RESPONSE TIME IMPROVEMENT ON ONE TIME PASSWORD (OTP) TECHNIQUE TO PREVENT REPLAY ATTACK IN A RADIUS ENVIRONMENT.

BY

Yusuf ABDULLAHI

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

FACULTY OF ENGINEERING AHMADU BELLO UNIVERSITY ZARIA, NIGERIA

JANUARY, 2017

RESPONSE TIME IMPROVEMENT ON ONE TIME PASSWORD (OTP) TECHNIQUE TO PREVENT REPLAY ATTACK IN A RADIUS ENVIRONMENT.

BY

Yusuf ABDULLAHI

(P15EGCP8021)

A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES, AHMADU BELLO UNIVERSITY, ZARIA
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF A MASTER OF SCIENCE (MSc) DEGREE IN COMPUTER ENGINEERING

,,

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

FACULTY OF ENGINEERING

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA

FEBRUARY, 2017

DECLARATION

I Yusuf Abdullahi declare that the work in this dissertation titled "**Response Time Improvement on One Time password (OTP) technique to prevent replay attack in a RADIUS environment**" has been carried out by me in the Department of Electrical and Computer Engineering, Ahmadu Bello University Zaria. The information derived from the literature has been duly acknowledged in the text and a list of references provided. No part of this thesis was previously presented for another degree or diploma at this or any other institution.

Yusuf Abdullahi _____ _____

Signature Date

CERTIFICATION

This dissertation titled "**RESPONSE TIME IMPROVEMENT ON ONE TIME PASSWORD (OTP) TECHNIQUE TO PREVENT REPLAY ATTACK IN A RADIUS ENVIRONMENT** "by Yusuf Abdullahi meets the regulations governing the award of the degree Master of Science (M.Sc.) of the Ahmadu Bello University, and is approved for its contribution to knowledge and literary presentation.

Prof. M. B. Muazu	_____	_____
Chairman, Supervisory Committee	Signature	Date
Dr. E. A. Adedokun	_____	_____
Member, Supervisory Committee	Signature	Date
Dr. Y. Jibril	_____	_____
Head of Department	Signature	Date
Prof. S. Z. Abubakar	_____	_____
Dean, School of Postgraduate Studies	Signature	Date

DEDICATION

This research work is dedicated to my Late Father Alhaji Abdullahi Malumfashi. May Allah in his infinite mercy reward him with Jannatul Firdausi, Ameen.

ACKNOWLEDGEMENT

Thanks to almighty God, the first and the last. Many thanks and sincere gratefulness goes to my supervisors Prof. M.B Muazu and Dr. Adewale Adedokun. I am indebted to their support, supervision and guidance. I would like to thank my late father, mother, dear wife, brothers and sisters for their support which gave me the strength and enthusiasm during the whole period of this thesis. I must mention my elder brother General Ismaila Abdullahi (rtd) for his encouragement and support from my undergraduate to this level.

I am most grateful to all my colleagues in the Network Infrastructure and Security Unit of the Institute of Computing and ICT, Ahmadu Bello University Zaria, the Head of the Unit Mal. M.B Sani, Alhassan Babajo, Mr. Akin Okinbaloye, Engr. Abdussamad Musa, Engr. Bashir Sani, Engr. Ibrahim Ladan, Engr. Ovie Oteri, Mr. John Dayok, Mr. Christopher Anthony and the rest of others whose names were not mentioned here, for their encouragement.

I am also grateful to all my lecturers in the Department of Electrical and Computer Engineering, Ahmadu Bello University Zaria.

ABSTRACT

This research is aimed at the modification of the Remote Access Dial in User Server (RADIUS) protocol with the one-time password (OTP) technique for the authentication environment with a captive portal to prevent replay attacks. One of the important network security measures on a campus network is the use of authentication for identification of legitimate users and one of the most widely used solution in network authentication is the RADIUS protocol. However, there are potential security vulnerabilities in the RADIUS network especially for networks using captive portal, such as the replay attack. The Ahmadu Bello University (ABU) network is simulated using the Graphical Network Simulator (GNS3) software on a virtualized environment using Virtualbox. An OTP generator was developed using Hypertext Processor (PhP) programming language for the three variants of the OTP, Time One Time Password (TOTP), Challenge Response One Time Password (CROTP) and Hash One Time Password (HOTP). Before improvement on the OTP technique using a PhP developed script, the result obtained shows the average response time for TOTP, CROTP and HOTP as 2.5s, 5.2s and 5.7s respectively, this result showed no improvement in the TOTP, CROPT and HOTP response time respectively when compared with the recommended response time of a RADIUS server in a captive portal environment which is 1000ms. After improving the OTP technique by integrating all the variants of OTP with the RADIUS server on a single server using the simulated ABU campus network using GNS3, the result shows a significant improvement over the above results. The results obtained shows the average response time for TOTP, CROTP and HOTP as 1.2s, 2s and 1.9s. The validation, based on the developed and simulated configuration was carried out using live servers, routers and switches and the results shows improvement over the above results, the average response time for TOTP, CROTP and HOTP were obtained as 0.4s, 0.9s and 0.9s respectively. This shows significant improvement in the TOTP, CROPT and HOTP respectively. The result shows the average response time is less than the recommended 1000ms for RADIUS server response time in a captive portal environment.

TABLE OF CONTENTS

DECLARATION i
CERTIFICATION ii
DEDICATION iii
ACKNOWLEDGEMENT iv
ABSTRACT v
TABLE OF CONTENTS vi
LIST OF FIGURES ix
LIST OF ABBREVIATIONS xi
CHAPTER ONE INTRODUCTION 1
1.1 BACKGROUND 1
1.2 USER AUTHENTICATION WITH CAPTIVE PORTAL 5
1.3 PROBLEM STATEMENT 6
1.4 AIM AND OBJECTIVES 6
CHAPTER TWO LITERATURE REVIEW 8
2.1 INTRODUCTION 8
2.2 REVIEW OF FUNDAMENTAL CONCEPTS 8
2.2.1 COMPUTER SECURITY 8
2.2.1.1 Goals of computer security 8
2.2.1.2 Components of computer security 8
2.2.2 SECURITY ARCHITECTURE 9
2.2.3 RADIUS PROTOCOL 9
2.2.4 AUTHENTICATION, AUTHORIZATION AND ACCOUNTING (AAA) 10
2.2.5 SYSTEM ARCHITECTURE 11
2.2.5.1 Network access server (NAS) 12
2.2.6 PACKET FORMAT 14
2.2.7 SHARED SECRETS 17
2.2.8 THE VULNERABILITY OF RADIUS PROTOCOL 18
2.2.8.1 Attacker database 19
2.2.8.2 Replay attack components 20
2.2.8.3 Replay attack architecture 21
2.2.9 ONE TIME PASSWORD (OTP) TECHNIQUE 22

2.2.9.1 Generation OTP and distribution 23
2.2.9.2 Justification for OTP 24
2.2.9.3Approaches for the generation of OTP 24
1. TIME SYNCHRONIZATION 24
2. EVENT SYNCHRONIZATION 25
3. CHALLENGE – RESPONSE TECHNIQUE 25
2.2.9.4 Types of OTP techniques 25
1. HASH ONE TIME PASSWORD (HOTP) TECHNIQUE 25
2.2.9.5 Time one time password (TOTP) technique 28
2.2.9.5 Cryptography one time password (CROTP) technique 32
2.2.10 SIMULATION SOFTWARE 35
2.2.10.1 OMNET++ 35
2.2.10.2 Optimized network engineering tools (OPNET) 35
2.2.10.3 Network simulator version 2 (NS2) 36
2.2.10.4 Graphical network simulator-3 (GNS3) 36
2.2.11 VIRTUALIZATION OVERVIEW 37
2.2.11.1 Server virtualization 37
1. VMWARE WORKSTATION 38
2. PARALLELS WORKSTATION 38
3. VIRTUALBOX VIRTUALIZATION 39
2.2.12 PHP PROGRAMMING LANGUAGE 40
2.3 REVIEW OF SIMILAR WORKS 41
CHAPTER THREE MATERIALS AND METHODS 45
3.1 INTRODUCTION 45
3.2 METHODOLOGY 45
3.3 ABU CAMPUS NETWORK MODELING AND SIMULATION USING GNS3 45
3.4 RADIUS SERVER ENVIRONMENT 47
3.5 CAPTIVE PORTAL IMPLEMENTATION 49
3.6 OTP GENERATION SCENARIOS 50
3.7 OBTAINING USERNAMES, PIN AND OTP 51
3.8 USER LOGIN SCENARIO 51

3.9 THE IMPROVED OTP TECHNIQUE 53
3.10 VALIDATION 56
CHAPTER FOUR RESULTS AND DISCUSSION 59
4.1 INTRODUCTION 59
4.2 DATA COLLECTION AND ANALYSIS 59
4.3 IMPROVED OTP TECNIQUE 63
4.4 VALIDATION 67
CHAPTER FIVE CONCLUSIONS AND RECOMMENDATIONS 70
5.1 INTRODUCTION 70
5.2 SIGNIFICANT CONTRIBUTION 70
5.3 CONCLUSION 70
5.4 LIMITATIONS 71
5.5 RECOMMENDATION FOR FUTURE WORK 71
REFERENCES 72
APPENDIX I 75
APPENDIX II 79

LIST OF FIGURES

Figure1.1: The Core Physical Connectivity diagram 2

Figure 1.2: Core to Distribution Physical Connectivity Diagram 2

Figure 1.3: Distribution to Access Physical Connectivity Diagram 3

Figure 1.4: Access Aggregate Physical Connectivity 4

Figure 2.1: An Illustration of Security Architecture 8

Figure 2.2: RADIUS Authentication and Authorization Flow 9

Figure 2.3: Relationship between RADIUS and AAA 10

Figure 2.4: Captive Portal Display on ABU Zaria Network 11

Figure 2.5: RADIUS Packet Format 12

Figure 2.6: RADIUS Attributes 14

Figure 2.7: Attacker database 21

Figure 2.8: Replay attack architecture 22

Figure 2.9: OTP Generation 19

Figure 2.10: HOTP Flowchart 23

Figure 2.11: Flow chart for TOTP 25

Figure 2.12: Flowchart for CROTP 27

Figure 2.13: Virtualbox Environment 32

Figure 3.1: ABU Network Model using GNS3 39

Figure 3.2: Putty SSh Client 41

Figure 3.3: Accessing RADIUS Server 41

Figure 3.4: Running RADIUS Server 42

Figure3.5: Captive Portal 43

Figure 3.6: Developed HOTP Generator 44

Figure 3.7: Developed CROTP Generator 44

Figure 3.8: Developed TOTP Generator 44

Figure 3.9: Response Time for the Three OTP Variants 45

Figure 3.10: Flowchart for the Improved OTP Technique 46

Plate 3.1 Validation Setup 47

Figure 4.1: TOTP Response time 53

Figure 4.2: HOTP Response Time 53

Figure 4.3: CROTP Response Time 53

Figure 4.4: Combined results before improvement 59

Figure 4.5: Improved TOTP Response Time 55

Figure 4.6: Improved HOTP Response Time 55

Figure 4.7: Improved CROTP Response Time 55

Figure 4.8: Combined results after improvement 62

Figure 4.9: Validation TOTP Response Time 57

Figure 4.10: Validation HOTP Response Time 57

Figure 4.11: Validation CROTP Response Time 57

LIST OF ABBREVIATIONS

Acronyms	Definition
AAA	Authentication, Authorization and Accounting
AES	Advanced Encryption Service
AS	Authentication Server
CHAP	Challenge-Handshake Protocol
CIA	Confidentiality Integrity Availability
CROTP	The OATH Challenge-Response Algorithm
DOS	Denial of Service
EAB	Extensible Authentication Protocol
EKE	Encrypted Key Exchange
EPPKE	Efficient Password-Proven Key Exchange
HOTP	Hash one-time password
IP	Internet Protocol
IPSec	IP Security
MOE	Margin of Error
NAS	Network Access Server
OTP	One-Time Password
PAKE	Password-authenticated key exchange
PAP	Password Authentication Protocol
PIN	Personal Identify Number

PKI	Public Key Infrastructure
PRNG	Pseudo Random Number Generator
PSK	Pre-Shared Key
RADIUS	Remote Authentication Dial in User Service
RSA	Ron Rivest, Adi Shamir and Leonard Adelman
SHA	Secure Hash Algorithm
SMS	Short Message Service
TCP	Transmission Control Protocol
TDES	Triple Data Encryption
TOTP	Time one-time password
UDP	User Datagram Protocol
VPN	Virtual private network

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND

The entire Ahmadu Bello University Campus Network Infrastructure, runs on fiber optic technology for transmission and is built based on the Cisco standard hierarchical design standard for campus networks (core, distribution and access levels) providing high speed and redundancy. The network is built on Cisco technology using high end devices which include Cisco Catalyst 6500 series as the core switch, 4500, 3700 and 3560 series switches as distribution switches and 2960 series and gigabit small business series switches as access switches. This setup guarantees gigabit transmission to every host on the network. The network covers all the campuses of ABU Zaria which include Samaru, Kongo, Shika and NAPRI all connected with over 60km of optical fiber cable.

The core network as in Figure1.1 is built on the Virtual Switching System technology for high capacity using ether channel technology, whereby so many fiber ports are bundled together for more bandwidth capacity on a link, all the servers are part of the core network including the authentication server. Distribution is built on Layer 3 switches the Cisco 3750G all the distribution points are connected back to the core network through a fiber link, with static addressing of point to point nodes. Dynamic routing protocol is enabled running Open Shortest Path First (OSPF), with each distribution switch used as the OSPF Area Border Router (ABR) with separate areas. The distributions also host the virtual local area network VLAN of each access layer switches.

The access layer which comprises mainly of Small Business Series switches and 2960 series switches is built on Layer 2 switching technology. Access layer devices have access to Dynamic Host Configuration Protocol (DHCP) services from the distribution switch to which the Access switch is connected.

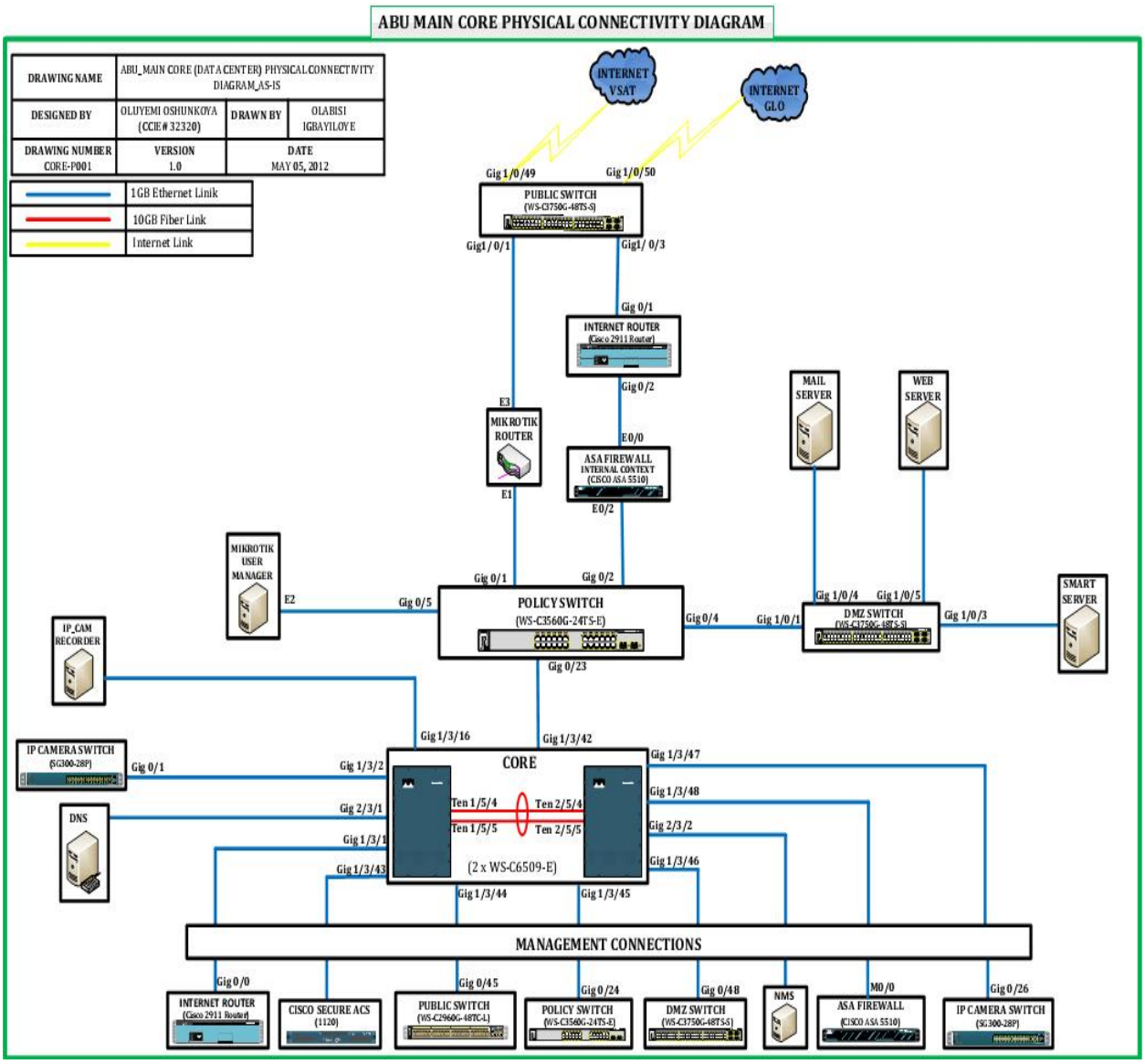


Figure 1.1: The Core Physical Connectivity Diagram. (ABU Network,2012)

Core Network: The core network as in Fig, 1.1 which provides a high-speed connectivity to the distribution network with most of link having up to 10Gbps for a single link. The core also hosts all the servers on the network includes the authentication server. If there is a failure in the core most network services will be down.

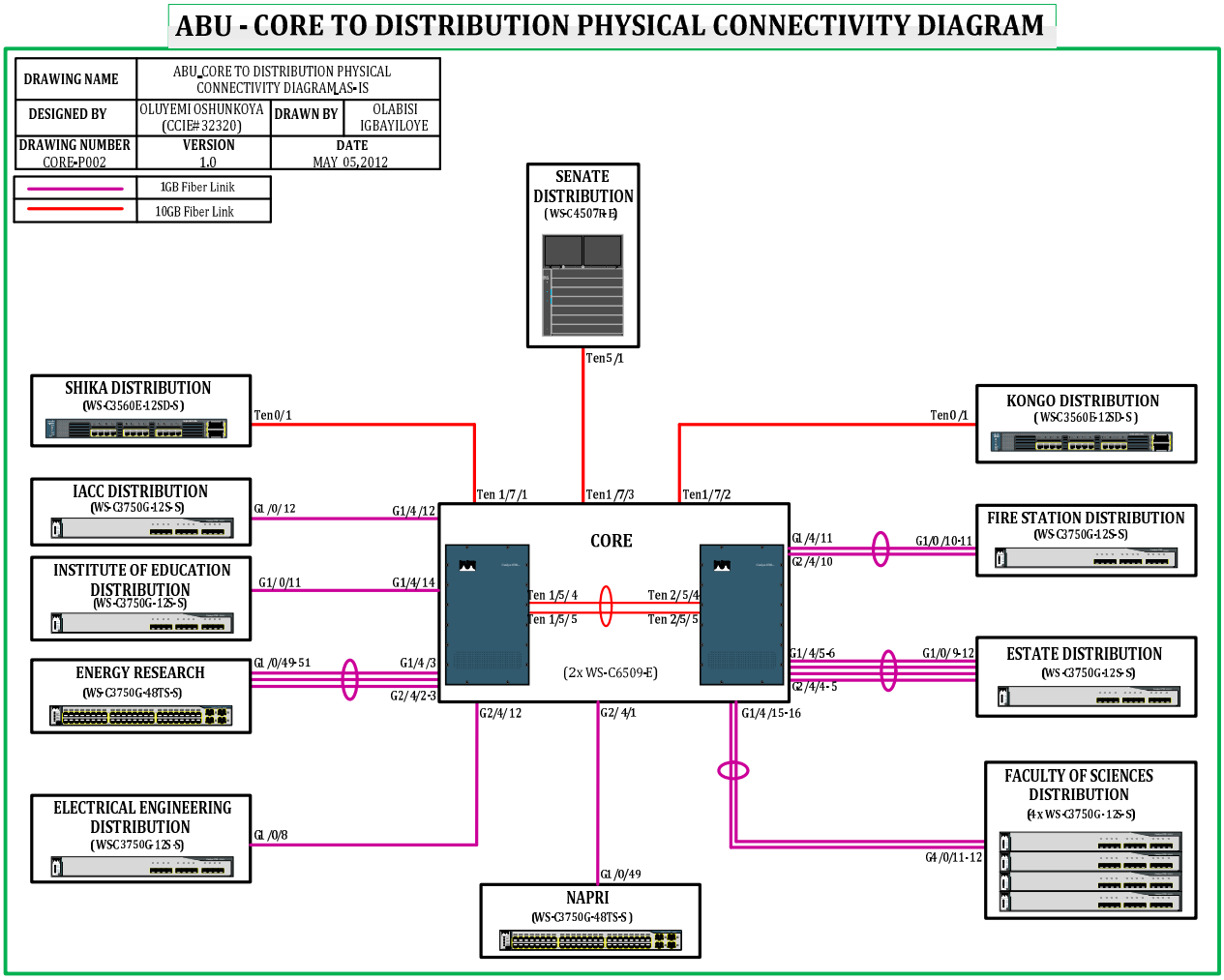


Figure 1.2: Core to Distribution Physical Connectivity Diagram. (ABU Network, 2012)

Distribution layer Network: This layer as in Figure 1.2 ensures that packets are properly routed between subnets and VLANs in the network. It is at this layer where control is exerted over network transmissions, including what comes in and what goes out of the network. Broadcast domains virtual LANs can be created and limited in this layer, if necessary, and various management tasks conducted, including obtaining route summaries. In a route summary, traffic from many subnets are consolidated into a core network connection. This layer is also called the Workgroup layer. (Lammle, 2014)

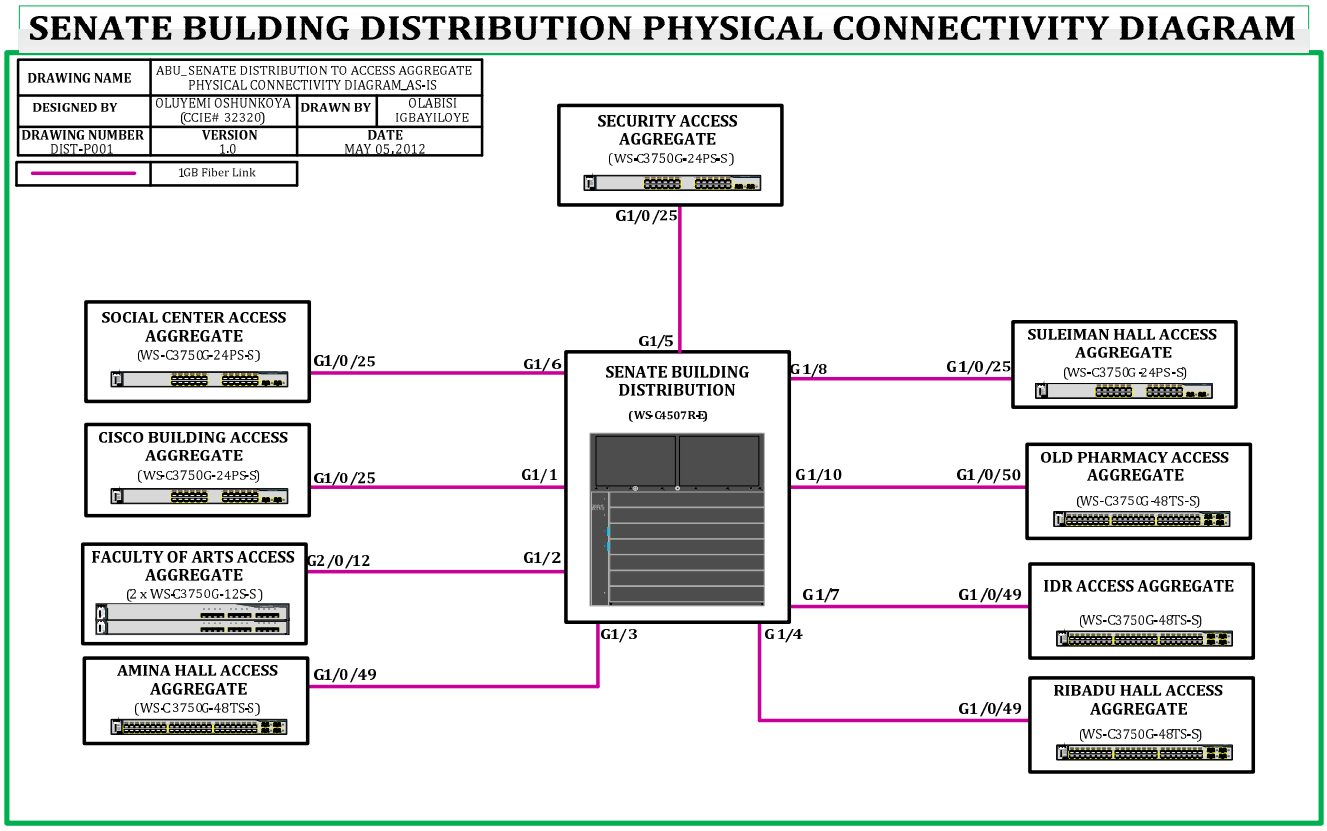


Figure 1.3: Distribution to Access Physical Connectivity Diagram. (ABU Network 2012)

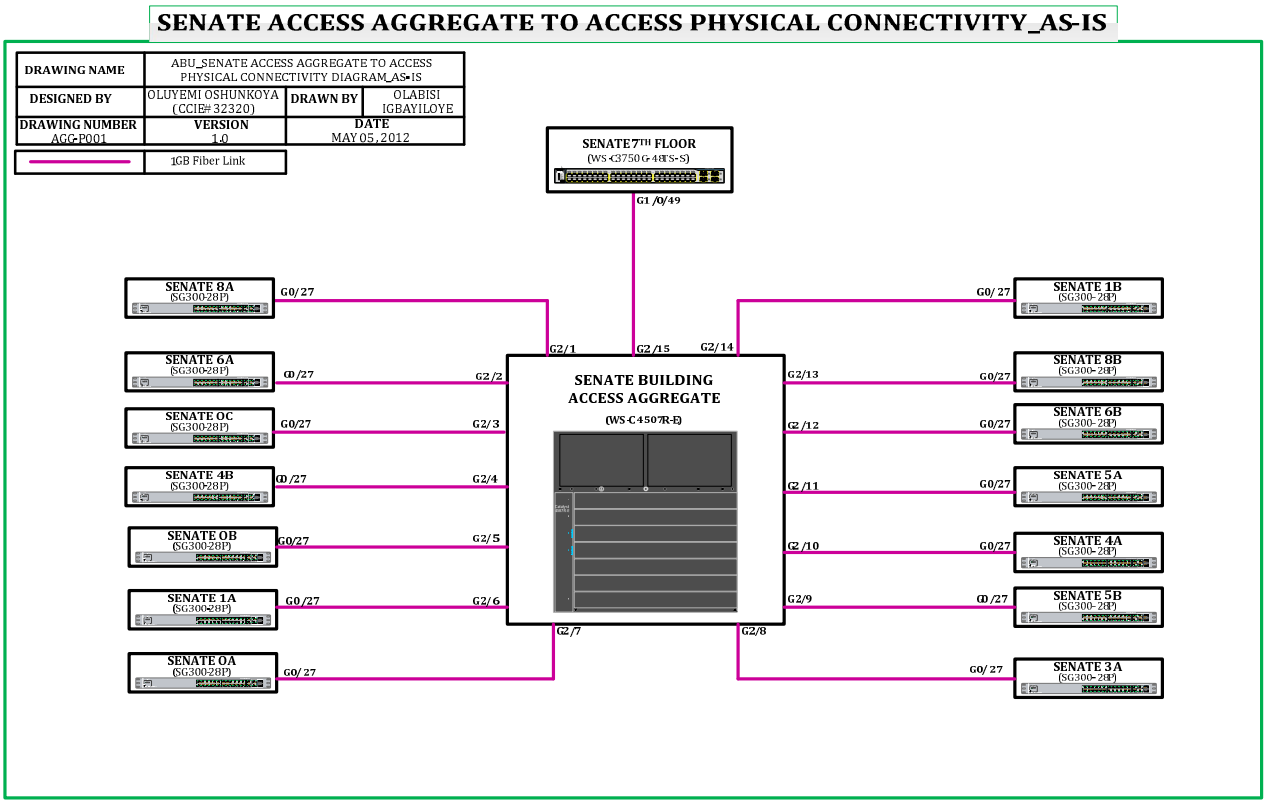


Figure 1.4: Access Aggregate Physical Connectivity. (ABU Network, 2012)

Access Layer Network: The access layer as in Figs. 1.3 and 1.4 as shown above contains devices that allow workgroups and users to use the services provided by the distribution and core layers. In the access layer, the user connects to the network through their laptops, desktops etc. This is where the users authenticate with their user name and password through any browser used.

1.2 USER AUTHENTICATION WITH CAPTIVE PORTAL

Authentication is the process of verifying a person's (or machine's) declared identity. The most common form of authentication is using a combination of login ID and a password, in which the knowledge of the password is a representation that the user is authentic. (Mikko, 2004)

The captive portal is the page that appears to the user to login with username and a password.

The emerging campus networks are migrating from a purely dedicated wired LAN infrastructure to a hybrid that incorporates both wired and wireless LAN users like the Ahmadu Bello University campus network, contends with the challenges of securing the network especially with the large number of mobile users. The use of RADIUS services is one of the most effective ways to secure the network.

This is particularly true from an access and policy enforcement perspective with the proliferation of wireless and wired devices including notebooks, laptops and desktop PC and control the amount of traffic each user is generating and billing functionalities, the challenge to ensure that staff and students are subjected to the same network enforcement management policies regardless of how they connected to the network. The challenges with using RADIUS protocol with captive portal is the issue of the replay attack and response time of the RADIUS protocol. (Alan, 2013).

This research will focus on the prevention and improvement on these issues. The Ahmadu Bello University network uses the Mikrotik operating system to serve as a user management and a captive portal. The Mikrotik user management is placed at ABU core network as shown in Figure 1.1

1.3 PROBLEM STATEMENT

Emerging campus networks are migrating from a dedicated wired LAN infrastructure to high speed hybrid campus networks that incorporates both wired as well as wireless users like the Ahmadu Bello University campus network, the challenges of securing both users and network integrity becomes more complex. One of the most effective ways of securing users access is the use a captive portal with Radius services. The implementation of RADIUS services is however bedeviled with large database overtime and this is prone to replay attacks on the network. This therefore suggests the need for multiple level of authentication on networks. One-time password (OTP) techniques are used to prevent replay attacks. There are several OTP techniques used today and this research work is aimed at analyzing and comparing three variants of the OTP namely TOTP, HOTP and CROTP in RADIUS protocol. (Amna, 2014) using the response time of each technique as the performance metric.

1.4 AIM AND OBJECTIVES

The aim of this research work is to prevent replay attack in RADIUS environment by improving response time of OTP.

The main objectives of this research work are as follows:

1) Simulation of the ABU Zaria network using GNS3 Simulator
2) Modification of the three variants of the OTP technique (TOTP, HOTP and CROTP) and selection of the best technique using the response time as the performance metric
3) Validation of the improved authentication technique by comparing its response time with that of the standard technique

1.5 METHODOLOGY

The following methodology will be adopted in carrying out this research work:

1) ABU Campus Network Modeling and Emulation using GNS3 to simulate all the switches, routers, firewall and servers on the network
2) Installation of Operating Systems (Linux), Free Radius 2.0 and Firewall
3) Implementation of network Access Server which provides the captive portal on a Virtual Machine (VM) Virtualbox is the virtualization software of choice.
4) Generate OTP for the 3 techniques using an open source generator written in PhP scripts.
5) Development of an OTP technique with the highest response time from the three available techniques of OTP, using the PhP scripts.
6) Development of an improve RADIUS Server on the network access server and generate OTP for the improved technique.

1.6 DISSERTATION ORGANIZATION

The general introduction of computer networks, ABU campus network, statement of the problem, methodology, and aims and objectives has been presented in chapter one. The rest of the chapters are presented as follows: a detailed review of the fundamental concepts of RADIUS Protocol, Captive portal, OTP Techniques, PhP programming language, GNS3, Vitrualbox, OPNET as well as a review of similar research works is presented in chapter two, detailed methodology and step by step guide in the configuration and implementations of three technique of OTP in a campus network are presented in chapter three, analysis and discussions of the results are presented in chapter four, summary, conclusions, significant contributions, limitations and recommendations are presented in chapter five.

CHAPTER TWO

LITERATURE REVIEW

2.1 INTRODUCTION

This section contains an overview of concepts fundamental to the research and a review of similar works relevant to the research.

2.2 REVIEW OF FUNDAMENTAL CONCEPTS

This section will contain an overview of concepts fundamental to the research such as the protocols, authentication techniques, etc.

2.2.1 COMPUTER SECURITY

Computer security is to prevent attackers from achieving their objectives through unauthorized access or unauthorized use of computers and networks.

2.2.1.1 Goals of computer security

The goals of computer security can be categorized as follows (Lapiedra, 2002):

1) **Detection:** to detect activities that violate the security policy, detect intruders that sniff network and detect other attacks such as passive attack or active attack
2) **Prevention:** is ideal, because then there are no successful attacks, to prevent someone from violating security policy.
3) **Recovery**: to stop policy violations to assess and repair damage, ensure availability in presence of an ongoing attack and retaliation against the attacker.

2.2.1.2 Components of computer security

The components of a computer security system include (Leidigh, 2013):

a) Confidentiality: Keeping data and resources secret or hidden.
b) Integrity: Ensuring authorized modifications and Includes correctness and trust
c) Availability: Ensuring authorized access to data and resources when desired.

d) Accountability: Ensuring that an entity's action is traceable uniquely to that entity.

e) Security assurance: Assurance that all four objectives are met.

f) Authentication: Identity authentication (a person; organizational entity; software agent; device).

2.2.2 SECURITY ARCHITECTURE

The security architecture explains the requirements that includes policies, information services, and security mechanism which are used to protect information from attackers and intruders as shown in Figure 2.1(Rigney, 2007).

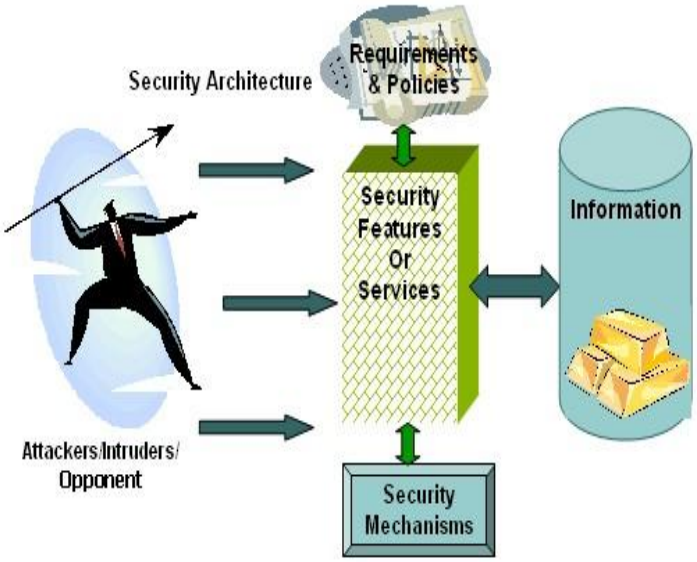


Figure 2.1: An Illustration of Security Architecture (Rigney, 2007)

2.2.3 RADIUS PROTOCOL

The RADIUS protocol was first defined in Request for Comment (RFC) 2058, in January 1997, contains proposed standard of RADIUS protocol. Also in January 1997 RADIUS accounting was introduced in RFC 2059, status of which is informational. Later in April 1997 these RFCs were obsoleted by RFC 2138 and RFC 2139. Then in June 2000 RFC 2865 defined RADIUS draft standard and obsoleted RFC 2138. RADIUS allows several clients to use one centralized authentication and authorization server for user authentication. User passwords transmitted to the

server are encrypted and client can authenticate the server from reply. Replies are also protected from alteration. RADIUS protocol is used for user authentication and authorization and to pass configuration data between two servers. These servers are RADIUS server and Network Access Server (NAS) that acts as client for RADIUS server. NAS sends requests to RADIUS server which replies whether it denies or accepts the request and to pass configuration information concerning the request as shown in Figure 2.2 (M'Raihi, 2011).

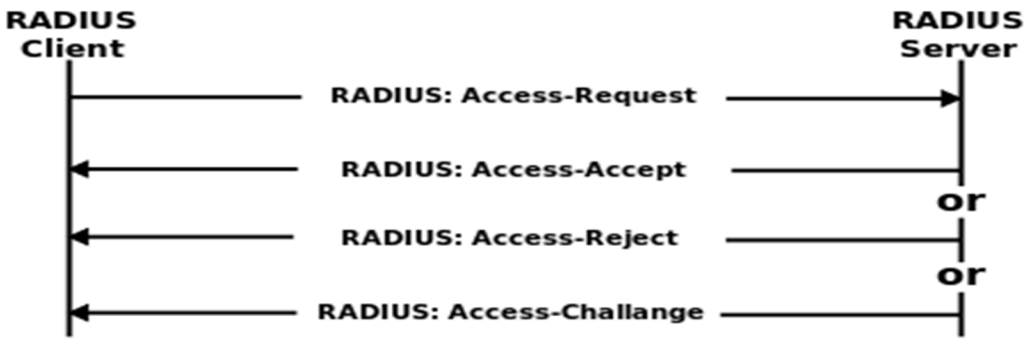


Figure 2.2: RADIUS Authentication and Authorization Flow (M'Raihi, 2011)

2.2.4 AUTHENTICATION, AUTHORIZATION AND ACCOUNTING (AAA)

Authentication, Authorization, and Accounting (AAA) is a framework for intelligently controlling access to computer resources, enforcing policies, auditing usage, and providing the information necessary to bill for services. These combined processes are considered important for effective network management and security (M'Raihi, 2011)

As the first process, authentication provides a way of identifying a user, typically by having the user go through a defined identification. The AAA server compares a user's authentication credentials with other user credentials stored in a database. If the credentials match, the user is granted access to the network. If the credentials are at variance, authentication fails and network access is denied.

Following authentication, a user must gain authorization for doing certain tasks. After logging into a system, for instance, the user may try to issue commands. The authorization process determines

whether the user has the authority to issue such commands. Simply put, authorization is the process of enforcing policies: determining what types or qualities of activities, resources, or services a user is permitted. Usually, authorization occurs within the context of authentication. Once you have authenticated a user, they may be authorized for different type/level of access or activity (Namrata, 2013).

The final plank in the AAA framework is accounting, which measures the resources a user consumes during access. This can include the amount of system time or the amount of data a user has sent and/or received during a session. Accounting is carried out by logging of session statistics and usage information and is used for authorization control, billing, trend analysis, resource utilization, and capacity planning activities (Rigney, 2007).

Authentication, Authorization, and Accounting services are often provided by a dedicated AAA server and this process is described in Figure 2.3

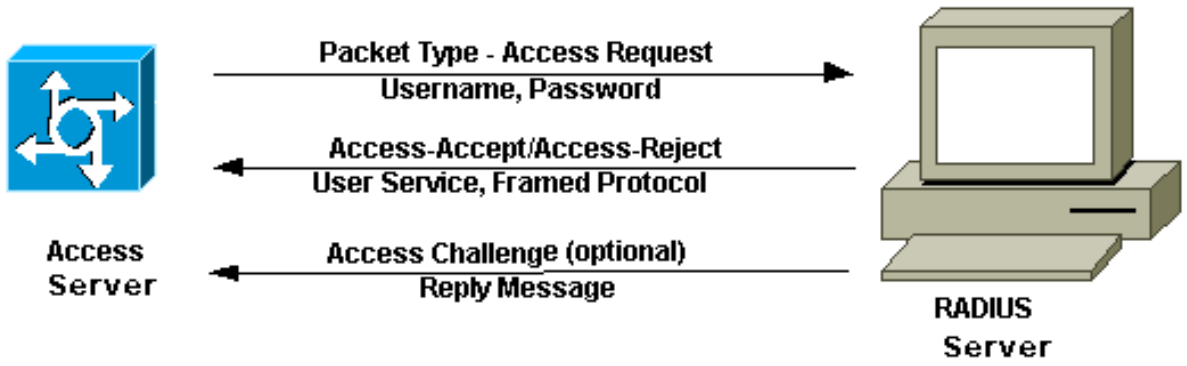


Figure 2.3: Relationship between RADIUS and AAA (Hassel, 2003)

2.2.5 SYSTEM ARCHITECTURE

RADIUS protocol is used between two servers. RADIUS server is a shared authentication server that has a list of valid clients. There is a shared secret between the RADIUS server and these clients. This secret cannot be empty, but otherwise it is not defined by the protocol standard how strong it must be. It is only recommended that it is 16 octets minimum. This secret is used to authenticate the RADIUS server to the NAS and to hide the user password. For these purposes the secret is part of value that is hashed and the hash value is sent (Jaakko, 2014).

RADIUS server also has a database of users containing their passwords, possible other requirements for these users to gain access and configuration data. According to information in this database the RADIUS server accepts or rejects the request or sends a challenge to user. RADIUS server can also act as a proxy relaying requests to other RADIUS server and to NAS. When acting as proxy RADIUS server replies messages between the NAS and other RADIUS server. There can be many RADIUS servers as proxies between the NAS and the RADIUS server that handles the authentication and authorization of the request (Jaakko, 2014).

2.2.5.1 Network access server (NAS)

The Network Access Server (NAS) acts as a client to the RADIUS server. Users call in and NAS prompts for needed authentication information, for example user name and password. The NAS then can use RADIUS server for user authentication. When doing so the NAS sends request to the RADIUS server containing attributes that have information about user that the RADIUS server needs. When sending request containing user password, the password is not sent as clear-text, instead it is encrypted (Rigney, 2007).

The captive portal also resides in the Network access server as shown in Figure 2.4

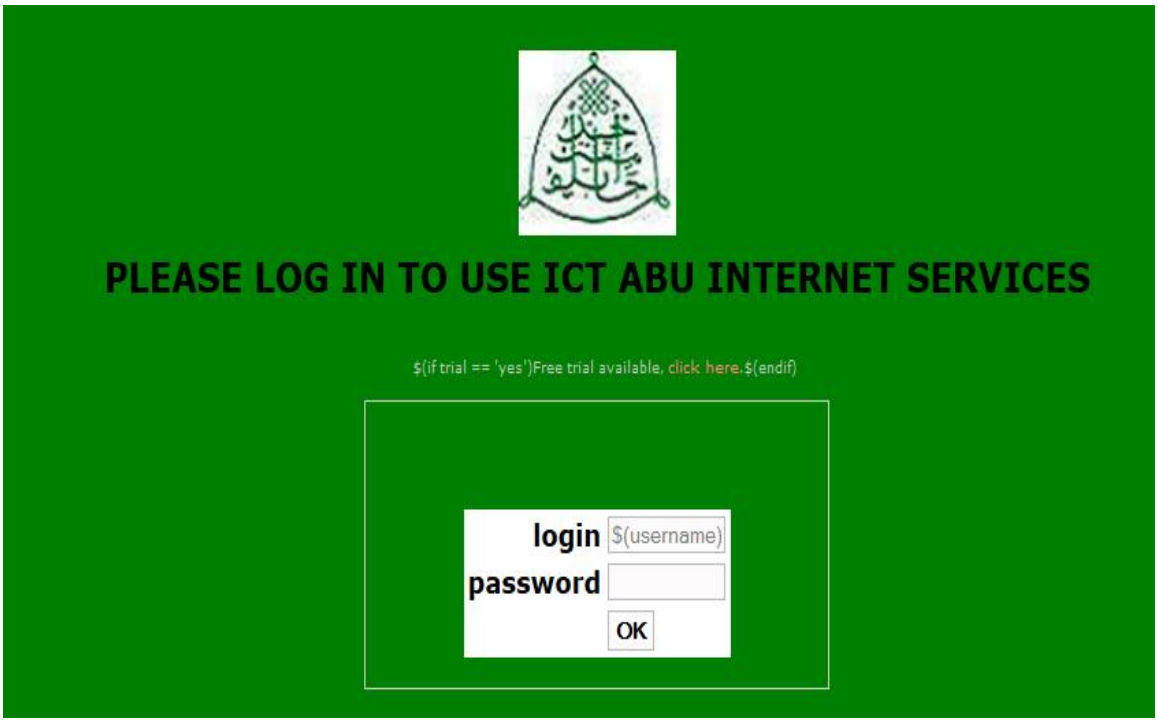


Figure 2.4: Captive Portal Display on ABU Zaria Network

Captive portals have been known to have incomplete firewall rule sets. In some deployments the rule set will route DNS requests from clients to the Internet, or the provided DNS server will fulfill arbitrary DNS requests from the client. This allows a client to bypass the captive portal and access the open Internet by tunneling arbitrary traffic within DNS packets. (Jaakko, 2014)

Some captive portals may be configured to allow appropriately equipped user agents to detect the captive portal and automatically authenticate. User agents and supplemental applications can sometimes transparently bypass the display of captive portal content against the wishes of the service operator as long as they have access to correct credentials, or they may attempt to authenticate with incorrect or obsolete credentials, resulting in unintentional consequences such as accidental account locking (Etutorials, 2015).

A captive portal that uses MAC addresses to track connected devices can sometimes be circumvented by connecting via hard-wire a router that allows setting of the router MAC address. Many router firmware calls this MAC cloning. Once a computer or tablet has been authenticated to the captive portal using a valid username and valid password, the MAC address of that computer

or tablet can be entered into the router which will often continue to be connected through the captive portal as it shows to have the same MAC address as the computer or tablet that was previously connected (Etutorials, 2015).

The prevalent use of captive portals is for user authentication. However captive portals are gaining increasing use on free open wireless and wired networks where instead of authenticating users, they often display a message from the provider along with the terms of use. Although the legal standing is unclear, a click through a page may display terms of use and release the provider from any liability.–Institutions will often require acknowledgement of an Acceptable_use_policy in addition to authentication (Rigney, 2007).

Institutions implementing Network Access Server (NAS) often use captive portals to collect machine information, which the user must supply before gaining admission to the network, to provide online assistance for self-remediation of security problems, and to inform quarantined users when their network access has been revoked. (Jaakko, 2014)

2.2.6 PACKET FORMAT

RADIUS protocol uses UDP to carry its packets. One UDP datagram contains exactly one RADIUS packet. RADIUS packet consists of five different fields: Code, Identifier, Length, Authenticator and Attributes as in Figure 2.5. Length of the entire RADIUS packet is 20 octets for the Code, Identifier, Length and Authenticator. In addition to these various numbers of Attributes can be included, and the total length of the RADIUS packet is in the Length field. (Rigney, 2007)

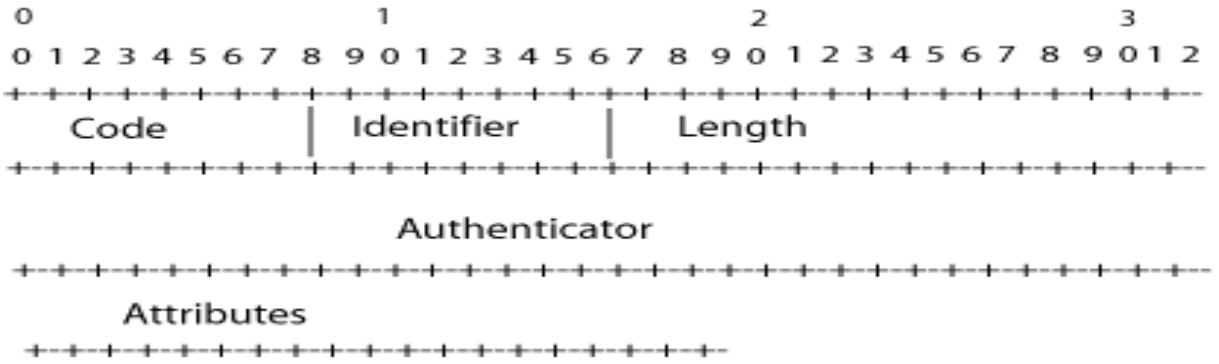


Figure 2.5: RADIUS Packet Format (Rigney, 2007)

First field in RADIUS packet is the Code. It is one octet long. This field determines the type of the RADIUS packet. Originally six Code values were defined (four for authentication and authorization plus two for accounting), with two values reserved for possible use in the future. Also value 255 was reserved. Later further 26 new RADIUS packets Code values were defined by various vendors. All packets with invalid Code are not processed and no error message is sent. (Rigney, 2007)

Second field in RADIUS packet is the Identifier. The Identifier is one octet long. Purpose of this field is to match the requests and replies. The source IP address UDP port of the client is also used for match identification. Each request must have new identifier value, if for the previous request a proper reply was received or if there is any changes in the Attributes of that request. The RADIUS server replies with the same Identifier value in the reply. (Rigney, 2007)

The Length field is third field in the RADIUS packet and it is two octets long. For all RADIUS packets have the Code, Identifier, Length and Authenticator fields, the minimum length of the RADIUS packet 20 octets and therefore the minimum value for Length is 20. Maximum value is 4096. If the RADIUS packet is longer than Length field states, all data that is outside the stated length is ignored. This is done to avoid over flows. If the RADIUS packet is longer than Length field states, the packet is not processed and no error message is sent. (Namrata, 2013)

Fourth field in RADIUS packet is the Authenticator. This field is 16 octets and the most significant octet comes first. This field is used for two security functions. It authenticates the reply from the RADIUS server to the NAS and is also used in encryption of User-Password attribute. Two different kinds of Authenticator fields are defined (Namrata, 2013).

Request Authenticator is the name of the Authenticator field in Access-Request type packets. Request Authenticator is a random number that the NAS generates in order to be able to authenticate that the reply is intended exactly for the request that the Request Authenticator was

generated for. Therefore, it must be unique and unpredictable. NAS also uses Request Authenticator when encrypting User-Password attribute (Trupti, 2012).

Response Authenticator is the name of the Authenticator field in Access-Accept, Access-Reject and Access-Challenge type packets. The value of the Response Authenticator is calculated by the RADIUS server. For this calculation the RADIUS server uses the values of the Code, Identifier and Length fields of the response being made, the Request Authenticator of the request, the Attributes of the response being made and the shared secret (Trupti, 2012).

These are concatenated in this order and then Message Digest Key (MD5) hash is calculated of this concatenated string. The hash value is the response authenticator and it is expressed as follows: (Mikko, 2004)

Response Authenticator = MD5 (Code + Identifier + Length + Request Authenticator + Attributes + Shared Secret) **(2.1)**

In addition to previous four fields RADIUS packet can contain a number of attributes as shown in Figure 2.5. RADIUS uses attributes to carry additional information such as configuration data, information about the user and service. Standard length for RADIUS attribute is three fields as shown in Figure 2.6, but some attributes have more fields. These fields are Type, Length and Value. **(**Rigney, 2003**)**

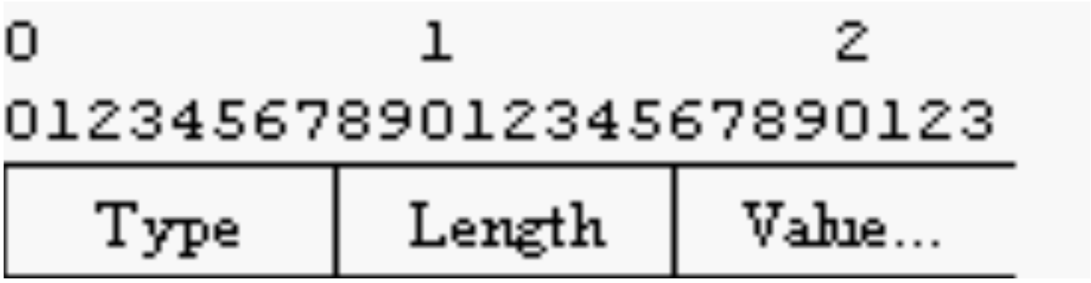


Figure 2.6: RADIUS Attributes (Rigney, 2007)

Table 2.1: Typical Standard RADIUS Attributes (Hassel, 2003)

6	Accounting Status	**30**	New Pin
7	Password Request	**31**	Terminate Session
8	Password Ack	**32**	Password Expired
9	Password Reject	**33**	Event Request
10	Accounting Message	**34**	Event Response
21	Resource Free Request	**40**	Disconnect Request
22	Resource Free Response	**41**	Disconnect Ack
23	Resource Query Request	**42**	Disconnect Nak
24	Resource Query Response	**43**	Change Filters Request
25	Alternate Resource Reclaim Request	**44**	Change Filters Ack
26	NAS Reboot Request	**45**	Change Filters Nak
27	NAS Reboot Response	**50**	IP Address Allocate
29	Next Passcode		

2.2.7 SHARED SECRETS

To strengthen security and increase transactional integrity, the RADIUS protocol uses the concept of shared secrets. Shared secrets are values generated at random that are known to both the client and the server. The shared secret is used within all operations that require hiding data and concealing values. The only technical limitation is that shared secrets must be greater than 0 in length, but the RFC recommends that the secret be at least 16 octets (Hassel, 2003).

A secret of that length is virtually impossible to crack with brute force. The same set of best practices that dictate password usage also govern the proper use of RADIUS shared secrets. Shared secrets are unique to a particular RADIUS client and server pair. For instance, if an end user subscribes to multiple Internet service providers for his dial-up access, he indirectly makes requests to multiple RADIUS servers. The shared secrets between the client NAS equipment in ISPs A, B, and C that are used to communicate with the respective RADIUS servers should not match. While some larger scale RADIUS implementations may believe that protecting transactional security by using an automated shared-secret changer is a prudent move, there is no guarantee the clients and servers can synchronize to the new shared secret at the most appropriate time. And even if it was certain that the simultaneous synchronization could occur, if there are

outstanding requests to the RADIUS server and the client is busy processing, then those outstanding requests will be rejected by the server. The code below shows how the secret code is implemented in the RADIUS environment (Hassel, 2003)

```
client NAME {
  ipaddr = IPADDRESS
  secret = SECRET
}
```

2.2.8 THE VULNERABILITY OF RADIUS PROTOCOL

RADIUS consistently provides some levels of protection against sniffing and active attacks. Unfortunately, there are several vulnerabilities in RADIUS protocol that are either caused by the protocol or caused by poor client implementation such as:

1) **Offline dictionary attack** on RADIUS shared secret via message-authenticator attribute where attacker can attempt offline attack on any packet with a message-authenticator attribute. (Rigney, 2007)
2) **Online attack against the PAP password in** this attack, RADIUS servers enabling replay of request authenticator (16 octets) and identifier using PAP. Attacker can then try an online dictionary attack against the user password of 16 characters or less. (Rigney, 2007)
3) **Response authenticator based shared secret attack** attacker observes a valid Access-Request packet and the associated Access-Accept or Access-Reject packet. They can launch an off-line exhaustive attack on the shared secret. The attacker can pre-compute the MD5 state and then resume the hash once for each shared secret guess. (Etutorials, 2015)
4) **User-password attribute based shared secret attack**, an attacker can gain information about the shared secret and attempts to authenticate to the client with a known password.
5) **User-password based password attack**, the attacker attempts firstly to authenticate to the client using a valid username, then captures the resulting Access-Request packet and

determines the result of the MD5 then it's replay modified Access-Request packets, using the same request authenticator and MD5. (Etutorials, 2015)

6) **DOS arising from the prediction of the request authenticator**, the attacker can predict future values of the request authenticator then create a dictionary of future request authenticator values. The attacker can then masquerade as the server and responds to the client's requests with valid looking Access-Reject packets then creating a denial of service.
7) **Replay attack:** Attacker can get user password through passive eavesdropping or sniff traffic; an attacker can build a dictionary attack to find patterns and break a cipher, then replay to server with valid login. The adversary records a data transfer and replies it at any time through the network. Replay attack is a method of exploiting a captured packet or packets and resend to user that cause unexpected or unwanted behavior from the server. If the server does not detect the reused data and accepts the repeatedly transmitted packets, the attack is successful. If an attacker would come across the data from a user that is generated by the JavaScript, it would be possible to login as the user without the server noticing any difference. The data is usually gathered either by listening to the traffic or by installing malicious software on the user computer. (Florian, 2010).

   To resolve these security issues, the OTP technique was developed.

2.2.8.1 Attacker database

Database for attacker includes two tables; the first table called users table includes username of all users with their passwords that have been captured through sniffing on ports or implement malicious script. The second table is OTP table that includes Id PIN and OTP stolen from users of the system. A typical attacker database is shown in Figure 2.7 (Gagan, 2013)

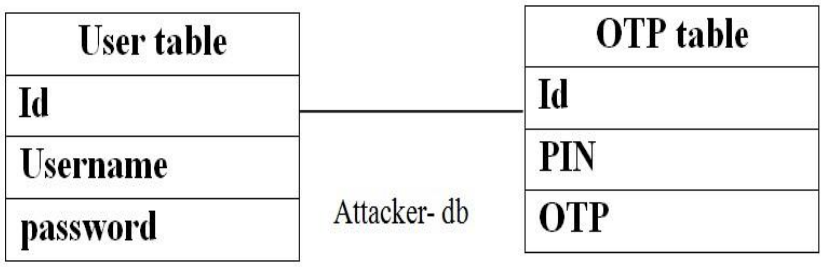


Figure 2.7: Attacker Database (Gagan, 2013)

2.2.8.2 Replay attack components

Replay attack components are described as follows: (Tse, 2014)

1. **User:** User is a person (victim) who requests access to the system services.
2. **Captive Portal:** The System is a web page that is designed to provide services to end users.
3. **RADIUS Server:** The RADIUS server checks the user information (username and password) which is entered in the sign-in form. The RADIUS server has a database to store the user data.
4. **Authentication server (AS):** AS is responsible of the second phase in the system, after the system user enters the PIN and the generated OTP in the OTP page, the authentication server will check the secret key, PIN and last OTP and then will send a response to the system.
5. **Attacker server**: Attacker server sends malicious software script to user to listen on any request processed to system.
6. **Attacker**: Is malicious user that can sniff to network traffic and steal user data. Attack depends on the incoming requests to the server continuously without interruption.
7. **Response time:** Server Response Time is defined as the elapsed time between the end request or demand on a computer system by user and the beginning of server response. The server response time can also be defined as the length of time taken by a system to respond to an instruction by server. The response time of a RADIUS server must not be more than 1000ms (Rigney, 2007).

2.2.8.3 Replay attack architecture

The attacker server is able to capture the username and password from user within a malicious script which is recording the private information for users, then attacker server sends the username and password to attacker who is stealing and trying to authenticate username and password in the system. The attacker listens and take any information from user in each traffic over the network. Replay attack architecture is shown in Figure 2.8 (Amna, 2014).

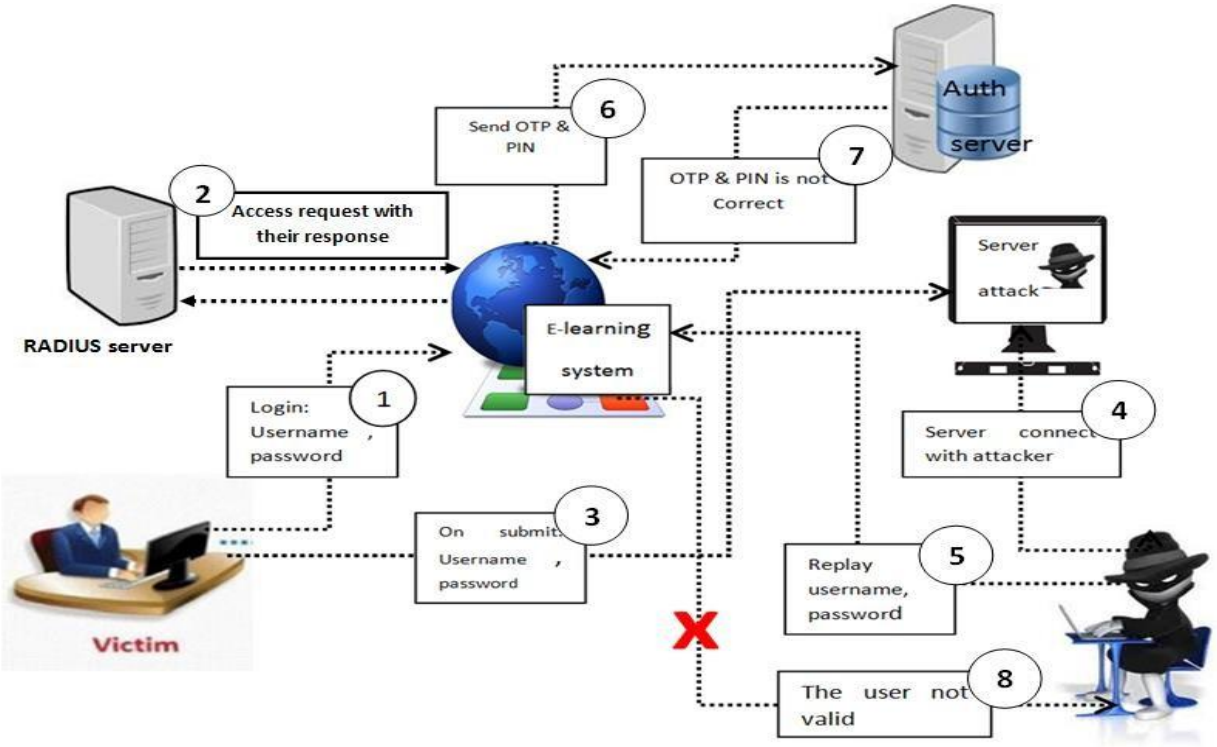


Figure 2.8 : Replay Attack Architecture (Amna, 2014)

Table 2.1 Replay Attack Steps (Jaakko, 2014)

Step No.	Description
1.	The user enters the username and password in the login screen.
2.	The system sends user data to the RADIUS server to authenticate the user, the RADIUS server verifies the username and password and sends a response either accept or reject.
3.	Attack depends on the reception of incoming requests to the server continuously without interruption
4.	Attacker server sends malicious software script to user to listen on any request processed to system.
5.	Attacker steals data and last request, then sends these data to RADIUS Server
6.	RADIUS server receives stolen data from attacker and replays for AS.
7.	High response time on the RADIUS server results in time out but with valid credentials, which the attacker can use.
8.	AS query and verifies PIN, OTP, secret key, last OTP and response with the user is not authenticated or authenticated.

2.2.9 ONE TIME PASSWORD (OTP) TECHNIQUE

One Time Password OTP is an instant password, in other words it is a code that changed after each use and uses it to authenticate (Sung-Jae, 2011). OTP are passwords that are only valid for a single or small number of transactions. An attacker has a smaller period of time to gain access to resources protected by such password because any previously stolen passwords will likely have become invalid. That means adding some uncertain factors in the procedure of authorization. The information transmitted over network is different, thus the security is improved (Trupti, 2012). OTP has a characteristic making it impossible to predict the next password from the current password; also they are not vulnerable to replay attacks (Gagan, 2013). OTPs avoid a number of shortcomings that are associated with traditional (static) passwords (Sung-Jae, 2011). OTP is based on a cryptographic algorithm (Himika, 2012).

$$\boldsymbol{Cryptogram = f(k)} \qquad \textbf{(2.2)}$$

Where key k is a cryptographically generated

Computing the cryptogram with factors makes the output random and on time, cryptographic algorithms based counter also called even and based time (e.g. seconds) with triple factor, f1:

key k a cryptographic is generated, f2:T refers to time factor, f3: c refers to counter factor. While f1.i is number of factor, equation (2.1) shows a cryptographic algorithm of OTP. Figure 2.7 explains OTP generation process.

$$\boldsymbol{Cryptogram = f\ (k, C, T)} \qquad \textbf{(2.3)}$$

2.2.9.1 Generation OTP and distribution

The process of the OTP generation consists of: (Bellare, 2011)

1) Input value
2) OTP generation
3) OTP extraction
4) Time

The OTP generation algorithm generates an OTP value from an input value (user's strong password and secret key) as shown in Figure 2.9. It is based on hash functions for message digest key (MD5) and uses the shared input value between the server and the OTP generator (Himika, 2012).

A time value, a counter value and a challenge value are used as the key and data of the generation algorithm. The extraction algorithm of the OTP value extracts the real OTP value from the output value of the OTP generation algorithm. (Bellare, 2011)

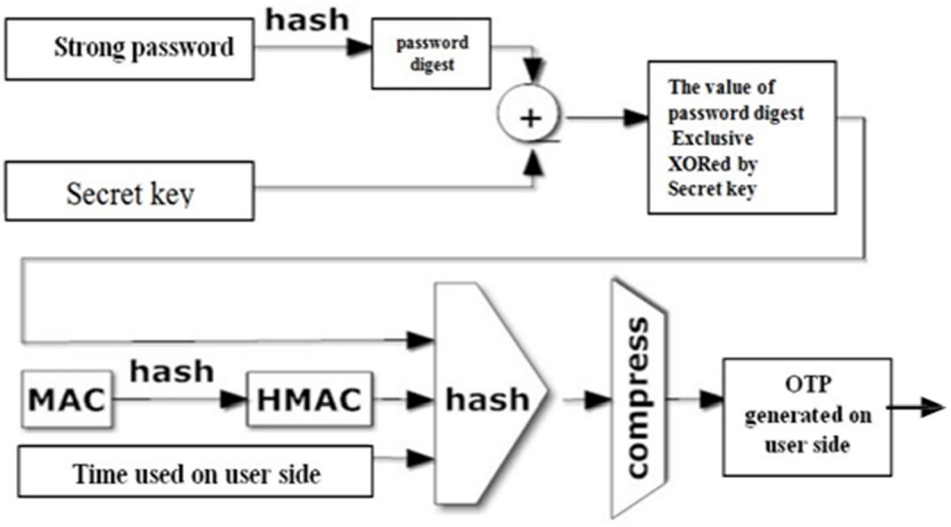


Figure 2.9: OTP Generation (Xuguang, 2012)

In this research three main techniques of OTP **(TOTP, HOTP and CROTP)**, are implemented and simulated on the modeled ABU network and their response time compared in order to determine the most appropriate approach to adopt and improve on the most appropriate technique.

2.2.9.2 Justification for OTP

One-time passwords are passwords that are only valid for a single or small number of transactions. This contrasts with conventional passwords which are valid for many transactions as users are reluctant to voluntarily change passwords frequently. Since OTPs are only valid for a limited number of uses, an attacker has a smaller window of time to gain access to resources guarded by such a password because any previously stolen passwords will likely have become invalid. As with traditional passwords, one-time passwords are vulnerable to man-in-the-middle attacks. By observing the OTP before it is successfully received by the authenticator, an attacker has a valid password. Because of this undesirable property, both OTPs and conventional passwords must travel securely.

2.2.9.3Approaches for the generation of OTP

There are three major technique used for generation of OTP (Namrata, 2013)

1. TIME SYNCHRONIZATION

In this technique, both the client and server will have synchronous time clocks. In this approach time is used as a changing factor which changes every 3 minutes. The generation time must be synchronized with the authentication server time. If the authentication server and the user do not keep the same time, then the expected OTP value won't be produced and the user authentication will fail (Namrata, 2013)

With time-synchronized OTPs, the user typically must enter the password within a certain period of time before it is considered expired and another one must be generated (Binod, 2013)

2. EVENT SYNCHRONIZATION

In this technique, both the client and server will typically have a counter value. Whenever client wants to login, it generates OTP from the counter value and any other input Personal Identifier Number (PIN) and updates the counter. User submits the generated OTP to server. Server also generates the password using the counter. If password match, the server authenticates the user and updates the counter (increment/ decrement the counter), it may happen that the counter on client and server may drift (due to passwords generated by client but not submitted, passwords submitted by client but does not reach to server due to network failure, etc.). (Namrata, 2013)

3. CHALLENGE – RESPONSE TECHNIQUE

In this technique a random number (PIN) chosen by the authentication server is sent to a user, the user enters PIN value then sends response to the server. This technique based on a challenge response. (Bellare, 2011)

2.2.9.4 Types of OTP techniques

The following are the different types of OTP techniques:

1. HASH ONE TIME PASSWORD (HOTP) TECHNIQUE

The HOTP technique is based on an increasing counter value. Both the client and server will typically have a counter value. Server generates the password to use the counter. If passwords match, the server authenticates the user and updates the counter increment/ decrement the counter), it may happen that the counter at client and server may drift (due to passwords generated by client but not submitted, or passwords submitted by client but does not reach to server due to network failure, etc.) (Sung-Jae, 2011).

In this case it will respond to server with denial service. The steps in generation of HOTP technique is shown in Table 2.2 (Bellare, 2011)

Table 2.2: Steps in Generation of HOTP Technique (Bellare, 2011)

Steps	Description
1.	The user enters the username and password in the login screen. The password should contain more than six characters.
2.	The system sends user data to the RADIUS server to authenticate the user.
3.	The RADIUS server verifies the username and password and sends a response either accept or reject.
4.	The user opens the OTP application and enters the PIN.
5.	The OTP application provides the generated OTP for the system user.
6.	The user enters the PIN and the generated OTP through the system.
7.	The system sends a request to the AS to check the last OTP, the PIN and the secret key of the system user.
8.	The AS verifies user OTP and sends a response to system either accept or reject.

HOTP: HMAC-Based One-Time Password Algorithm

On both the client and the server, the choice of algorithm for passcode generation is essentially arbitrary, so long as it provides adequate security and can be used in a user friendly manner. The HOTP is a counter-based algorithm called HMAC-Based One Time Password (HOTP) Algorithm that is relatively easy to implement and met the necessary usability requirements. (Binod, 2013). The HOTP algorithm is based on a monotonically increasing counter value and a static symmetric key known only to the client and the server. In order to create the HOTP value, the HMAC-SHA-1 algorithm is used. Each client has a unique shared secret, typically 128 bits or 160 bits in length. The shared secret is combined with an increasing counter, also shared between the client and the server, to generate the current passcode. (Binod, 2013)

The obtained HOTP is as follows: (Himika, 2012)

$$\boldsymbol{HOTP(K,C) = Truncate(HMAC - SHA - 1(K,C))} \qquad \textbf{(2.4)}$$

Where: Truncate represents the function that converts an HMAC-SHA-1 value into an HOTP value; and the key (K), the counter (C), and Data values are hashed high- order byte first. (Himika, 2012)

The actual HOTP algorithm is relatively simple to understand. First, a SHA-1 HMAC generator is initialized using the shared secret. Then the HMAC of the current counter, or moving factor, is computed. Next, through a process called dynamic truncation, certain bytes are extracted from the HMAC. Finally, these bytes are taken modulo 10n, where n is the number of digits desired in the passcode, to produce the current pass code. (Yeh, 2009)

In order for a client to authenticate to a server, both must generate the same passcode. Specifically, assuming that the server has already distributed the shared secret to the client, the client counter and the server counter must be synchronized. When the counters are not synchronized, a process called resynchronization must occur. The HOTP algorithm has two basic mechanisms to resynchronize the server with the client. The most straightforward method is for the client to simply send the counter value over to the server. The server would merely need to verify that the new counter is greater than the current counter. (Binod, 2013)

The second method is for the server to maintain a look-ahead window of future passcodes. If the client provides a passcode that lies within this window, the server will ask the user to generate the next passcode and send it to the server. If two consecutive passcodes match, then the server will resynchronize. The flowchart for HOTP is shown in Figure 2.10 (Gagan, 2013)

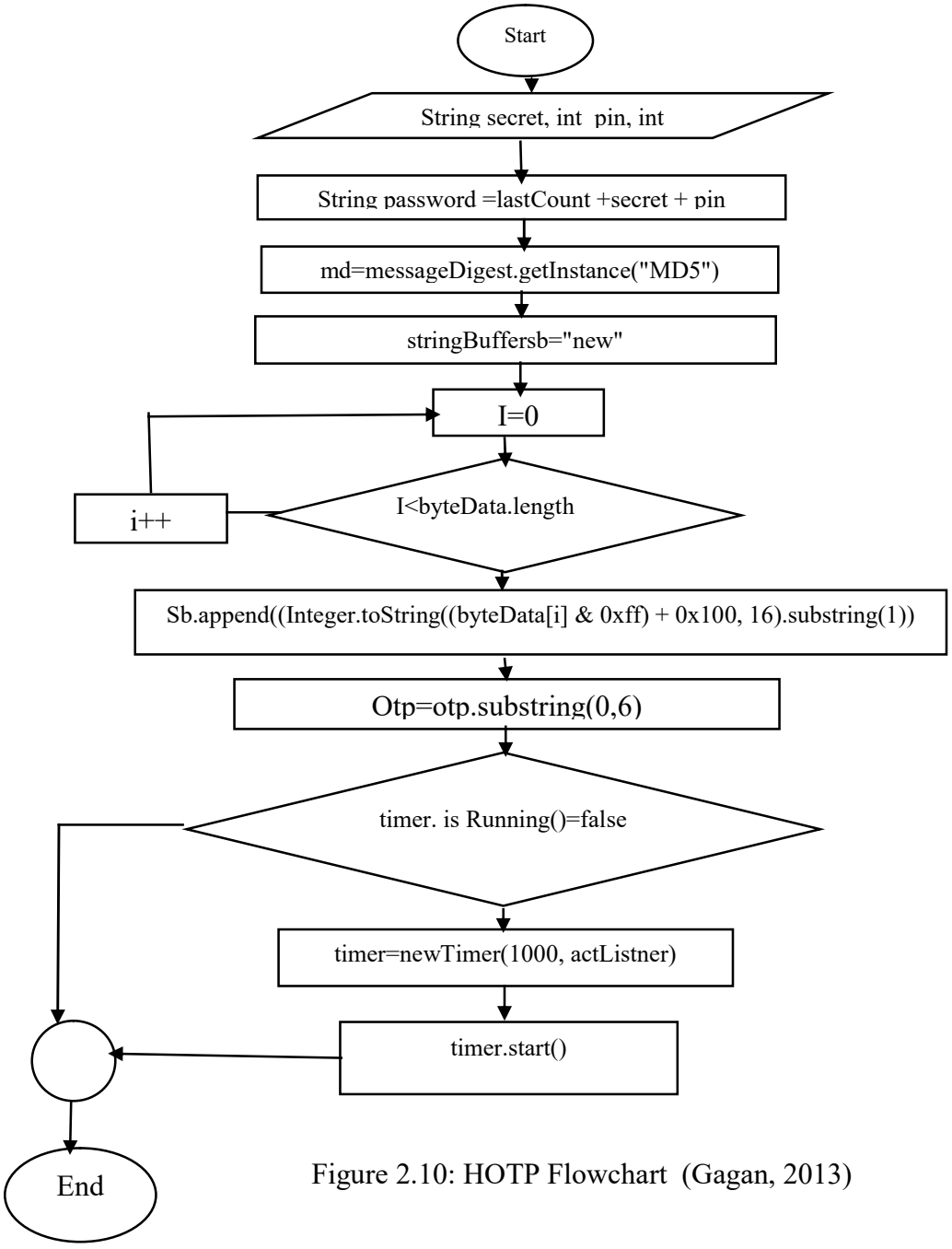


Figure 2.10: HOTP Flowchart (Gagan, 2013)

2.2.9.5 Time one time password (TOTP) technique

The TOTP technique (M'Raihi, 2011) that is a time based OTP. Time dynamism is an OTP generation principle widely utilized in two factor authentication schemes; two factor authentication based on PIN identification as first factor and OTP generation (different password) as second factor to increase security. Different password is needed for different time to prevent number of

attacks, this schema depends on two periods, static period is designed for user to input the OTP into the login form upon receipt of the dynamic period password, and the length of which can always be customized by the client (Xuguang, 2012). The steps in generation of TOTP is shown in Table 2.3 (Amna, 2014)

Table 2.3: Steps in Generation of TOTP (Amna, 2014)

Steps	Description
1.	The user enters the username and password in the login screen. The password should contain more than six characters.
2.	The system sends user data to the RADIUS server to authenticate the user.
3.	The RADIUS server verifies the username and password and sends a response either accept or reject.
4.	The user opens the OTP application and enters the PIN.
5.	The OTP application provides the generated OTP for the system user. This OTP is expired after 3 minutes.
6.	The user enters the PIN and the generated OTP through the system.
7.	The system sends a request to the AS to check the last OTP, the PIN and the secret key of the system user.
8.	The AS verifies user OTP and sends a response to system either accept or reject.

TOTP Algorithm

A time-based variant of the OTP algorithm provides short-lived OTP values, which are desirable for enhanced security.

The proposed algorithm can be used across a wide range of network applications, from remote Virtual Private Network (VPN) access and Wi-Fi network logon to transaction-oriented Web applications. The TOTP is defined as: (Florian, 2010)

$$\boldsymbol{TOTP = HOTP(K,T)} \qquad \textbf{(2.5)}$$

Where T is an integer represent time, k present secret key.

The current time represents initial time for the user to log in . (Florian, 2010)

T = (Current Unix time - T0) ÷ X.

X represents the time step in seconds (default value X = 180 seconds) and is a system parameter.

T0 is the Unix time to start counting time steps (default value is 0, i.e. the Unix epoch) and is also a system parameter.

The flowchart for the TOTP algorithm is shown in Figure 2.11 (Yang, 2012) also implemented in Appendix I using PhP script.

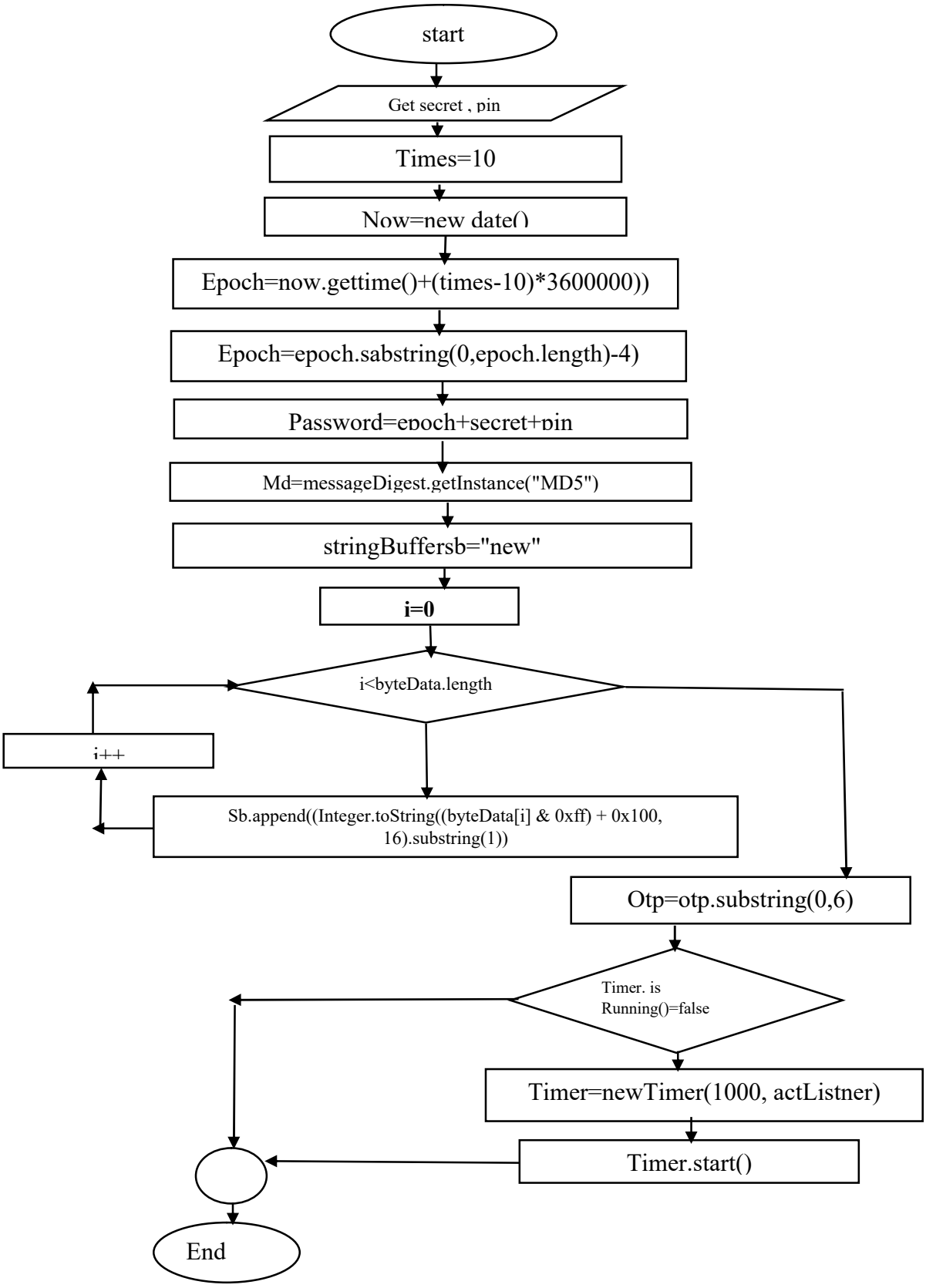


Figure 2.11: Flow chart for TOTP (Yang, 2012)

2.2.9.5 Cryptography one time password (CROTP) technique

CROTP (Bellare, 2011) is a generalization of HOTP with variable data inputs that based on an incremented counter and secret key values (Sung-Jae, 2011). It is used in several network protocols such as RADIUS, Kerberos and digest access authentication. The definition of CROTP requires a cryptographic function, a key K and a set of data input parameters (Alan, 2013). Instead of a shared counter value, a challenge is issued from the validation server to the user. The user would then input this challenge to receive the OTP value: (Alan, 2013)

$$\boldsymbol{CROTP = CryptoFunction(K, DataInput)} \quad \textbf{(2.6)}$$

Where K represents a shared secret key,

Data input function represents input data values; crypto function represents functions that perform the CROTP. In details Inputs that are based on challenge response (CR) authentication mechanism consists of the server which presents the user with an unpredictable challenge every time the user attempts to authenticate. For every challenge there is an associated response that allows the user to be authenticated. (Bellare, 2011)

As a consequence of the unpredictability of the challenge, this family of mechanisms ensures that the entity being authenticated is active when authentication takes place. Passive attacks are not possible, and even though active attacks remain feasible, they need to be carried out in a real time, which means that the time interval between the moment when the attacker captures the response and when it is used needs to be very short for the user in order to accept it. Table 2.4 shows the steps for generation of CROTP technique. Figure 2.12 shows the flow chart for CROTP and also implemented in Appendix I using PhP programming language.

Table 2.4: Steps for Generation of CROTP Technique (Kun Tang, 2014)

Steps	Description
1.	The user enters the username and password in the login screen. The password should contain more than six characters.
2.	The system sends user data to the RADIUS server to authenticate the user.
3.	The RADIUS server verifies the username and password and sends a response either accept or reject.
4.	The system provides a random PIN to challenge the user
5.	The user opens the OTP application and enters the system random PIN.
6.	The OTP application provides the generated OTP for the system user.
7.	The user enters the generated OTP through the system.
8.	The system sends a request to the AS to check the last OTP, the random PIN and the secret key of the system user.
9.	The AS verifies the user OTP and sends a response to system either accept or reject.

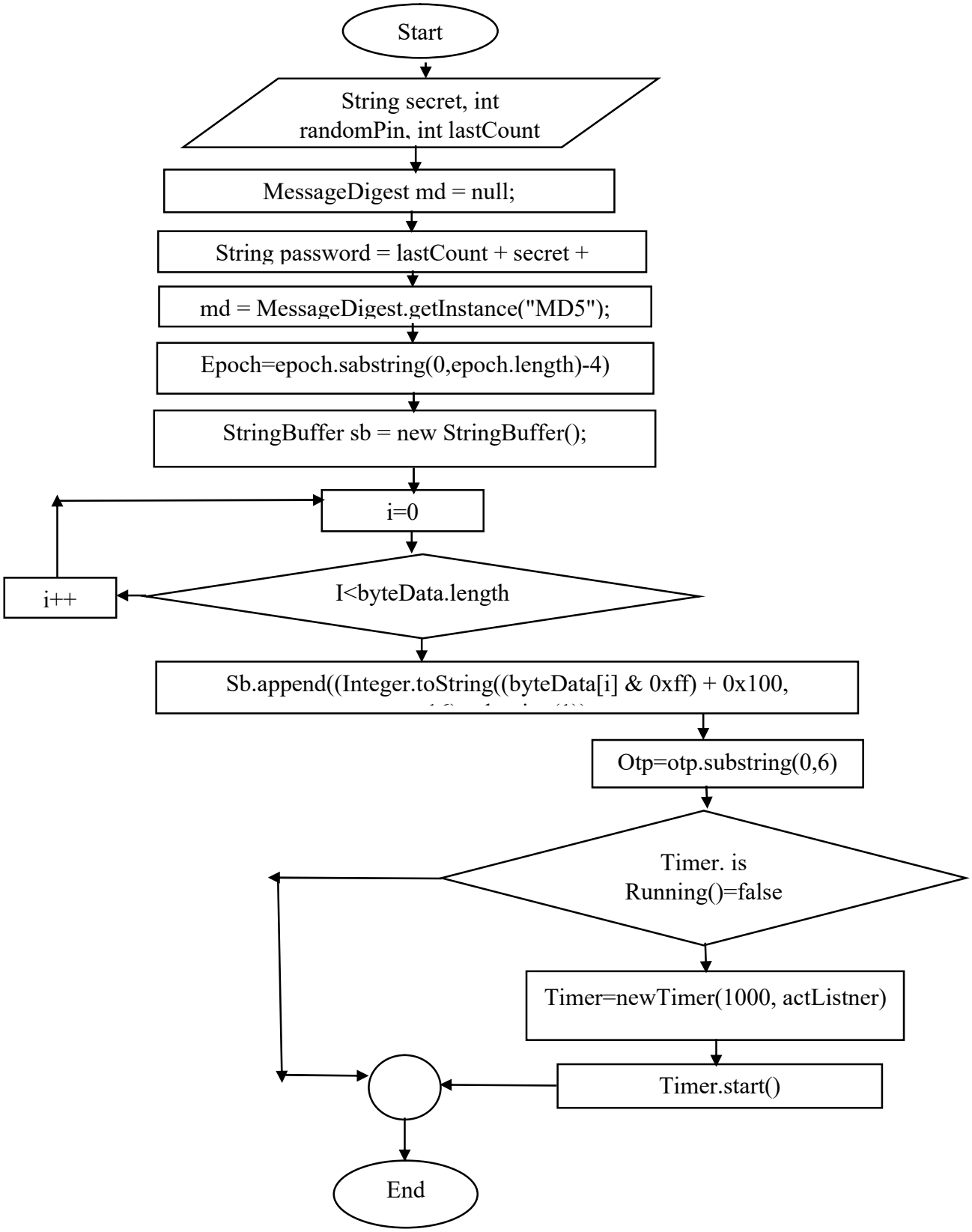


Figure 2.12: Flowchart for CROTP (Kun Tang, 2014)

2.2.10 SIMULATION SOFTWARE

Simulation is the science of creating statistically accurate models to represent the behavior of real life systems in order to subject them to predictive experimentation. These experiments or scenarios can then enable questions to be answered without risk or disturbance to the real life system (Lanners, 2015). Discrete Event Simulation works by modelling the individual events that occur at given time intervals, taking into account resources, constraints and the way they interact. It also includes the randomness and variability you will see in real life. In this way, it behaves like a real process. Some types of simulation software are:

1. OMNet++
2. OPNET Modeler
3. NS2
4. GNS3

2.2.10.1 OMNET++

It is a component-based, modular and open architecture discrete event simulator framework. The most common use of OMNeT++ is for simulation of computer networks, but it is also used for queuing network simulations amongst others . It has a C++ class library which consists of the simulation kernel and utility classes (for random number generation, statistics collection, topology discovery etc.). It, however, lacks the capabilities to connect these modules to virtualization software and also has poor documentation for these modules. (Saba, 2012)

2.2.10.2 Optimized network engineering tools (OPNET)

It is an extensive and powerful simulation software with wide variety of possibilities to simulate entire heterogeneous networks with various protocols. The advantages of OPNET compared to other simulators include GUI interface, comprehensive library of network protocols and models, graphical interface to view the results, availability of documentation for the user to develop the

network models etc. It provides a comprehensive development environment supporting the modeling of communication networks and distributed systems. Both behavior and performance of modeled systems can be analyzed by performing discrete event simulations. The disadvantages of using OPNET are that the simulation requires a lot of processing power and can be very time consuming particularly for networks with a large number of routers and switches (Saba, 2012).

2.2.10.3 Network simulator version 2 (NS2)

NS-2 is the simulator targeted at network research. The user interfaces with the simulator by using object-oriented script language OTcl on the UNIX systems (although it is possible to install NS-2 in Windows OS). In order to build the simulation topologies on NS-2, one has to know OTcl language. Moreover, Mikrotik router operating system modules are not available in its libraries (these modules have to be implemented using third party applications). Documentation is not available for all modules. (Scroggins, 2013)

2.2.10.4 Graphical network simulator-3 (GNS3)

The Graphical Network Simulator-3 (GNS3) is a free open source software. It works by using real routers, switches with supported IOS images which are emulated using a program called Dynamips.GNS3 is really like the GUI part of the overall software. With this GUI, users get an easy to use interface that allows them to build complex labs consisting of a variety of supported Cisco, juniper, Mikrotik, HP etc. routers and switches. The program that does the real job of emulating the routers using real IOS images is Dynamips, which is referred to as the back-end of the whole operation whereas Dynagen is referred to as the front-end. This is mainly because Dynagen communicates with Dynamips using a hypervisor (Harry, 2010). All this put together helps to make the configuration process simpler. GNS3 allows the combination of some virtualization software like the QEMU and Virtualbox, which allows it to use virtual servers. This research work will focus on GNS3 because of its advantages and flexibility. (Bray, 2009).

2.2.11 VIRTUALIZATION OVERVIEW

In computing, virtualization means to create a virtual version of a device or resource, such as a server, storage device, network or even an operating system where the framework divides the resource into one or more execution environments, even something as simple as partitioning a hard drive is considered virtualization because you take one drive and partition it to create two separate hard drives (Swathi, 2014).

Devices, applications and users are able to interact with the virtual resource as if it were a real single logical resource. The term virtualization has become somewhat of a buzzword, and as a result the term is now associated with a number of computing technologies including the following (Swathi, 2014):

1) Storage virtualization: the amalgamation of multiple network storage devices into what appears to be a single storage unit.
2) Server virtualization: the partitioning a physical server into smaller virtual servers.
3) Operating system-level virtualization: a type of server virtualization technology which works at the operating system (kernel) layer.
4) Network virtualization: using network resources through a logical segmentation of a single physical network.
5) Application virtualization

In this research work the Server virtualization is used.

2.2.11.1 Server virtualization

Server virtualization is the masking of server resources, including the number and identity of individual physical servers, processors, and operating systems, from server users. The server administrator uses a software application to divide one physical server into multiple isolated

virtual environments. The virtual environments are sometimes called virtual private servers, but they are also known as guests, instances, containers or emulations. There are different types of server virtualization software these include (Rao, 2014):

1) Vmware workstation
2) Parallels
3) Virtualbox

1. VMWARE WORKSTATION

It is a hypervisor that runs on x64 computers (an x86 version of earlier releases was available); it enables users to set up virtual machines (VMs) on a single physical machine, and use them simultaneously along with the actual machine. Each virtual machine can execute its own operating system, including versions of Microsoft Windows, Linux, BSD, and MS-DOS. An operating systems license is needed to use proprietary ones such as Windows. Ready-made Linux VMs set up for different purposes are available. This software is not free and source code is not available. (Scroggins, 2013)

2. PARALLELS WORKSTATION

It is a hardware emulation virtualization software, in which a virtual machine engine enables each virtual machine to work with its own processor, RAM, floppy drive, CD drive, I/O devices, and hard disk. Parallels Workstation virtualizes all devices within the virtual environment, including the video adapter, network adapter, and hard disk adapters. It also provides pass-through drivers for parallel port and USB devices. Because all guest virtual machines use the same hardware drivers irrespective of the actual hardware on the host computer, virtual machine instances are highly portable between computers. For example, a running virtual machine can be stopped, copied to another physical computer, and restarted. Parallels Workstation is able to

virtualize a full set of standard PC hardware. The Parallels Workstation has been discontinued for Windows and Linux as of 2013. (Scroggins, 2013)

3. VIRTUALBOX VIRTUALIZATION

It is an open-source software, a cross-platform virtualization application, it installs on your existing Intel or AMD-based computers, whether they are running Windows, Mac, Linux or Solaris operating systems. Secondly, it extends the capabilities of your existing computer so that it can run multiple operating systems (inside multiple virtual machines) at the same time. So, for example, you can run Windows and Linux on your Mac, run Windows Server 2008 on your Linux server, run Linux on your Windows PC, and so on, all alongside your existing applications. Multiple virtual machines can be installed and run simultaneously; the only practical limits are disk space and memory. VirtualBox is deceptively simple yet also very powerful. It can run everywhere from small embedded systems or desktop class machines all the way up to datacenter deployments and even Cloud environments (Rao, 2014).

The following screenshot shows you how VirtualBox, installed on a windows computer, is running different operating systems in a virtual machine window as shown in Figure 2.13, In this research work Virtualbox is used as the virtualization software because of its advantages over other virtualization software.

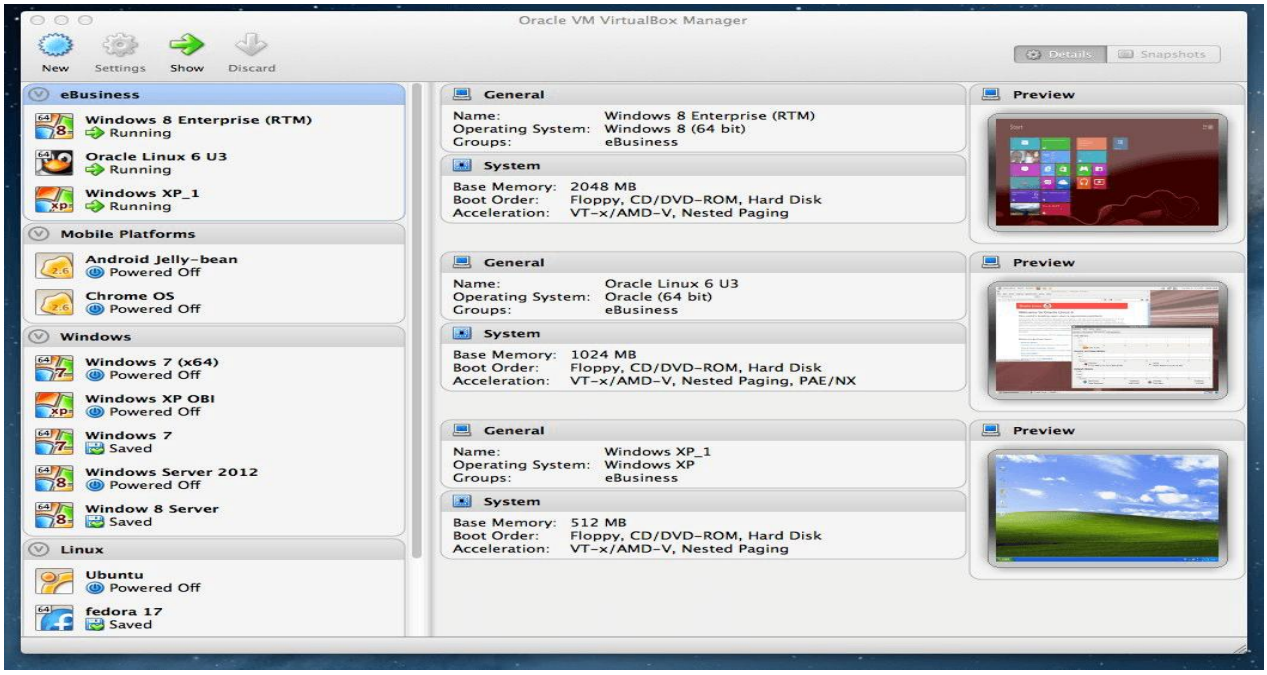


Figure 2.13: Virtual box Environment

2.2.12 PHP PROGRAMMING LANGUAGE

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. Originally created by Rasmus Lerdorf in 1994, the PHP reference implementation is now produced by The PHP Group While PHP originally stood for Personal Home Page it now stands for the recursive backronym PHP, Hypertext Preprocessor (Lerdorf, 20013)

PHP code may be embedded into HTML code, or it can be used in combination with various Web template systems and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications.

There are three main areas where PHP scripts are used: (Xue Liu, 2003)

1) Server-side scripting. This is the most traditional and main target field for PHP. Three things are needed to make this work. The PHP parser (CGI or server module), a web server and a web browser. There is a need to run the web server, with a connected PHP installation. The PHP program output can also be accessed with a web browser,
2) Command line scripting. This can make a PHP script to run it without any server or browser. Only the PHP parser is used in this way. This type of usage is ideal for scripts regularly executed using Linux. These scripts can also be used for simple text processing tasks.
3) Writing desktop applications. PHP is probably not the very best language to create a desktop application with a graphical user interface, also have the ability to write cross-platform applications. PHP-GTK is an extension to PHP, not available in the main distribution. (Bray, 2009).

2.3 REVIEW OF SIMILAR WORKS

In the course of the research, several literatures were consulted and a review of the major ones is presented as follows:

Mikko *et al* (2004) presented the implementation of legacy user authentication into IPSec remote access scenario using the proposed Pre-IKE Credentials Provisioning Protocol (PIC). The research provided a comparison between their technique and other alternative techniques like OTP. The results of this comparison showed that the technique had good interoperability, usability and efficiency with IPSec and OTP between routers only. This technique is, however, only useful between routers and does not protect the users behind the routers.

Kenneth *et al* (2010) presented a model for the secure use of OTP in Password-Authenticated Key Exchange (PAKE) protocols; considering the idea that such protocols should be secure even if previous or future OTP have been compromised. They have provided a generic technique for constructing secure OTP. This construction can be used with pseudo randomly generated OTP, providing greater efficiency in OTP distribution to reduce the damage of many

attacks such as replay attack and spyware. This protocol provided security to OTP to prevent replay attacks on servers only but did not provide any security at the user level.

Havard *et al* (2010) developed protocols that enabled individuals to use their mobile phones as OTP generators using a web-based service. Their phones run a Java MIDlet which communicates with a server to generate OTPs. This is an implementation of the OTP generator with web-based service and does not improve on any of the OTP techniques.

Rohan *et al* (2012) developed a proposal to improve the communication efficiency between NAS and RADIUS server by allowing the RADIUS server to communicate its state (active/dead) to NAS. Their proposal has effectively helped to improve CPU utilization in the network. The paper explained how to deal with many of server processes such as closing the session after no response from server side (wait specific time) and this helped to reduce time and reduce retransmissions. This research did not consider prevention of replay attacks in the simulation model of the interaction between NAS and RADIUS Server.

Xi Yang et al (2012) used OTP techniques to establish "The Generic Security Service Application Program Interface" (GSS-API) security context between two communicating peers. This compared what was proposed with Kerberos and public-key technologies. While OTP techniques provided greater security for user authentication, SHA-1 algorithms for integrity message was used to enhance the security. This did not prevent replay attack but rather increased the integrity of the Kerberos, which is not a RADIUS protocol.

Hyun *et al* (2013) analyzed the problems of vulnerability of authentication mechanisms by using existing shared key authentication mechanism. CROTP and TOTP used public key infrastructure to solve it. The proposed mechanism can prevent spoofing attack in advance by authenticating user with the use of certificate information, and solved the problems of replay attack, Time synchronization and integrity by generating password though applying hash function for label L

and random value R which are only used in applicable session. Also, they transferred the generated password by electronically signing with the user's private key. The process of generating private key by an individual user to compare it with the public key is a long process and difficult to achieve by users. It is also vulnerable to attacks by malware programs since the private key generated will be stored on an individual's system.

Jonghoon *et al* (2013) introduced a new protocol to assure more secure authentication. This protocol did not only prevent cloning the OTP generator, it prevented phishing attack through transaction information. The proposed protocol requires using the OTP generator equipped with keypad. Their protocol enhanced security and provided more robust authentication method than the existing ones. This research developed OTP generator with a keypad, which, however, was not meant to prevent replay attacks.

Xuguang *et al* (2014) suggested a secure dynamic user authentication scheme that is based on a dynamic OTP with both time and space (location). Their schema used time synchronization to add time factor to OTP and effectively improve two-factor authentication to protect users account against various attacks such as phishing attack, replay attack, and perfect-man-in-the-middle attack. This method required more time before synchronization took place thereby keeping a user for longer time before being authenticated. This also led to the increase in response time between the RADIUS Server and the user and prevented the replay attack in RADIUS environment. The CPU overhead for TOTP was less than that of HOTP and CROTP. The research focused mainly on the CPU overhead, algorithm speed, server response time and OTP duration, but the response time was high because of the duration of the OTP, thus this method does not solve the problem of response time in a captive portal environment, but works better in an e-commerce site.

Amna *et al* (2014) considered a set of factors like preventing replay attack, CPU overhead, algorithm speed, server response time and OTP duration. After measuring these factors through an e-learning software based on apache web server, the results showed that the three OTP techniques (TOTP, HOTP and CROTP) with the e-learning software, prevented the replay attack in RADIUS environment, the CPU overhead at TOTP technique is less than the CPU overhead at HOTP and CROTP techniques. The research focused mainly on the apache web server to prevent replay attack. The apache web server does not require low response time when using RADIUS protocol.

Based on this critical review of similar works, there is a need to improve on the performance of the authentication process based on the response time of at least one of the OTP techniques in a RADIUS environment with captive portal on a network. The improved technique is expected to prevent replay attacks with least response time. Having discussed that one of the greatest vulnerabilities of OTP is the response time, this research work is aim at improving the response time of OTP technique

CHAPTER THREE

MATERIALS AND METHOD

3.1 INTRODUCTION

This section provides the detailed descriptions and guides in the designed, configurations, and simulations of the methodology followed in the cause of the research

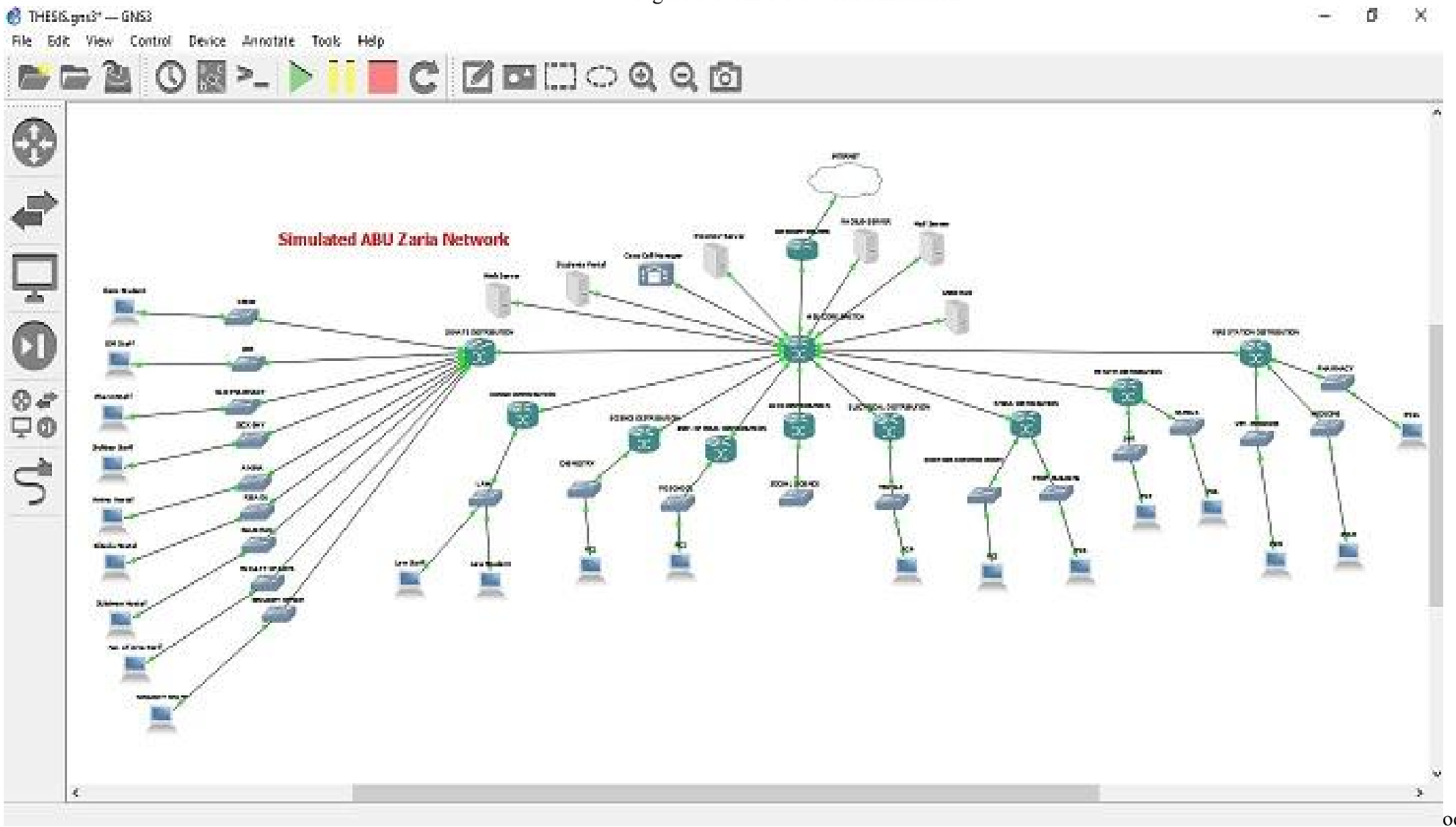
3.2 METHODOLOGY

The methodology adopted is based on the designed, configurations, and simulations of ABU campus network to identify the response time on the three variant of OTP in a RADIUS environment, using captive portal, GNS3 and Virtualbox simulation software's were used to configure the network devices and the installation of the Linux operating system with free RADIUS 2.0 with firewall on the virtualbox environment, the Network Access Server which provides the captive portal was also installed on the virtualbox. The proposed improved OTP techniques was also adopted with the result output showed some significant improvement on all the three variant of OTP used with the TOTP having the highest improvement on the response time both with the simulation and during validation with the active devices.

3.3 ABU CAMPUS NETWORK MODELING AND SIMULATION USING GNS3

ABU campus network (as shown in Figure 3.1) was simulated in GNS3 simulation environment. The routers in the topology were configured to run using Mikrotik router operating system to simulate the real live ABU network design and configuration. The authentication server uses Ubuntu server 14.0 with firewall and FreeRadius2 installed on it, the three OTP variant will also be implemented on this server together with the OTP generator. While the windows machine is used as the user's machine which are connected to the virtualbox using the GNS3 simulator.

Figure 3.1: ABU Network Simulation



odel using GNS3

3.4 RADIUS SERVER ENVIRONMENT

The RADIUS Server environment for this system was prepared through a set of steps. These steps are shown as follows:

1) Ubuntu Linux operating system version 14 is used for the installation of Freeradius2 and the developed OTP generators. As Linux grows in terms of value to running critical applications, the need to manage Linux environments to high standards of service quality, availability, security, and performance becomes an essential requirement for success. The operating system is installed on the kernel based virtual machine.
2) The virtual machine (VM) with the Kernel-based Virtual Machine (KVM) software that is built into the operating system was created. The Oracle VM Virtualbox is a server virtualization based on multi operating system.
3) The freeRADIUS2 is installed and use putty.exe configuration tool to open sessions as shown in Figure 3.2 on type to be SSH on port 22. SSH is a secure shell against eavesdropping and encrypt data during traffic.
4) The Freeradius2 was installed together with the MySQL server. The MySQL server stores all the usernames and passwords for the users, also the logs for the response time will be stored in MySQL database, the database can be accessed via either command line tool using putty or the graphical user interface using PhPmyadmin.

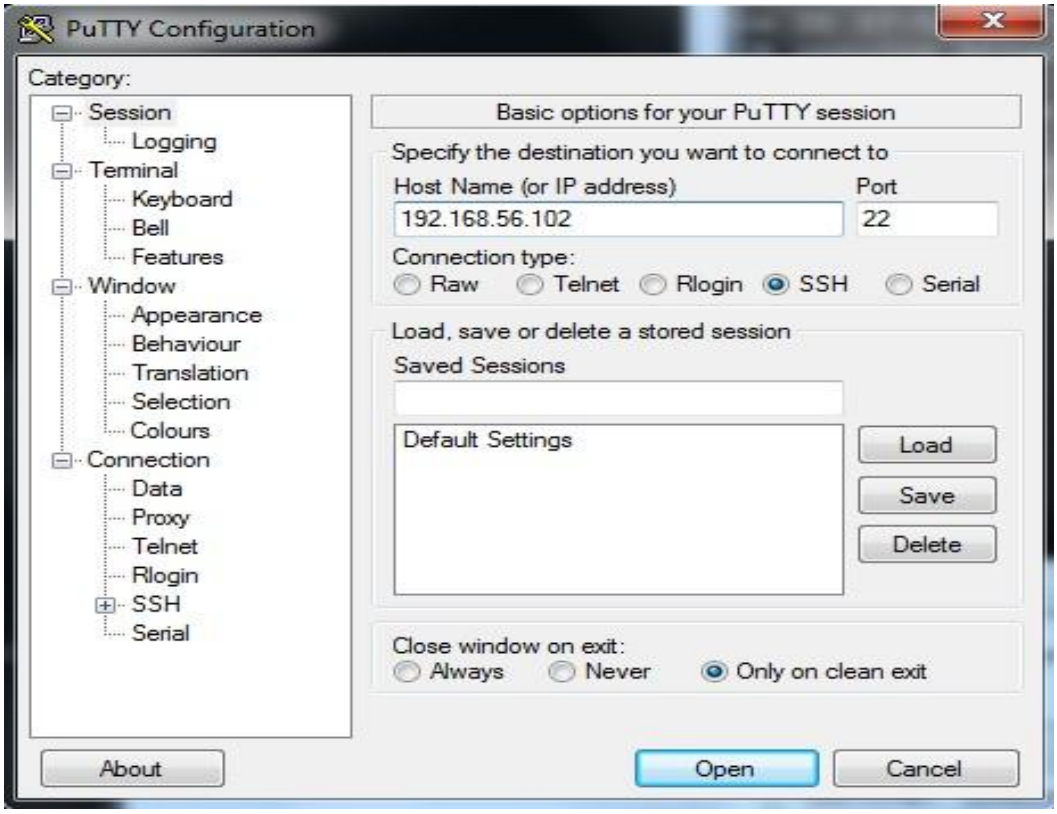


Figure 3.2: Putty SSh Client

To run the RADIUS server, enter the user name and password for RADIUS and then type the command "*radiusd –x* "as shown in Figure 3.3.

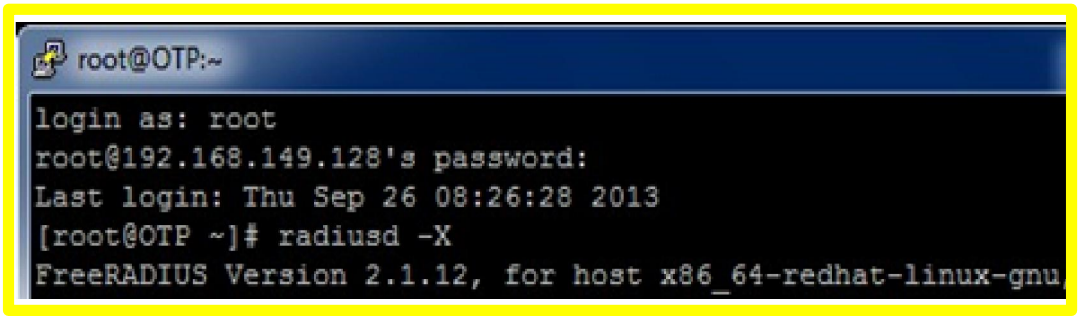


Figure 3.3: Accessing RADIUS Server

Thus, the RADIUS server will display a ready to process any request from the user as shown in Figure 3.4.



Figure 3.4: Running RADIUS Server

3.5 CAPTIVE PORTAL IMPLEMENTATION

The captive portal was implemented on the Network Access Server (NAS) which also resides on the Ubuntu Linux server, the captive portal is developed using the PhP programming language, the captive portal is the screen that appears when a user access resources on the network, it consists of fields such as the login, password, self-registration, drop down button to select different types of OTP technique. Figure 3.5 shows the developed page for user authentication. This page is similar to the captive portal in Figure 2.4 with the improved version of the page, there is now additional field for OTP technique and self-registration which are missing on the current ABU network captive portal. The source code for the implemented captive portal is shown in Appendix II

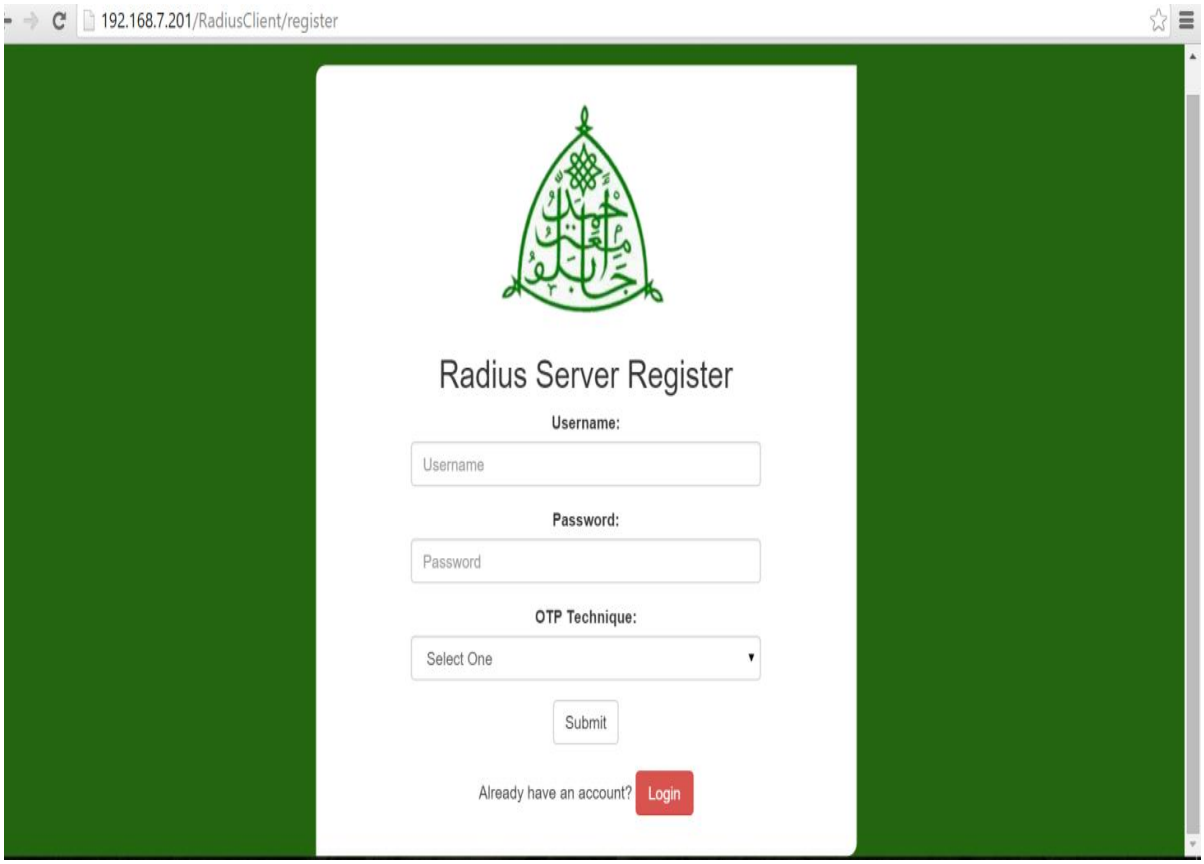


Figure3.5: Captive Portal

3.6 OTP GENERATION SCENARIOS

This sub-section describes the OTP generation scenario to show how the user OTP is generated through OTP application. The OTP application is developed using a PhP developed script. After the user is authenticated (sign-in process) by RADIUS server, the system transmits the user to the OTP page which is an important page in the login process in the captive portal. The user must open the OTP application and enter the PIN and then get the OTP which is generated by one of the three OTP variants. Finally, the user takes the OTP and enters the PIN and OTP in the OTP page of the system. After that the user can login to the system if the matching process is verified and certified ok by the authentication server. Figs. 3.6, 3.7 and 3.8 show the developed OTP application for the generation of OTP for the three variants of OTP techniques, using the algorithm for each technique. The source code for the developed OTP application is shown in Appendix I.



Figure 3.6: Developed HOTP Generator

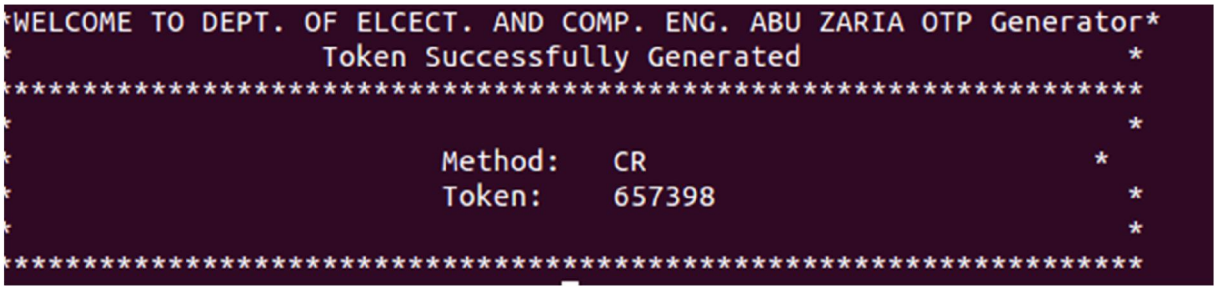


Figure 3.7: Developed CROTP Generator

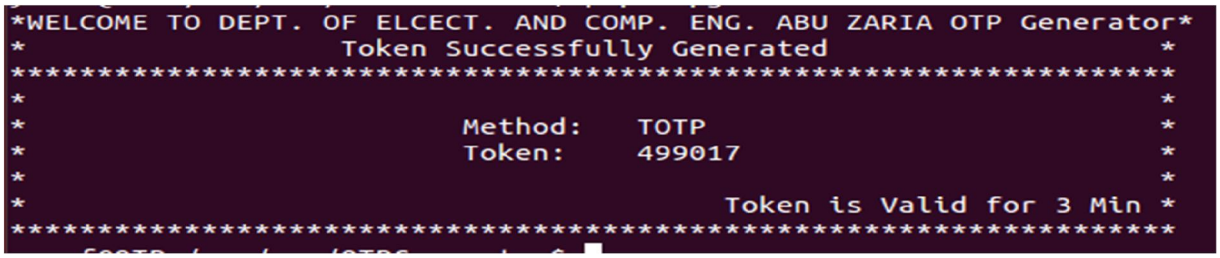


Figure 3.8: Developed TOTP Generator.

3.7 OBTAINING USERNAMES, PIN AND OTP

The user can register self on the server from the captive portal, then prompt the user to select desired PIN, then the user will generate the desires OTP from the three techniques using the developed OTP generator as shown in Figure 3.5

3.8 USER LOGIN SCENARIO

The user login scenario offers a set of cases for login process in the RADIUS server. In table 3.1, each case passes the system user in several stages in order to login to the network. These cases are mentioned as follows:

Case 1: The valid user

a) A user login to the network by providing username and password.

b) The RADIUS server sends a response accepting the credentials.

c) The system redirects the user to OTP page, the user provides PIN and OTP.

d) The system sends the PIN and OTP to authentication server (AS) which checks PIN, last OTP and secret key and then AS sends an acceptance response to the system.

e) The authentication result in this case is successful and the user can login to network.

Case 2: The missing user name

a) A user attempts to login to the network without providing username.

b) The RADIUS server sends a rejection response.

c) The system prevents this user from login to the network.

Case 3: Invalid password

a) A user login to the system by providing valid username and invalid password.

b) The RADIUS server sends a rejection response.

c) The system does not response to user

Case 4: Invalid OTP

a) A user login to the network by providing username and password.

b) The RADIUS server sends a response accepting the credentials.

c) The system redirects the user to the OTP page, the user provides valid PIN and invalid OTP.

d) The system sends the PIN and the invalid OTP to AS which verifies PIN, last OTP and secret key and then AS sends a rejection response.

e) The authentication result of this case is failed.

Case 5: Invalid PIN

a) A user login to the server by providing valid username and password.

b) The RADIUS server sends a response accepting the credentials

c) The system redirects the user the OTP page, and the user enters invalid PIN and valid OTP.

d) The system sends the PIN and the OTP to AS which verifies PIN, last OTP and secret key and then AS sends a rejection response.

e) The authentication result of this case failed.

The summary of the results is as shown in Table 3.1

Table 3.1: User Login Cases

	RADIUS Server			***OTP Techniques***			***Authentication server***				
Cases	*Authentication request*	*Username*	*Password*	*TOTP*	*HOTP*	*CROTP*	*OTP*	*PIN*	*Last OTP*	*Secret key*	*Authentication Result*
Case 1	***V***	***V***	***V***	***V***			***V***	***V***	***V***	***V***	*S*
Case 2	***V***	*I*	*Na*	*Na*			*Na*	*Na*	*Na*	*Na*	*F*
Case 3	***V***	***V***	*I*	*I*			*Na*	*Na*	*Na*	*Na*	*F*
Case 4	***V***	***V***	***V***	***V***			*V*	*I*	*Na*	*Na*	*F*
Case 5	***V***	***V***	***V***	***V***			***V***	*I*	*I*	*I*	*F*

KEY: V: Valid information; I: Invalid information; Na: Not applicable; S: authentication successful; F: authentication failed.

3.9 THE IMPROVED OTP TECHNIQUE

After obtaining the response time for each of the three OTP variants, it can be seen that the TOTP variant had the least response time (screen shot shown in Figure 3.9 shows the data taken during the test).

id	username	reqtime	restime	activity
1	p18900	1447419898	1447419899	TOTP
2	p18901	1447420040	1447420042	TOTP
3	p18902	1447510055	1447510056	TOTP
4	p18903	1447510225	1447510226	TOTP
5	p18904	1447510511	1447510512	TOTP
6	p18905	1447510737	1447510738	TOTP
7	p18906	1447513402	1447513403	TOTP
8	p18907	1447514101	1447514103	TOTP
9	p18908	1447514384	1447514386	TOTP
10	p18909	1447514696	1447514697	TOTP
11	p18900	1447505190	1447505193	HOTP
12	p18901	447504833	1447504834	HOTP
13	p18902	1447505111	1447505114	HOTP
14	p18903	1447505521	1447505523	HOTP
15	p18904	1447505985	1447505986	HOTP
16	p18905	1447506645	1447506647	HOTP
17	p18906	1447507065	1447507067	HOTP
18	p18907	1447507511	1447507513	HOTP
19	p18908	1447507873	1447507874	HOTP
20	p18909	1447508422	1447508424	HOTP
21	p18900	1447578609	1447578611	CROTP
22	p18901	1447578922	1447578924	CROTP
23	p18902	1447579244	1447579245	CROTP
24	p18903	1447579535	1447579536	CROTP
25	p18904	1447579869	1447579871	CROTP
26	p18905	1447580235	1447580237	CROTP
27	p18906	1447580739	1447580741	CROTP
28	p18907	1447581432	1447581433	CROTP
29	p18908	1447581927	1447581928	CROTP
30	p18909	1447582628	1447582631	CROTP

30 rows in set (0.00 sec)

Figure 3.9: Response Time for the Three OTP Variants

The OTP generator is merged with the RADIUS server using the PHP-developed script in order to eliminate the time taken to synchronize between the OTP generator and the RADIUS server. The improved OTP flowchart is shown in the Figure3.10. The source code for the improved OTP technique is shown in Appendix II.

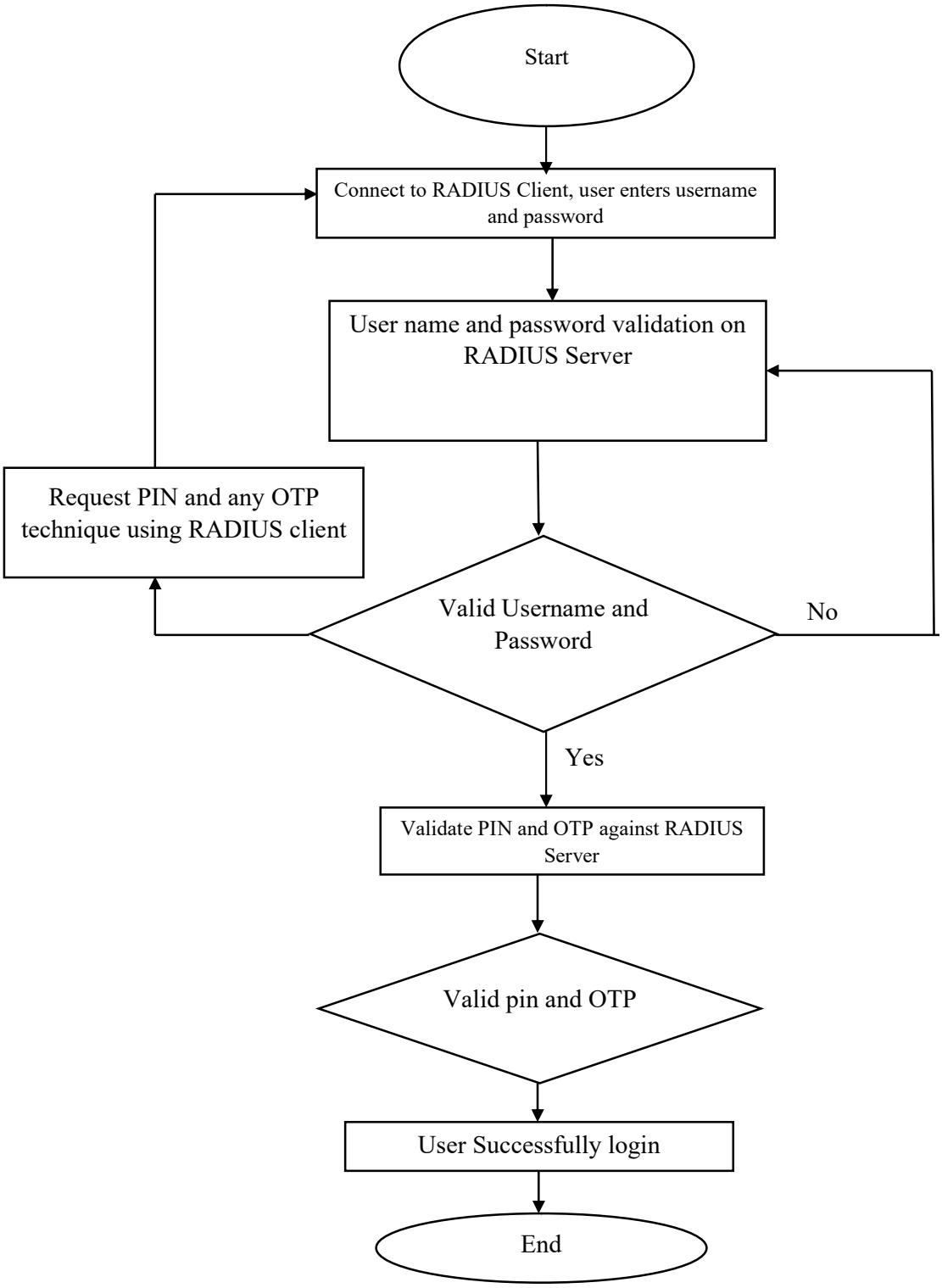


Figure 3.10: Flowchart for the improved OTP Technique

3.10 VALIDATION

The simulated ABU network environment using GNS3 was adopted and configured on a live server, using the following devices:

1) Dell power edge server with 8GB of RAM, 500GB of Hard disk drive and 2.3Ghz processor with Linux operating system, FreeRadius 2.0 and Firewall were installed on the server.

2) Mikrotik router with router operating system version 6.35

3) cisco switches 3750 and 2960

were used for the validation process in ABU Zaria Data Centre. plate 3.1 show the validation setup on one of the equipment rack in the ABU Data Centre. After obtaining the response time for each of the three OTP variants, it can be seen that still the TOTP variant had the least response time (screen shot shown in Table. 3.2 shows the data taken during the test).



Plate 3.1 Validation Setup

Table 3.2: Validation Results for Response Time

Case	Request time	Response time	Technique Type	Drift Second
1	1463735334	1463735336	TOTP	2
2	1463735711	1463735712	TOTP	1
3	1463736020	1463736020	TOTP	0
4	1463737558	1463737558	TOTP	0
5	1463737743	1463737743	TOTP	0
6	1463738712	1463738713	TOTP	1
7	1463739705	1463739705	TOTP	0
8	1463740402	1463740402	TOTP	0
9	1463741234	1463741234	TOTP	0
10	1463741629	1463741629	TOTP	0
11	1464080937	1464080938	HOTP	1
12	1464081291	1464081293	HOTP	2
13	1464081729	1464081730	HOTP	1
14	1464082154	1464082155	HOTP	1
15	1464082709	1464082712	HOTP	3
16	1464083079	1464083079	HOTP	0
17	1464083418	1464083419	HOTP	1
18	1464083917	1464083917	HOTP	0
19	1464084208	1464084208	HOTP	0
20	1464085323	1464085324	HOTP	1
21	1464088388	1464088391	CROTP	3
22	1464088788	1464088789	CROTP	1
23	1464089073	1464089074	CROTP	1
24	1464089744	1464089744	CROTP	0
25	1464090479	1464090479	CROTP	0
26	1464090913	1464090914	CROTP	1
27	1464091894	1464091895	CROTP	1
28	1464092302	1464092304	CROTP	2
29	1464093222	1464093222	CROTP	0
30	1464094031	1464094032	CROTP	1

CHAPTER FOUR

RESULTS AND DISCUSSIONS

4.1 INTRODUCTION

This chapter discusses the average results obtained in the cause of simulation of the response time for authentication server at each OTP technique, before the improvement of the technique, during the simulation of the improved technique and in the cause of validation processes.

4.2 DATA COLLECTION AND ANALYSIS

The modeled ABU campus network in GNS3 modeler environment is simulated for ten users averagely to represent a response time for each user using the three variant of OTP, the average response time were taken before improving, during simulation and during the validation process.

To measure the server response time, ten different cases are offered for each of the three OTP techniques before and after improving the OTP technique. These cases show the server response time for each request by the system user. Table 4.1 shows the user request time and its response time by the server before improving the OTP technique.

The Unix time stamp is a way to track time as a running total of seconds. This count starts at the Unix Epoch on January 1st, 1970 UTC. The Unix time stamp is merely the number of seconds between a particular date and the Unix Epoch. For example, the user request time for case 1 is equal to 1447419898 as timestamp. This case is equal to 13/11/2015, 3:04:58pm as current date using a timestamp converter.

From the Table 4.1, the average response time before improving these techniques are TOTP 2.5sec., HOTP is 5.7s and CROTP is 5.2sec. This results were obtained when the OTP generators and the RADIUS servers were separated. From these results it can be

concluded that the TOTP has the least response time, while the HOTP has the highest response time. These values are higher than the recommended response time for a captive portal in a RADIUS environment which is approximately 1000ms. The response time for each variant is as shown in Figs. 4.1 - 4.3

Table 4.1: Server Response Time for Three OTP Variants

Case	Request time	Response time	Technique Type	Drift Second
1	1447419892	1447419895	TOTP	3
2	1447420040	1447420044	TOTP	4
3	1447510055	1447510056	TOTP	1
4	1447510225	1447510226	TOTP	1
5	1447510511	1447510517	TOTP	6
6	1447510737	1447510738	TOTP	1
7	1447513402	1447513404	TOTP	2
8	1447514101	1447514106	TOTP	5
9	1447514384	1447514385	TOTP	1
10	1447514696	1447514697	TOTP	1
11	1447505190	1447505196	HOTP	6
12	1447504833	1447504837	HOTP	4
13	1447505111	1447505119	HOTP	8
14	1447505521	1447505525	HOTP	4
15	1447505981	1447505988	HOTP	7
16	1447506642	1447506650	HOTP	8
17	1447507060	1447507068	HOTP	8
18	1447507511	1447507516	HOTP	5
19	1447507873	1447507878	HOTP	4
20	1447508422	1447508425	HOTP	3
21	1447578601	1447578619	CROTP	8
22	1447578922	1447578930	CROTP	8
23	1447579244	1447579246	CROTP	2
24	1447579532	1447579538	CROTP	6
25	1447579869	1447579873	CROTP	4
26	1447580235	1447580241	CROTP	6
27	1447580739	1447580744	CROTP	5
28	1447581432	1447581437	CROTP	5
29	1447581927	1447581932	CROTP	5
30	1447582628	1447582631	CROTP	3

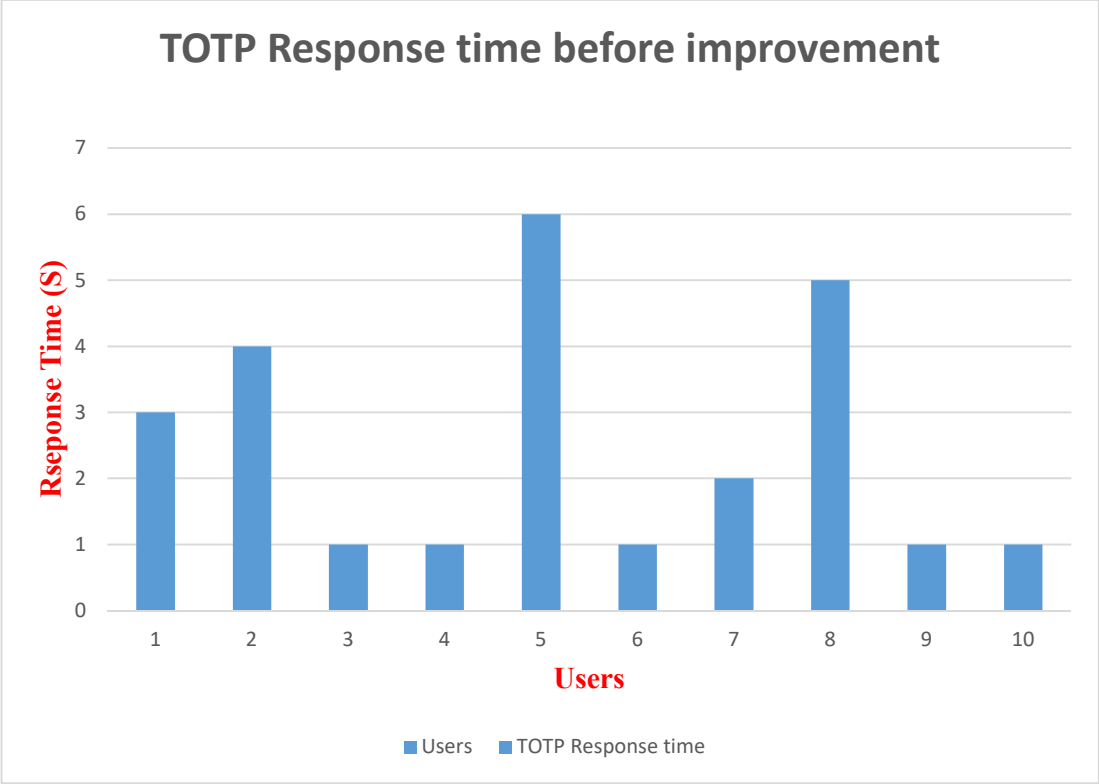


Figure 4.1: TOTP Response time

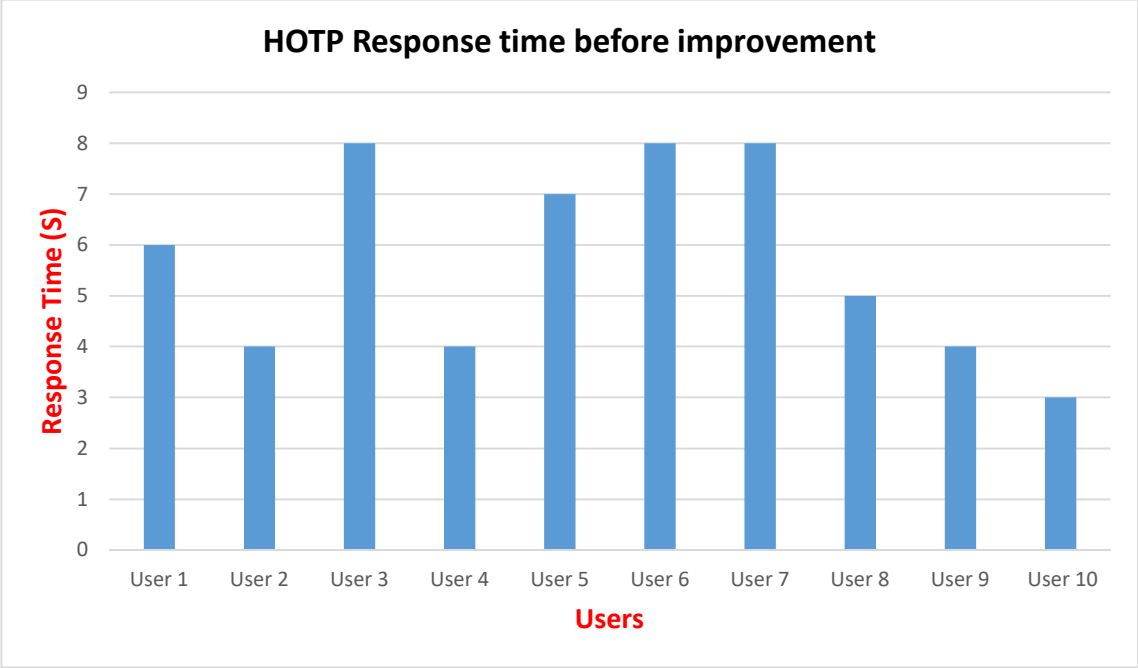


Figure 4.2: HOTP Response Time

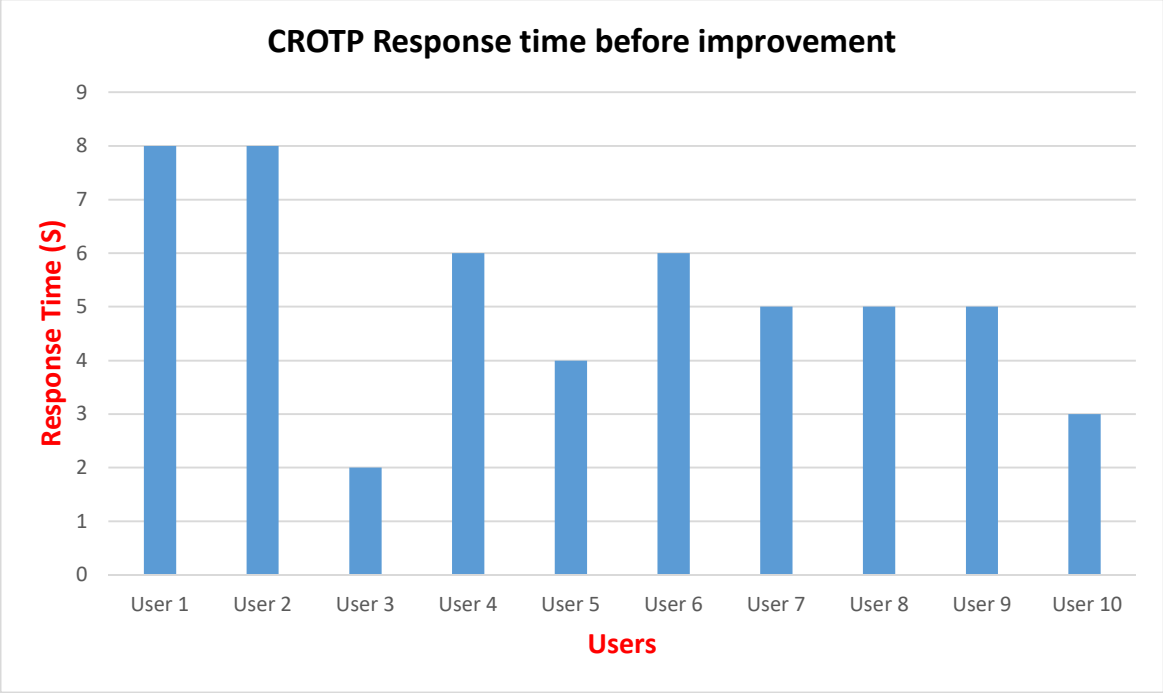


Figure 4.3: CROTP Response Time

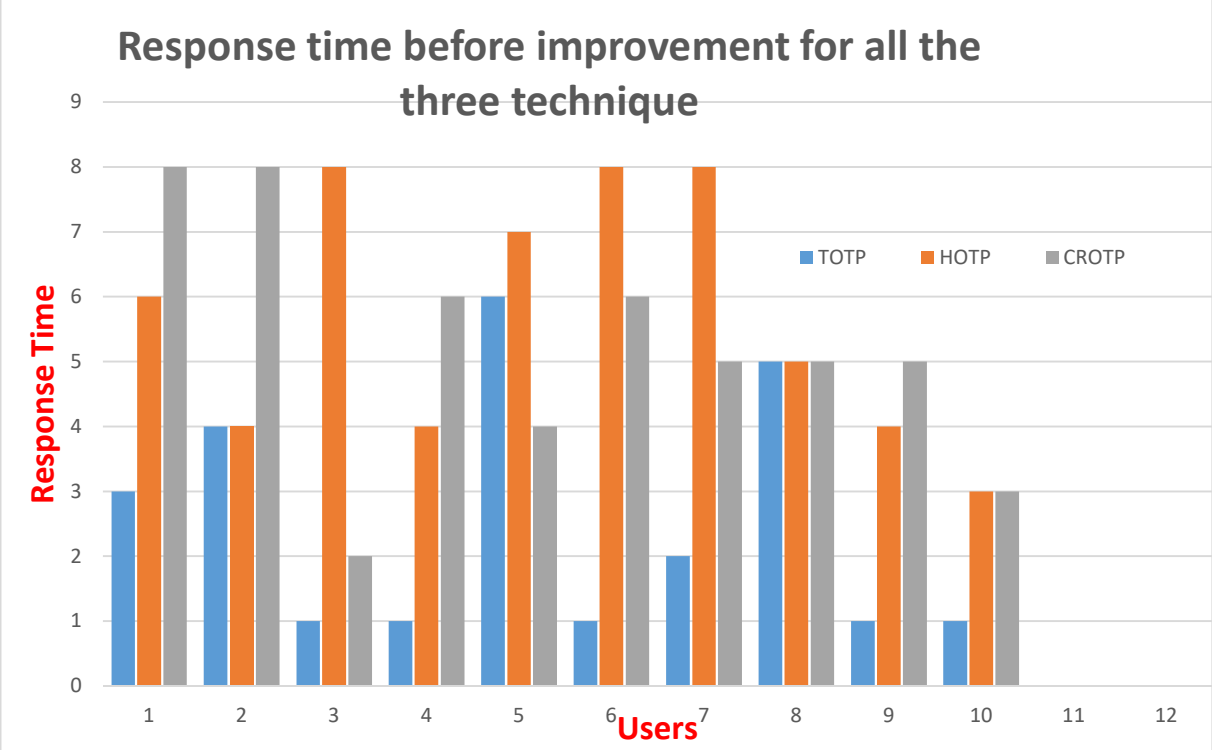


Figure 4.4: Combined results before improvement

4.3 IMPROVED OTP TECNIQUE

After the implementation of the modified OTP technique by integrating all the OTP techniques to resides in a RADIUS server rather than using the OTP generator separately for each technique. Table 4.2 shows the results for the modified techniques.

From Table 4.2 the average response time for the TOTP is 1.2s, that of CROTP is 2s and that of HOTP is 1.9s. From these results, it can be concluded that the TOTP has the least average response time, while the HOTP and CROTP have the highest average response times. The value for the TOTP is close the recommended response time for a captive portal in a RADIUS environment which is approximately 1000ms. This delay is due to the simulation software which require a lot of processing power. The response time for each technique is shown in Figs. 4.5 - 4.7, while Figure 4.8 shows the combined results o these results.

Table 4.2: Response time for the improved OTP technique

Case	Request time	Response time	Technique Type	Drift Second
1	1447419898	1447419899	TOTP	1
2	1447420040	1447420041	TOTP	1
3	1447510055	1447510057	TOTP	2
4	1447510225	1447510226	TOTP	1
5	1447510511	1447510512	TOTP	1
6	1447510737	1447510738	TOTP	1
7	1447513402	1447513403	TOTP	1
8	1447514101	1447514103	TOTP	2
9	1447514384	1447514385	TOTP	1
10	1447514697	1447514698	TOTP	1
11	1447505190	1447505193	HOTP	3
12	1447504833	1447504835	HOTP	2
13	1447505111	1447505113	HOTP	2
14	1447505521	1447505523	HOTP	2
15	1447505985	1447505986	HOTP	1
16	1447506645	1447506648	HOTP	3
17	1447507065	1447507067	HOTP	2
18	1447507511	1447507512	HOTP	1
19	1447507873	1447507874	HOTP	1
20	1447508422	1447508424	HOTP	2
21	1447578609	1447578612	CROTP	3
22	1447578922	1447578925	CROTP	3
23	1447579244	1447579247	CROTP	3
24	1447579535	1447579536	CROTP	1
25	1447579869	1447579871	CROTP	2
26	1447580235	1447580236	CROTP	1
27	1447580739	1447580740	CROTP	1
28	1447581432	1447581434	CROTP	2
29	1447581927	1447581929	CROTP	2
30	1447582628	1447582630	CROTP	2

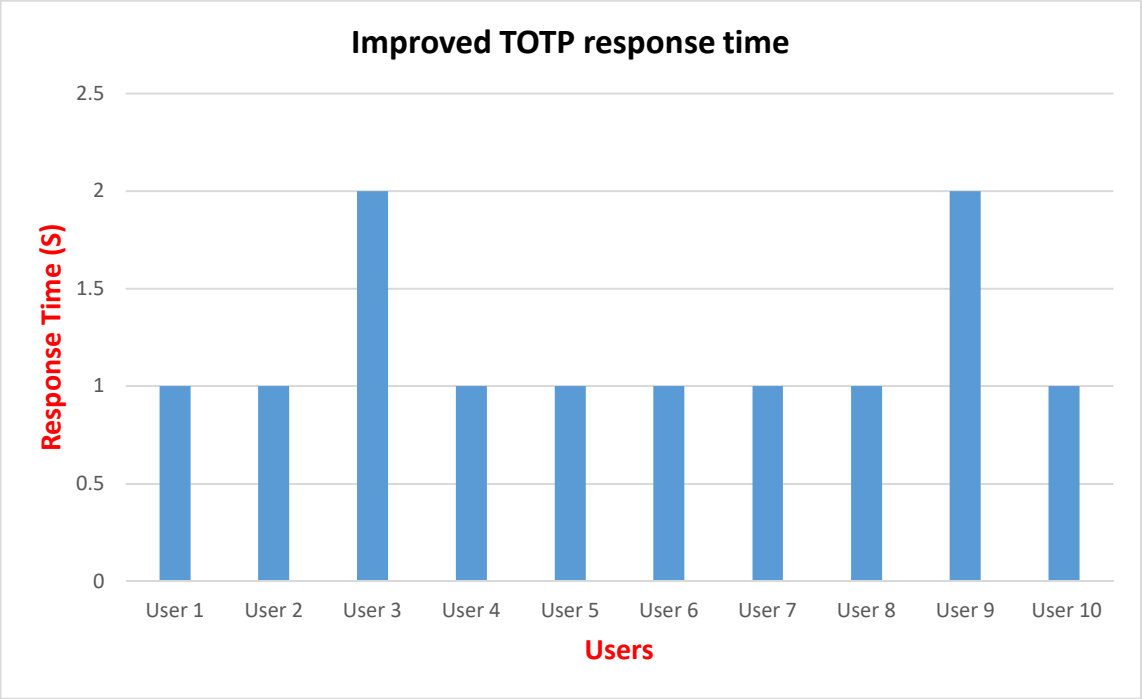


Figure 4.5: Improved TOTP Response Time

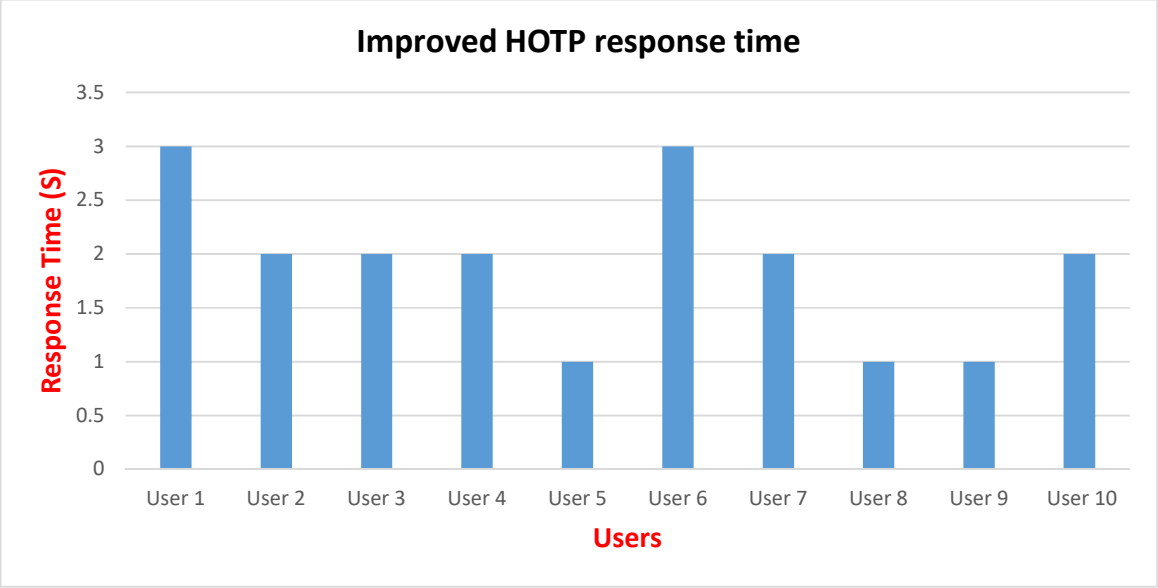


Figure 4.6: Improved HOTP Response Time

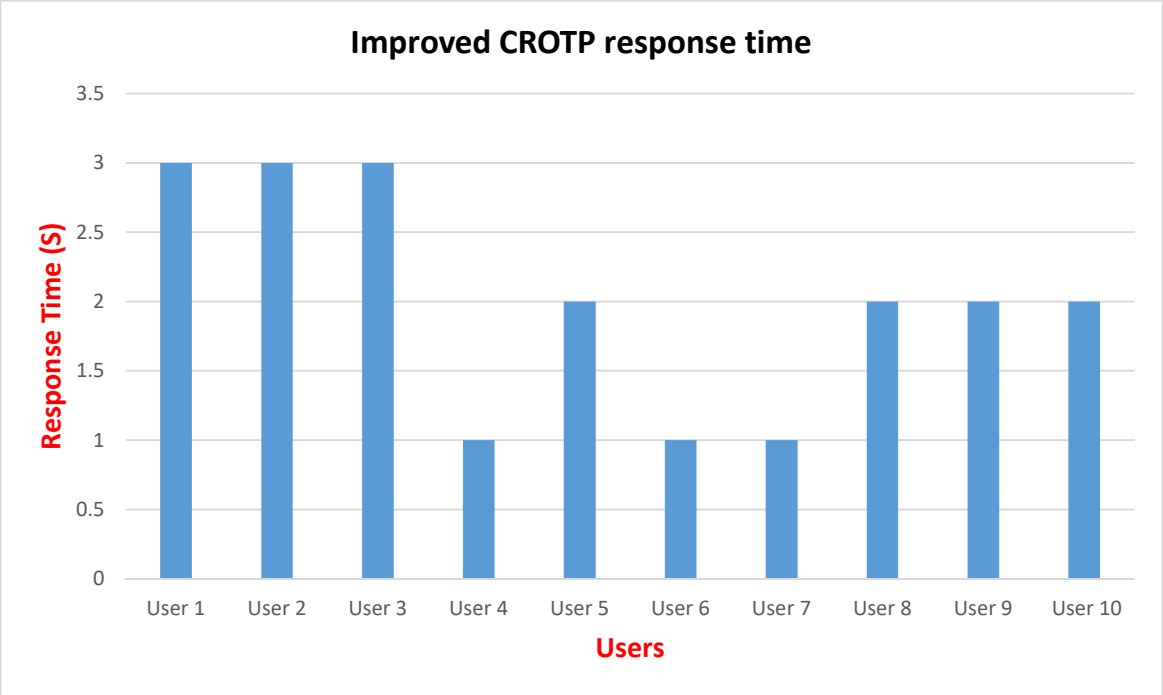


Figure 4.7: Improved CROTP Response Time

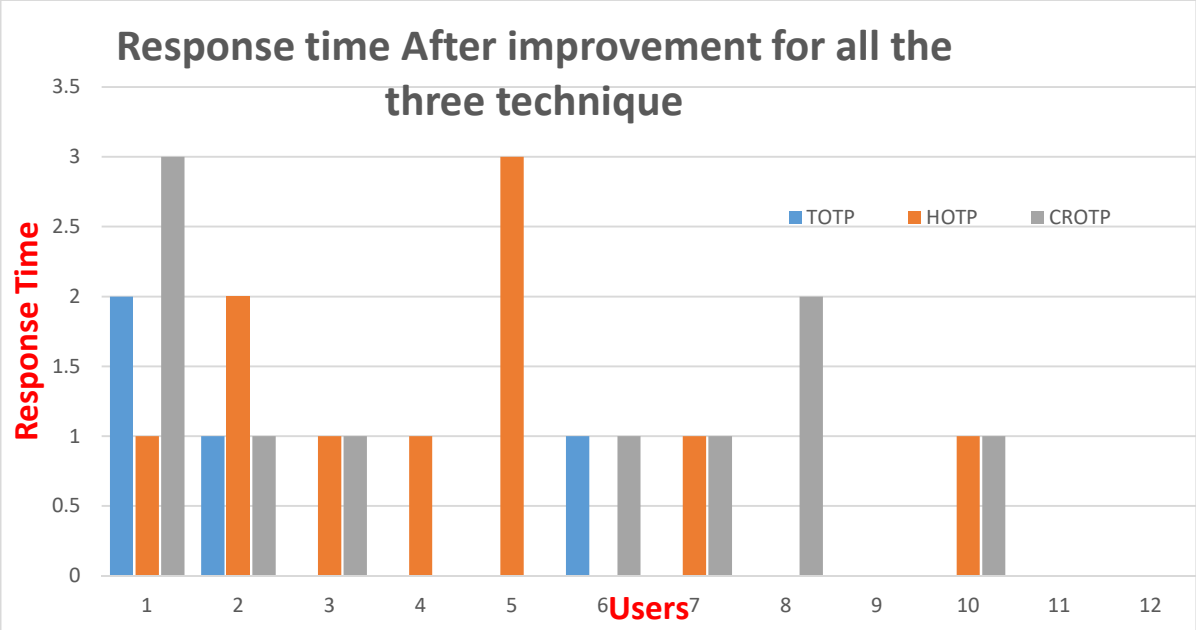


Figure 4.8: Combined results after improvement

4.4 VALIDATION

After the implementation of the improved OTP technique by integrating all the OTP techniques to resides in a RADIUS server rather than using the OTP generator separately for each technique using the live server and routers as shown in plate 3.1 for validation of the results. Table 4.3 shows the validation results for the modified techniques.

From Table 4.3 the average response time for the TOTP is 0.4s, that of CROTP is 1.0s and that of HOTP is 1.0s. From these results it can be concluded that the TOTP has the least response time, while the HOTP and CROTP have the highest average response times. The average response time for each technique is shown in Figs. 4.9 - 4.11, while Figure 4.12 shows the combine results obtained for the three techniques.

Table 4.3: Validation Results for Response Time

Case	Request time	Response time	Technique Type	Drift Second
1	1463735334	1463735336	TOTP	2
2	1463735711	1463735712	TOTP	1
3	1463736020	1463736020	TOTP	0
4	1463737558	1463737558	TOTP	0
5	1463737743	1463737743	TOTP	0
6	1463738712	1463738713	TOTP	1
7	1463739705	1463739705	TOTP	0
8	1463740402	1463740402	TOTP	0
9	1463741234	1463741234	TOTP	0
10	1463741629	1463741629	TOTP	0
11	1464080937	1464080938	HOTP	1
12	1464081291	1464081293	HOTP	2
13	1464081729	1464081730	HOTP	1
14	1464082154	1464082155	HOTP	1
15	1464082709	1464082712	HOTP	3
16	1464083079	1464083079	HOTP	0
17	1464083418	1464083419	HOTP	1
18	1464083917	1464083917	HOTP	0
19	1464084208	1464084208	HOTP	0
20	1464085323	1464085324	HOTP	1
21	1464088388	1464088391	CROTP	3
22	1464088788	1464088789	CROTP	1
23	1464089073	1464089074	CROTP	1
24	1464089744	1464089744	CROTP	0
25	1464090479	1464090479	CROTP	0
26	1464090913	1464090914	CROTP	1
27	1464091894	1464091895	CROTP	1
28	1464092302	1464092304	CROTP	2
29	1464093222	1464093222	CROTP	0
30	1464094031	1464094032	CROTP	1

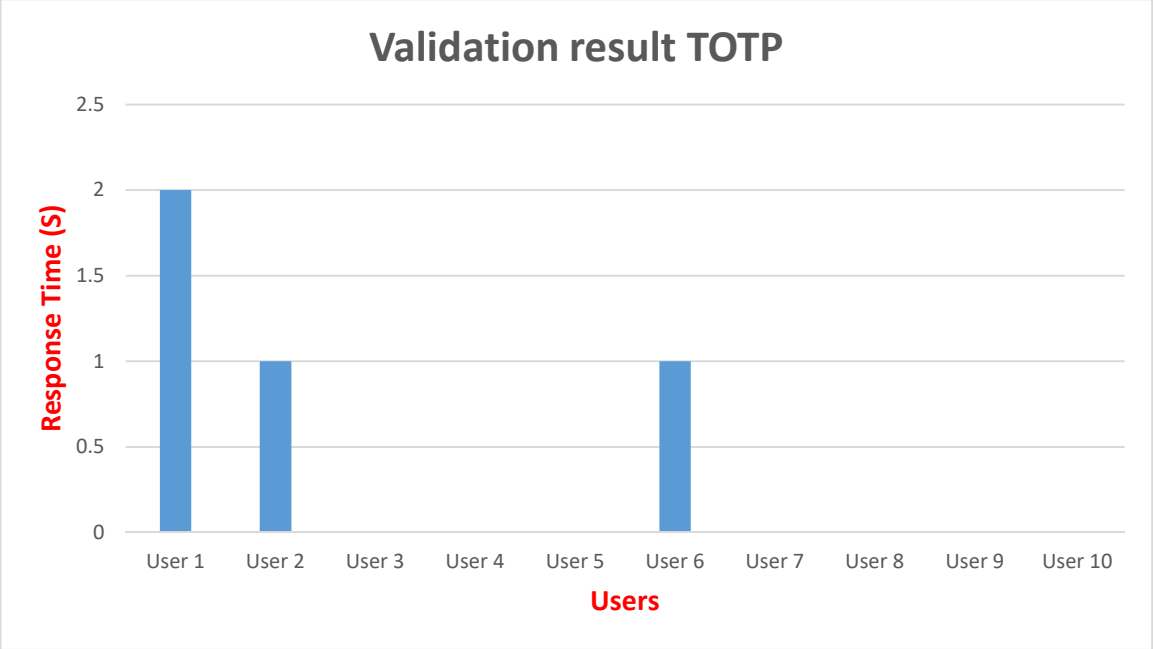


Figure 4.9: Validation TOTP Response Time

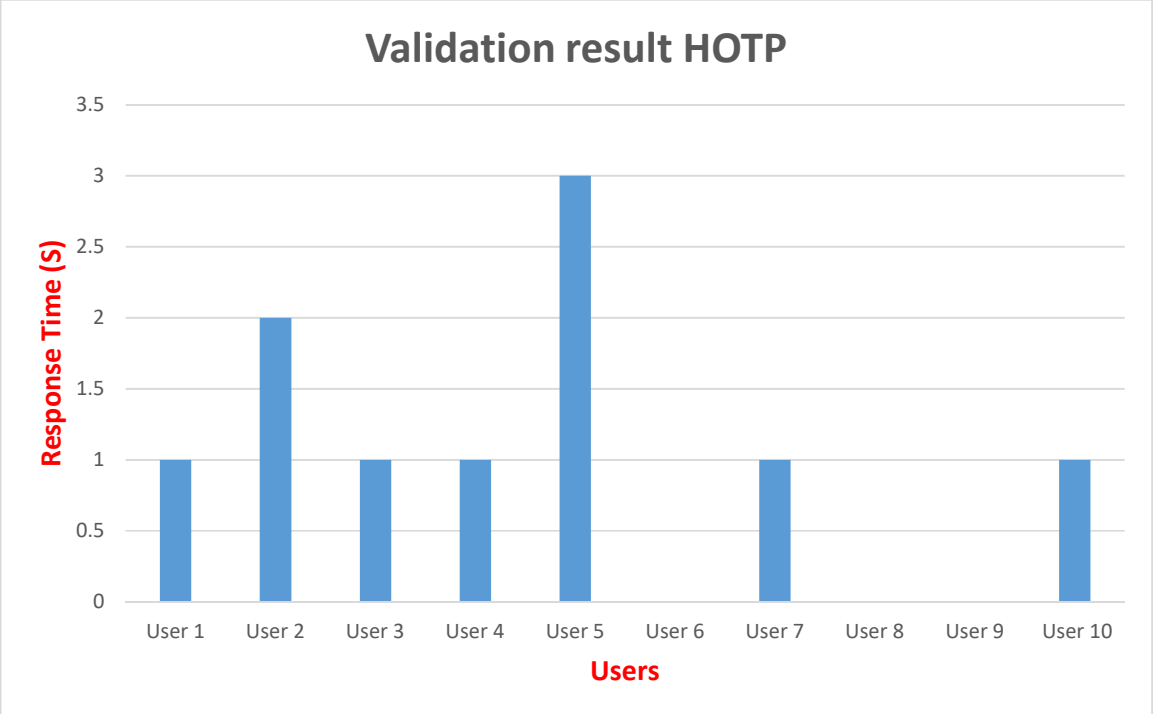


Figure 4.10: Validation HOTP Response Time

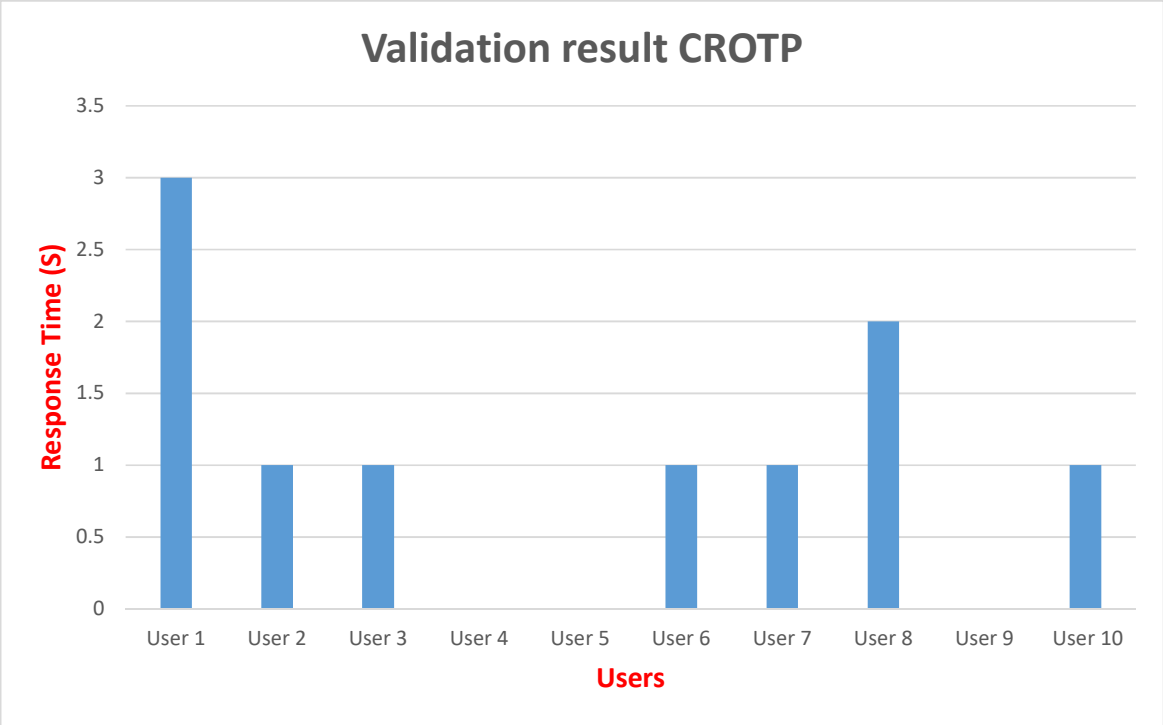


Figure 4.11: Validation CROTP Response Time

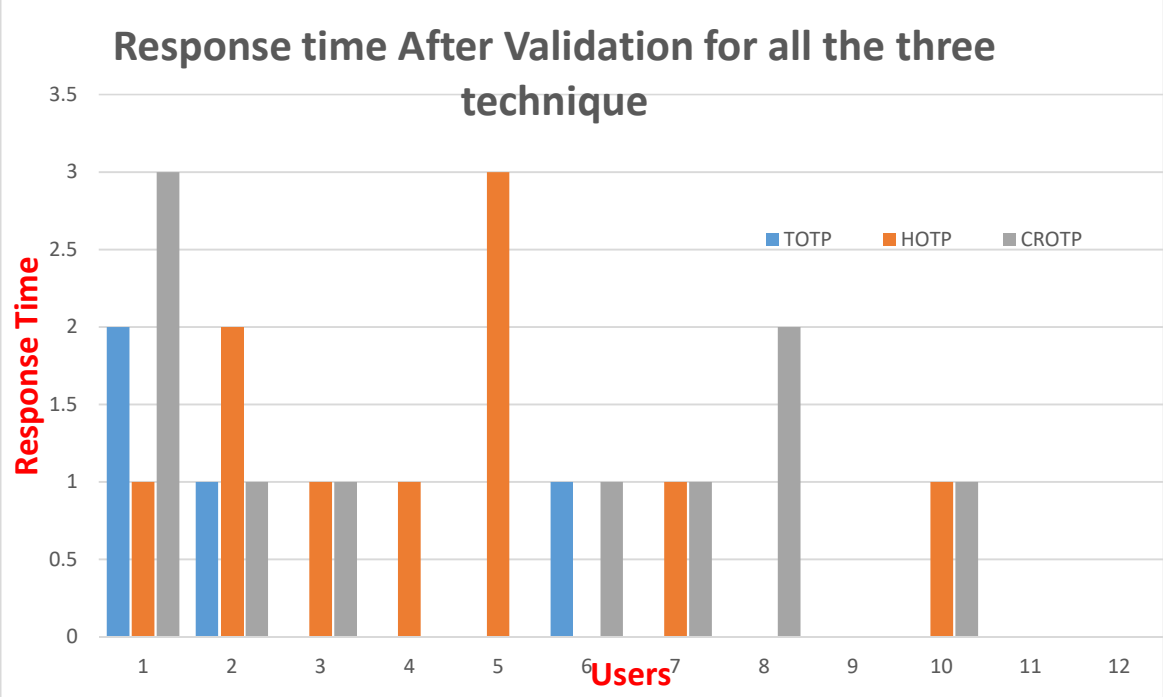


Figure 4.12: Combined results after validation

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 INTRODUCTION

The sub-section presents the summary of the research, limitation, conclusion, recommendations, and further work.

5.2 SIGNIFICANT CONTRIBUTION

Nowadays, user authentication has become one of the most important issue in the security trend. RADIUS protocol provides remote services for user authentication, but there are some vulnerabilities in RADIUS protocol such as the replay attack, which this research is based on. Therefore, the significant contributions of this research work are as follows:

1. Development of an improved php-based OTP generator for the three variants of the OTP: TOTP, CROTP and HOTP, which was successfully integrated into the Linux server application.
2. The average response time obtained for TOTP, CROTP and HOTP using the improved OTP generator of 0.4s, 0.9s and 0.9s improvement compared with the response times obtained using the standard OTP generator for TOTP, CROTP and HOTP of 2.5s, 5.2s and 5.7s respectively.

5.3 CONCLUSION

The RADIUS protocol with vulnerability of replay attack and the three techniques to prevent the replay attack in a RADIUS environment with captive portal is implemented. This research work is aimed at preventing a replay attack on users on a network by evaluating the three variants of OTP techniques with the aim of selecting and adopting the one that has the best response time for further improvement. Previous studies have suggested several approaches to reduce the effect of replay attack such as PKI, PSK,

IPsec, session key, sets clock and OTP techniques, in order to enhance the security in RADIUS environment. From the results obtained this research work has shown the improved response time of all the OTP techniques compares with the previous result. The result obtained is lower than the recommended response time of a RADIUS server in a captive portal environment which is 1000ms. The TOTP technique is the recommended technique to adopt having the lowest response time.

5.4 LIMITATIONS

During the course of this research work, certain limitations were observed which are itemized as follows:

1. GNS3 emulator consumes a lot of computer resources during simulation; a computer system with very high specs is required.
2. The Virtualbox was not connecting to the GNS simulator.

5.5 RECOMMENDATION FOR FUTURE WORK

In this section, a set of points are drawn to extend for future work and recommendations. these points are as follows:

1. Integrates model that can forward secret key to user via the SMS service or email.
2. Develop a batch file that will connect the user to the server using private key infrastructure PKI, for users not having to login to the RADIUS server for OTP generation.
3. The length of the generated OTP in this work is 5 characters. Further research should consider increasing the length of the OTP since longer OTP makes the system more secure.

REFERENCES

Alan, H. ,. (2013). A Methodology for Analyzing the Performance of Authentication Protocols. *IEEE, 5*(10).

Amna. (2014). *Implementing and Comprising of OTP Techniques (TOTP,HOTP,CROTP) to Prevent Replay Attack in RADIUS Protocol using ELSBOT*. Retrieved from http://library.iugaza.edu.ps/thesis/113489.pdf

Bellare, H. N. (2011). CROTP: OATH Challenge-Response Algorithm. *IEEE*.

Binod, V. (2013). HOTP-Based User Authentication Scheme in Home Networks. *IJCSNS International Journal of Computer Science and Network Security*, 1-10.

Bray, T. (2009, 03 16). *Processing Instructions*. Retrieved November 12, 2015, from Extensible Markup Language (XML): www.w3.org

Etutorials. (2015). *Access Control: IEEE 802.1X, EAP, and RADIUS*. Retrieved from etutorials: http://etutorials.org/Networking/802.11+security.+wi-fi+protected+access+and+802.11i/Part+II+The+Design+of+Wi-Fi+Security/Chapter+8.+Access+Control+IEEE+802.1X+EAP+and+RADIUS/

Florian, D. L. (2010). Secure protocol implementation. *IEEE*, 179=182.

Gagan, D. N. (2013). replay attack prevention in Kerberos authentication protocol using triple password. *International Journal of Computer Networks & Communications (IJCNC), 5*(2).

Harry, J. (2010, March 23). *Using the GNS3 Network Simulator*. Retrieved December 10, 2015, from Pluralsight blog: http://blog.pluralsight.com/using-gns3-network-simulator

Hassel, J. (2003). *Securing Public Access to Private Resources RADIUS* (First Edition ed.). (J. Sumser, Ed.) California CA: O'Reilly & Associates, Inc.

Havard, R. L. (2010). Security Analysis of Mobile Phones Used as OTP Generators. *International Federation for Information Processing (IFIP)* , 324-331, .

Himika, P. N. (2012). Generation Of Secure One- Time Password Based On Image Authentication. *IEEE*, 195-206.

Hyun-Chul, K. (2013). *A Design of One-Time Password Mechanism using Public Key Infrastructure*. Retrieved from http://www.diva-portal.org/smash/get/diva2:512701/FULLTEXT01.pdf

Jaakko, T. (2014). *Overview, details and analysis of Radius protocol*. Retrieved from Cisco how radius works, Authentication, Authorization and Accounting: http://www.cisco.com/image/gif/paws/12433/32.pdf

Jonghoon, L. J. (2013). Advanced OTP Authrntication Protocol Using PUF. *The Fifth International Conference on Evolving Internet IARIA*, (pp. 48-51). London.

Kenneth, G. P. (2010). One-time-password-authenticated key exchange. *Australasian Conference on Information Security and Privacy (ACISP)*.

Kun Tang. (2014). A Dynamic User authentication based on time and space. *IJCSNS International Journal of computer science and network security, volume 15*(13).

Lammle, T. (2014). *"CCNA Routing and Switching Review Guide".* Sybex.

Lanners. (2014, March 10). *Process Modeling & Simulation Software Explained*. Retrieved December 1, 2015, from Process Modeling & Simulation Software Explained: http://www.lanner.com/en/simulation-explained.cfm

Lapiedra, J. (2002). *The Information Security Process Prevention, Detection and Response.* SANS Institute.

Leidigh, C. (2013). Fundamental Principles of Network Security. *Schneider*, 99-120.

Lerdorf, R. (20013, April 26). *presentation by Rasmus Lerdorf given at the MySQL Conference in Santa Clara*. Retrieved 12 18, 2015, from php.net: php.net

M'Raihi. (2011). TOTP: Time based one time password algorithm. *Internet Engineeting Task Force (IETF)*.

M'Raihi, D. (2011). TOTP: Time-Based One-Time Password Algorithm. *Internet Engineering Task Force IETF*.

Mikko, S. (2004). *Legacy User Authentication with IPSEC*. Retrieved from http://www.tml.tkk.fi/Publications/Thesis/msaarinen.pdf

Mikrotik. (2013). *Mikrotik Documentation*. Retrieved August 15, 2015, from Radius Client: http://wiki.mikrotik.com/wiki/Manual:RADIUS_Client

Namrata, T. V. (2013). An Approach of Authentication in Public Cloud using Two Step Verification Code. *International Journal of Emerging Research in Management &Technology, 2*(5).

Rao, V. (2014). Server Virtualization and Network Virtualization in Cloud Computing. *IAES, 3*(4).

Reddy, S. (2014). VIRTUALIZATION IN CLOUD COMPUTING. *IJCSMC, 3*(5), 540-546.

Rigney, C. S. (2007). Remote Authentication Dial In User Services (RADIUS). *Internet Engineering Task Force (IETF)*.

Rohan, D. (2012). Interactive Remote Authentication Dial In User Service (RADIUS) Authentication Server Model. *International Conference on Wireless and Mobile Communication ICWMC*, (pp. 238-241). Boston.

Saba, S. A. (2012). Network Simulation Tools Survey. *International Journal of Advanced Research in Computer and Communication Engineering*, 1-10. Retrieved January 10th, 2016

Scroggins, R. (2013). Virtualization Technology. *Global Journal of Computer Science and Technology Interdisciplinary, 13*(1).

Sung-Jae. (2011). Low-Power Design of Hardware One-Time Password Generators for Card-Type OTP. *ETRI Journal, Volume 33*(Issue 4).

Trupti, H. (2012). remote client authentication using mobile phone. *International Journal of Scientific and Research Publications, Volume 2*(Issue 5), 5549-5555.

Tse, C. (2014). "An Anti-Replay Window Protocol with Controlled Shift". *International Conference on Advanced Information Networking and Applications*.

Xue Liu, L. S. (2003). *Online response time optimization of Apache web server.* Berlin: springer-verlad.

Xuguang, R. (2012). TSPass: A Dynamic User. *International Journal of Computer Science and Network Security IJCSNS, 12*(10).

Yang, X. (2012). *The Use of One-Time Password and RADIUS Authentication in a GSS-API Architecture*. Retrieved from http://www.diva-portal.org/smash/get/diva2:512701/FULLTEXT01.pdf

Yeh, T. (2009). A Secure One-Time Password Authentication Scheme Using Smart Cards. *IEICE Transaction on Communication*, 2515-2518.

APPENDIX I

Developed OTP Generators for the three OTP techniques.

```
$interval = 10;
$ timeInUnit = floor($time/$interval);
$i = 0;
$ epoch = $timeInUnit – 18;
while ($i < 36) {
$otp = $epoch.$user->getSecret().$pin;
$otp = md5($otp(;
$epoch++;
$i++;}

$No
 beginning of roll =counter client .
$roll end =counter + MOE Function sync_OTP
{
$ Counter=Counter+1;
$i=0;
$While (i<=50)
$Current counter=counter client –i;

For (i=0;i<i*2;i++)
$Counter++;
$i++;
}

<?php
namespace OTPHP { /**
 * One Time Password Generator
  * The OTP class allow the generation of one-time
 * password that is described in rfc 4xxx.
class OTP {
    /**
     * The base32 encoded secret key
     * @var string
     */
    public $secret;
    /**
     * The algorithm used for the hmac hash function
     * @var string
     */
    public $digest;
    /**
```

```
     * The number of digits in the one-time password
     * @var integer
     */
    public $digits;
    /**
     * Constructor for the OTP class
     * @param string $secret the secret key
     * @param array $opt options array can contain the
     * following keys :
     *   @param integer digits : the number of digits in the one time password
     *   @param string digest : the algorithm used for the hmac hash function
         * @return new OTP class.
     */
    public function __construct($secret, $opt = Array()) {
      $this->digits = isset($opt['digits']) ? $opt['digits'] : 6;
      $this->digest = isset($opt['digest']) ? $opt['digest'] : 'sha1';
      $this->secret = $secret;
    }
    /**
     * Generate a one-time password
     *
     * @param integer $input : number used to seed the hmac hash function.
     * This number is usually a counter (HOTP) or calculated based on the current
     * timestamp (see TOTP class).
     * @return integer the one-time password
     */
    public function generateOTP($input) {

      $hash = hash_hmac($this->digest, $this->intToBytestring($input), $this-
>byteSecret());
      foreach(str_split($hash, 2) as $hex) { // stupid PHP has bin2hex but no hex2bin WTF
        $hmac[] = hexdec($hex);
      }
      $offset = $hmac[19] & 0xf;
      $code = ($hmac[$offset+0] & 0x7F) << 24 |
        ($hmac[$offset + 1] & 0xFF) << 16 |
        ($hmac[$offset + 2] & 0xFF) << 8 |
        ($hmac[$offset + 3] & 0xFF);
      return $code % pow(10, $this->digits);
    }
    /**
     * Returns the binary value of the base32 encoded secret
     * @access private
     * This method should be private but was left public for
     * phpunit tests to work.
     * @return binary secret key
         public function byteSecret() {
      return \Base32::decode($this->secret);
    }
    /**
     * Turns an integer in a OATH bytestring
    * @param integer $int
```

```
  * @access private
   * @return string bytestring
   */
  public function intToBytestring($int) {
    $result = Array();
    while($int != 0) {
      $result[] = chr($int & 0xFF);
      $int >>= 8;
    }
    return str_pad(join(array_reverse($result)), 8, "\000", STR_PAD_LEFT);
  }
 <?php
        require './otphp/lib/otphp.php';
        require './conFigurephp';
        //echo $config['secret']." \n";
        $techniques = array('HOTP', 'TOTP', 'CR');
        if(isset($argv[1]) && isset($argv[2]) && in_array($argv[1], $techniques)) {
                $technique              =       $argv[1];
                $pin                    =       $argv[2];
                if($technique == 'HOTP') {
                        $counter = file_get_contents("./counter.conf");
                        $hotp = new \OTPHP\HOTP($config['secret'].$pin);
                        $token = $hotp->at($counter);
echo "*WELCOME TO DEPT. OF ELCECT. AND COMP. ENG. ABU ZARIA OTP
Generator*
                        echo "*              Token Successfully Generated              * \n";
                        echo
"************************************************************** \n";
                        echo "*                                                * \n";
                        echo "*                  Method:   ".$technique."                  *
\n";
                        echo "*                 Token:    ".$token."                * \n";
                        echo "*                                                * \n";
                        echo
"************************************************************** \n";
                        file_put_contents("./counter.conf", (int)$counter+1);
                }else if($technique == 'TOTP') {
                        $totp = new \OTPHP\TOTP($config['secret'].$pin, array('interval'
=> 180));
                        $token = $totp->now();


echo "*WELCOME TO DEPT. OF ELCECT. AND COMP. ENG. ABU ZARIA OTP
Generator*
                        echo "*              Token Successfully Generated              * \n";
                        echo
"************************************************************** \n";
                        echo "*                                                * \n";
                        echo "*                  Method:   ".$technique."                  *
\n";
                        echo "*                 Token:    ".$token."                * \n";
                        echo "*                                                * \n";
```

```
                        echo "*                                Token is Valid for 3 Min * \n";
                        echo "
********************************************************************** \n";
                }else if($technique == 'CR') {
                        $counter = file_get_contents("./counter.conf");
                        $cr = new \OTPHP\HOTP($counter.$config['secret'].$pin);
                        $token = $cr->at($counter);
echo "*WELCOME TO DEPT. OF ELCECT. AND COMP. ENG. ABU ZARIA OTP
Generator*
                        echo "*              Token Successfully Generated                * \n";
                        echo
"********************************************************************** \n";
                        echo "*                                                          * \n";
                        echo "*                    Method:   ".$technique."                      *
\n";
                        echo "*                  Token:    ".$token."                   * \n";
                        echo "*                                                          * \n";
                        echo
"********************************************************************** \n";
                        file_put_contents("./counter.conf", (int)$counter+1);
        }else {
echo "*WELCOME TO DEPT. OF ELCECT. AND COMP. ENG. ABU ZARIA OTP
Generator*
                echo "*                                                          * \n";
                echo "*    Error: Use Generator as: php otpgen [technique] [pin]       * \n";
                echo "*                        --techniques (HOTP, TOTP, CR)       * \n";
                echo "*                                                          * \n";
                echo "
********************************************************************** \n";
        }
?>
```

APPENDIX II

Improved OTP Technique

```
<?php
use Helpers\Assets;
use Helpers\Url;
use Helpers\Session;
?><div class="mainContainer">
	<img src="<?php echo Url::templatePath(); ?>images/logo.png" style="max-width: 100%;" />	<h2>Radius Server OTP</h2>
	<?php
	echo Session::message('error');
	if(!empty(Session::get('pin'))) {
	echo "<p>Generate Token Using PIN: <b>".Session::pull('pin')."</b> & Technique: <b>".Session::pull('technique')."</b></p>";
	}else if(!empty(Session::get('technique'))) {
	echo "<p>Generate Token Using Technique: <b>".Session::pull('technique')."</b></p>";
	?>
	<form action="<?php echo DIR; ?>otp" method="POST" role="form" style="max-width: 70%; margin: auto;">
		<div class="form-group">
			<label for="pin">PIN:</label>
			<input type="text" class="form-control" id="pin" name="pin" placeholder="PIN">
		</div>
		<div class="form-group">
			<label for="token">Token:</label>
			<input type="password" class="form-control" id="token" name="token" placeholder="Token">
		</div> <button type="submit" class="btn btn-default">Submit</button>
	</form> <br/>
	<p>Logged In as <?php echo Session::get('username'); ?> <a href="<?php echo DIR; ?>logout" class="btn btn-danger">Logout</a></p>
</div>
<?php
use Helpers\Assets;
use Helpers\Url;
use Helpers\Session;
?> <div class="mainContainer">
	<img src="<?php echo Url::templatePath(); ?>images/logo.png" style="max-width: 100%;" />	<h2>Radius Server Register</h2>
	<?php
	echo Session::message('error');
	?>
	<form action="<?php echo DIR; ?>register" method="POST" role="form" style="max-width: 70%; margin: auto;">
		<div class="form-group">
			<label for="username">Username:</label>
			<input type="text" class="form-control" id="username" name="username" placeholder="Username" required>
```

```
                </div>
                <div class="form-group">
                        <label for="password">Password:</label>
                        <input type="password" class="form-control" id="password"
name="password" placeholder="Password" required>
                </div>
        <div class="form-group">
                        <label for="technique">OTP Technique:</label>
                        <select class="form-control" id="technique" name="technique"
required>
                                <option>Select One</option>
                                <option value="HOTP">HOTP</option>
                                <option value="TOTP">TOTP</option>
                                <option value="CR">Challenge Response</option>
                </select>
                </div>
        <button type="submit" class="btn btn-default">Submit</button>
        </form>         <br/>
        <p>Already have an account? <a href="<?php echo DIR; ?>" class="btn btn-
danger">Login</a></p> </div>
<?php
use Helpers\Assets;
use Helpers\Url;
use Helpers\Session;
?>
<div class="mainContainer">
        <img src="<?php echo Url::templatePath(); ?>images/logo.png" style="max-width:
100%;" />
        <h2>Radius Server Secret</h2>
        <p> Registration Successfull, your secret is : <b><?php echo
Session::pull('secret'); ?></b>          </p>  <br/>
        <p><a href="<?php echo DIR; ?>" class="btn btn-primary">Login Now</a></p>
</div>
<?php
use Helpers\Assets;
use Helpers\Url;
use Helpers\Session;
?>
<div class="mainContainer">
        <img src="<?php echo Url::templatePath(); ?>images/logo.png" style="max-width:
100%;" />
        <h2>Radius Server Secured Page</h2>
        <p>This is a test secured page.</p>
        <br/>
        <p>Logged In as <?php echo Session::get('username'); ?> <a href="<?php echo
DIR; ?>logout" class="btn btn-danger">Logout</a></p>
# -*- text -*-
##
## clients.conf -- client configuration directives
##       $Id$
#########################################################################
#  Define RADIUS clients (usually a NAS, Access Point, etc.).
```

```
# Defines a RADIUS client.
 '127.0.0.1' is another name for 'localhost'.  It is enabled by default,
# to allow testing of the server after an initial installation.  If you
# are not going to be permitting RADIUS queries from localhost, we suggest
# that you delete, or comment out, this entry.
# Each client has a "short name" that is used to distinguish it from
# other clients.
# In version 1.x, the string after the word "client" was the IP
# address of the client.  In 2.0, the IP address is configured via
# the "ipaddr" or "ipv6addr" fields.  For compatibility, the 1.x
# format is still accepted.
client localhost {
        # Allowed values are:
        #       dotted quad (1.2.3.4)
        #     hostname    (radius.example.com)
        ipaddr = 127.0.0.1
        # OR, you can use an IPv6 address, but not both
        # at the same time.
#       ipv6addr = ::   # any.  ::1 == localhost
        #
        # A note on DNS:  We STRONGLY recommend using IP addresses
        # rather than host names.  Using host names means that the
        # server will do DNS lookups when it starts, making it
        # dependent on DNS.  i.e. If anything goes wrong with DNS,
        # the server won't start!
        # The server also looks up the IP address from DNS once, and
        # only once, when it starts.  If the DNS record is later
        # updated, the server WILL NOT see that update.
        # One client definition can be applied to an entire network.
        # e.g. 127/8 should be defined with "ipaddr = 127.0.0.0" and
        # "netmask = 8"
        # If not specified, the default netmask is 32 (i.e. /32)
        # We do NOT recommend using anything other than 32.  There
        # are usually other, better ways to achieve the same goal.
        # Using netmasks of other than 32 can cause security issues.
        # You can specify overlapping networks (127/8 and 127.0/16)
        # In that case, the smallest possible network will be used
        # as the "best match" for the client.
        # Clients can also be defined dynamically at run time, based
        # on any criteria.  e.g. SQL lookups, keying off of NAS-Identifier,
        # etc.
        # See raddb/sites-available/dynamic-clients for details.
        netmask = 32
          # The shared secret use to "encrypt" and "sign" packets between
        # the NAS and FreeRADIUS.  You MUST change this secret from the
        # default, otherwise it's not a secret any more!
        # The secret can be any string, up to 8k characters in length.
        # Control codes can be entered vi octal encoding,
        #       e.g. "\101\102" == "AB"
        # Quotation marks can be entered by escaping them,
        #       e.g. "foo\"bar"
        # A note on security:  The security of the RADIUS protocol
```

```
#  depends COMPLETELY on this secret!  We recommend using a
#  shared secret that is composed of:
#       upper case letters
#       lower case letters
#       numbers
#  And is at LEAST 8 characters long, preferably 16 characters in
#  length.  The secret MUST be random, and should not be words,
#  phrase, or anything else that is recognizable.
#  The default secret below is only for testing, and should
#  not be used in any real environment.
secret          = testing123
#  Old-style clients do not send a Message-Authenticator
#  in an Access-Request.  RFC 5080 suggests that all clients
#  SHOULD include it in an Access-Request.  The configuration
#  item below allows the server to require it.  If a client
#  is required to include a Message-Authenticator and it does
#  not, then the packet will be silently discarded.
#  allowed values: yes, no
require_message_authenticator = no
#  The short name is used as an alias for the fully qualified
#  domain name, or the IP address.
#  It is accepted for compatibility with 1.x, but it is no
#  longer necessary in 2.0
shortname       = localhost
# the following three fields are optional, but may be used by
# checkrad.pl for simultaneous use checks
# The nastype tells 'checkrad.pl' which NAS-specific method to
#  use to query the NAS for simultaneous use.
#  Permitted NAS types are:
        cisco
        computone
        livingston
        max40xx
        multitech
        netserver
        pathras
        patton
        portslave
        tc
        usrhiper
        other
nastype     = other       # localhost isn't usually a NAS...
#  The following two configurations are for future use.
#  The 'naspasswd' file is currently used to store the NAS
#  login name and password, which is used by checkrad.pl
#  when querying the NAS for simultaneous use.
login       = !root
password    = someadminpas
#  As of 2.0, clients can also be tied to a virtual server.
#  This is done by setting the "virtual_server" configuration
#  item, as in the example below.
virtual_server = home1
```

```
        #  A pointer to the "home_server_pool" OR a "home_server"
        #  section that contains the CoA configuration for this
        #  client.  For an example of a coa home server or pool,
        #  see raddb/sites-available/originate-coa
        coa_server = coa
}
# IPv6 Client
#client ::1 {
#       secret          = testing123
#       shortname       = localhost
#}
# All IPv6 Site-local clients
#client fe80::/16 {
#       secret          = testing123
#       shortname       = localhost
#}
#client some.host.org {
#       secret          = testing123
#       shortname       = localhost
#}
#  You can now specify one secret for a network of clients.
#  When a client request comes in, the BEST match is chosen.
#  i.e. The entry from the smallest possible network.
client 192.168.0.0/24 {
#       secret          = testing123-1
#       shortname       = private-network-1
#}
#
#client 192.168.0.0/16 {
#       secret          = testing123-2
#       shortname       = private-network-2
#}
client 172.20.10.13 {
        secret          = testing123
        shortname       = private-network-2
}
#client 10.10.10.10 {
#       # secret and password are mapped through the "secrets" file.
#       secret      = testing123
#       shortname   = liv1
#      # the following three fields are optional, but may be used by
#      # checkrad.pl for simultaneous usage checks
#       nastype     = livingston
#       login       = !root
#       password    = someadminpas
#}
#############################################################################
#
#  Per-socket client lists.  The configuration entries are exactly
#  the same as above, but they are nested inside of a section.
#  You can have as many per-socket client lists as you have "listen"
#  sections, or you can re-use a list among multiple "listen" sections.
```

```
#  Un-comment this section, and edit a "listen" section to add:
#  "clients = per_socket_clients".  That IP address/port combination
#  will then accept ONLY the clients listed in this section.
#clients per_socket_clients {
#         client 192.168.3.4 {
#                 secret = testing123
#       }
#}
```