711 1153 554 875 897 1117 1049 1424  0.000000 * 9857/10000 *  Frequency
1334 1364 954 871 849 818 945 802 905 1158  0.000000 * 9660/10000 *  BlockFrequency
871 1215 775 1069 1037 790 1079 1263 1110 791  0.000000 * 9988/10000 *  Runs
1032 1117 894 998 1029 969 904 1176 1095 786  0.000000 * 9958/10000 *   LongestRun

The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 9870 for a
sample size = 10000 binary sequences.
```

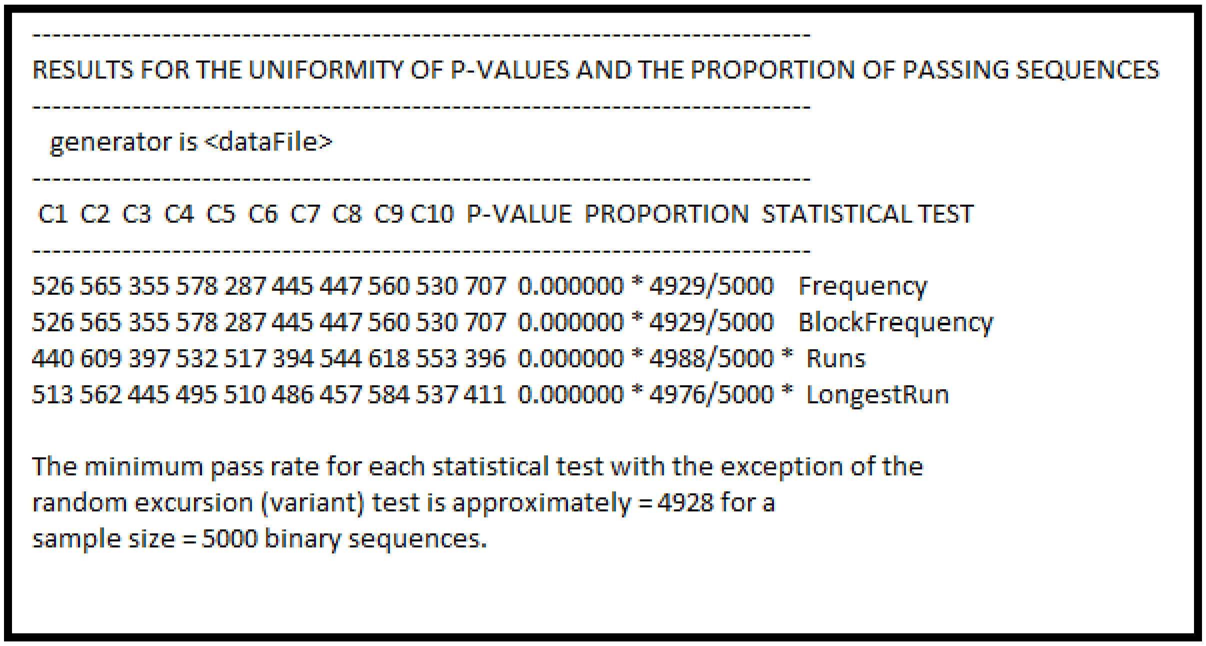
Figure 42 Randomness test using STS with block size 256 and sample data size 2560000



```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
  generator is <data/ABC>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
 29  21  30  29  34  16   0  32  31  34  0.000005 *  255/256    Frequency
 29  21  30  29  34  16   0  32  31  34  0.000005 *  255/256    BlockFrequency
 29  23  28  30  28  16  16  27  30  29  0.289667    251/256    Runs
 28  23  22  27  36  33  14  33  17  23  0.029205    251/256    LongestRun

The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 248 for a
sample size = 256 binary sequences.
```

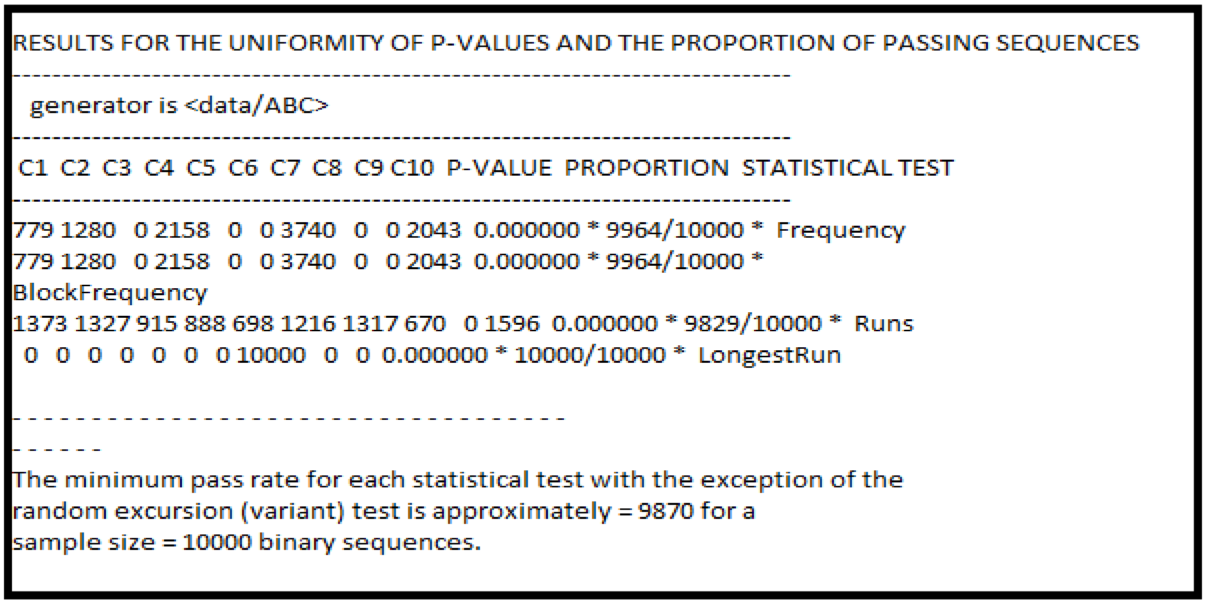
Figure 43 Randomness test using STS for block size 128 and sample data size 33000



```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
  generator is <dataFile>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
526 565 355 578 287 445 447 560 530 707  0.000000 * 4929/5000    Frequency
526 565 355 578 287 445 447 560 530 707  0.000000 * 4929/5000    BlockFrequency
440 609 397 532 517 394 544 618 553 396  0.000000 * 4988/5000 *  Runs
513 562 445 495 510 486 457 584 537 411  0.000000 * 4976/5000 *  LongestRun

The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 4928 for a
sample size = 5000 binary sequences.
```

Figure 44 Randomness Test using STS for block size 256-bits and sample data of 1280000-bits

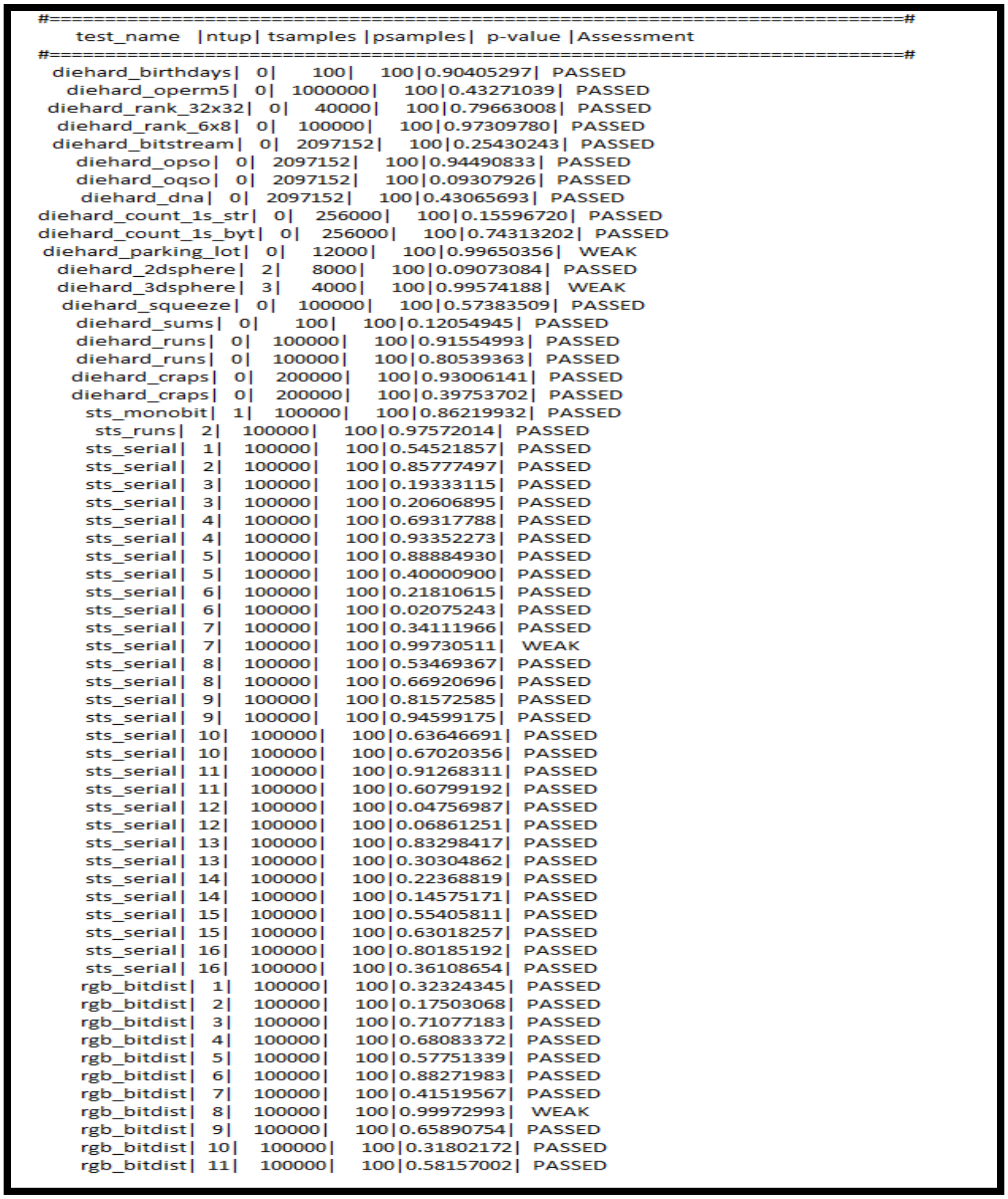


```
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
  generator is <data/ABC>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
779 1280   0 2158   0   0 3740   0   0 2043  0.000000 * 9964/10000 *  Frequency
779 1280   0 2158   0   0 3740   0   0 2043  0.000000 * 9964/10000 *
BlockFrequency
1373 1327 915 888 698 1216 1317 670   0 1596  0.000000 * 9829/10000 *  Runs
 0   0   0   0   0   0   0 10000   0   0  0.000000 * 10000/10000 *  LongestRun

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - -
The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 9870 for a
sample size = 10000 binary sequences.
```

Figure 45 Randomness Test using STS for block size 16-bits and sample size 160000 bits.

APPENDIX B



```
#=============================================================================#
    test_name   |ntup| tsamples |psamples|  p-value |Assessment
#=============================================================================#
  diehard_birthdays|   0|       100|     100|0.90405297|  PASSED
     diehard_operm5|   0|   1000000|     100|0.43271039|  PASSED
 diehard_rank_32x32|   0|     40000|     100|0.79663008|  PASSED
   diehard_rank_6x8|   0|    100000|     100|0.97309780|  PASSED
  diehard_bitstream|   0|   2097152|     100|0.25430243|  PASSED
       diehard_opso|   0|   2097152|     100|0.94490833|  PASSED
       diehard_oqso|   0|   2097152|     100|0.09307926|  PASSED
        diehard_dna|   0|   2097152|     100|0.43065693|  PASSED
diehard_count_1s_str|   0|    256000|     100|0.15596720|  PASSED
diehard_count_1s_byt|   0|    256000|     100|0.74313202|  PASSED
 diehard_parking_lot|   0|     12000|     100|0.99650356|   WEAK
    diehard_2dsphere|   2|      8000|     100|0.09073084|  PASSED
    diehard_3dsphere|   3|      4000|     100|0.99574188|   WEAK
     diehard_squeeze|   0|    100000|     100|0.57383509|  PASSED
        diehard_sums|   0|       100|     100|0.12054945|  PASSED
        diehard_runs|   0|    100000|     100|0.91554993|  PASSED
        diehard_runs|   0|    100000|     100|0.80539363|  PASSED
       diehard_craps|   0|    200000|     100|0.93006141|  PASSED
       diehard_craps|   0|    200000|     100|0.39753702|  PASSED
         sts_monobit|   1|    100000|     100|0.86219932|  PASSED
            sts_runs|   2|    100000|     100|0.97572014|  PASSED
          sts_serial|   1|    100000|     100|0.54521857|  PASSED
          sts_serial|   2|    100000|     100|0.85777497|  PASSED
          sts_serial|   3|    100000|     100|0.19333115|  PASSED
          sts_serial|   3|    100000|     100|0.20606895|  PASSED
          sts_serial|   4|    100000|     100|0.69317788|  PASSED
          sts_serial|   4|    100000|     100|0.93352273|  PASSED
          sts_serial|   5|    100000|     100|0.88884930|  PASSED
          sts_serial|   5|    100000|     100|0.40000900|  PASSED
          sts_serial|   6|    100000|     100|0.21810615|  PASSED
          sts_serial|   6|    100000|     100|0.02075243|  PASSED
          sts_serial|   7|    100000|     100|0.34111966|  PASSED
          sts_serial|   7|    100000|     100|0.99730511|   WEAK
          sts_serial|   8|    100000|     100|0.53469367|  PASSED
          sts_serial|   8|    100000|     100|0.66920696|  PASSED
          sts_serial|   9|    100000|     100|0.81572585|  PASSED
          sts_serial|   9|    100000|     100|0.94599175|  PASSED
          sts_serial|  10|    100000|     100|0.63646691|  PASSED
          sts_serial|  10|    100000|     100|0.67020356|  PASSED
          sts_serial|  11|    100000|     100|0.91268311|  PASSED
          sts_serial|  11|    100000|     100|0.60799192|  PASSED
          sts_serial|  12|    100000|     100|0.04756987|  PASSED
          sts_serial|  12|    100000|     100|0.06861251|  PASSED
          sts_serial|  13|    100000|     100|0.83298417|  PASSED
          sts_serial|  13|    100000|     100|0.30304862|  PASSED
          sts_serial|  14|    100000|     100|0.22368819|  PASSED
          sts_serial|  14|    100000|     100|0.14575171|  PASSED
          sts_serial|  15|    100000|     100|0.55405811|  PASSED
          sts_serial|  15|    100000|     100|0.63018257|  PASSED
          sts_serial|  16|    100000|     100|0.80185192|  PASSED
          sts_serial|  16|    100000|     100|0.36108654|  PASSED
         rgb_bitdist|   1|    100000|     100|0.32324345|  PASSED
         rgb_bitdist|   2|    100000|     100|0.17503068|  PASSED
         rgb_bitdist|   3|    100000|     100|0.71077183|  PASSED
         rgb_bitdist|   4|    100000|     100|0.68083372|  PASSED
         rgb_bitdist|   5|    100000|     100|0.57751339|  PASSED
         rgb_bitdist|   6|    100000|     100|0.88271983|  PASSED
         rgb_bitdist|   7|    100000|     100|0.41519567|  PASSED
         rgb_bitdist|   8|    100000|     100|0.99972993|   WEAK
         rgb_bitdist|   9|    100000|     100|0.65890754|  PASSED
         rgb_bitdist|  10|    100000|     100|0.31802172|  PASSED
         rgb_bitdist|  11|    100000|     100|0.58157002|  PASSED
```

Figure 46 Randomness Testing using Dieharder testing tool for sample size 160000 bits