TRIBHUVAN UNIVERSITY

INSTITUTE OF SCIENCE AND TECHNOLOGY

CENTER DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

KIRTIPUR, KATHMANDU

NEPAL

ANALYTICAL STUDY OF IEEE 802.11i AND ITS IMPROVEMENT

(A Dissertation Submitted as a Partial Fulfillment of the Requirements for the Master Degree in Computer Science and Information Technology)

Anil Awale

September 2007

CERTIFICATION

Mr. Anil Awale has carried out this dissertation work entitled "**Analytical Study of IEEE 802.11i And Its Improvement**" under my supervision and guidance. In my best Knowledge this is an original work in computer science. I, therefore, recommend for further evaluation.

.....

Sharad Ghimire
Institute of Engineering
Pulchowk Campus
Kathmandu, Nepal

We certify that we have read this dissertation and in our opinion, it is satisfactory in the scope and quality as dissertation in the partial fulfillment for the requirement of the Master's Degree in Computer Science and Information Technology.

Evaluation Committee

……	……
Dr. Tanka Nath Dhamala	Sharad Ghimire
Department Head	Supervisor
Center Department of Computer Science And Information Technology	
……	……
External Examiner	Internal Examiner

Date: …….

ABSTRACT

IEEE 802.11i is the standard designed by IEEE to provide the enhanced MAC security in the wireless network. The authentication process consists of three entities; the supplicant (wireless device), the authenticator (access point) and the authentication server (de facto RADIUS server). IEEE 802.11i provides mutual authentication between the network access point and user devices prior to user connectivity. The protocol consists of several parts, including an IEEE 802.1X authentication phase which uses TLS over EAP, the 4-Way Handshake to establish a fresh session key, and an optional Group Key Handshake for group communication. This study analyzes the IEEE 802.11i with respective to data confidentiality, integrity, mutual authentication and availability. Theoretically the analysis of the protocol was done by using Protocol Composition Logic (PCL) and practically it was done by using standard network simulator namely ns2 (Network Simulator 2) and also proposed the improvement to the protocol which was also verified by using PCL theoretically and ns2 practically. IEEE 802.11i appears to provide effective data confidentiality and integrity when CCMP is used. But since 802.11i design does not emphasize availability, several DoS attacks are possible. This study reviews the known DoS attacks on unprotected management frames and EAP frames, and discusses the way to minimize them in 802.11i.

Keywords:

IEEE 802.11i, IEEE 802.1X, RADIUS, 4-Way Handshake, Group Key Handshake, Denial of Service, Authentication, Key Management, Protocol Composition Logics.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisor, Sharad Ghimire, who continuously provided me help and encouragement with extensive knowledge, patient instructions, heuristics ideas and valuable criticism.

I would also like to thank all the family member of Center Department of Computer Science and Information Technology, Tribhuvan University, who have fully supported me in my 2 years study of Master Degree in this department. Special thanks goes to Dr. Tanka Nath Dhamala, Prof. Shashidhar Ram Joshi, Prof. Dr. Devi Dutta Paudyal and all other faculties of the department.

I also owe special thanks to all my colleagues, family and other friends for their direct and indirect support in completing this study.

Anil Awale

CONTENTS

LISTS OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\wedge$	Conjunction
$\vee$	Disjunction
$\neg$	Negation
$\exists$	Existential quantifier
$\diamondsuit \phi$	Some state in the past ϕ holds
$\Theta \phi$	In previous state ϕ holds
$\hat{X}$	Principal containing thread X
$\supset$	Implication

ABBREVIATIONS

AAA	Authentication, Authorization and Accounting
AES	Advance Encryption Standard
AP	Access Point
AS	Authentication Server
ASCII	American Standard Code for Information Interchange
BSS	Basic Service Set
CBC	Cipher Block Chaining
CBC-MAC	Cipher Block Chaining Message Authentication Code
CCM	Counter Mode CBC-MAC
CCMP	Counter Mode/CBC-MAC Protocol
CIA	Confidentiality, Integrity and Availability
CRC	Cyclic Redundancy Check
DoS	Denial of Service
EAP	Extended Authentication Protocol
EAPOL	EAP Over LAN
EAP-SIM	EAP Subscriber Identity Module
EAP-TLS	EAP Transport Layer Security
EAP-TTLS	EAP Tunneled Transport Layer Security
EBSS	Extended Basic Service Set
ESS	Extended Service Set
GEK	Group Encryption Key
GIK	Group Integrity Key
GMK	Group Master Key
GTK	Group Transient Key
IBSS	Independent Basic Service Set
ICV	Integrity Check Value
IEEE	Institute of Electrical and Electronics Engineers
IFS	Inter-Frame Space

IV	Initialization Vector
KCK	Key Confirmation Key
KEK	Key Encryption Key
LAN	Local Area Network
MAC	Medium Access Control
MIC	Message Integrity Check
MITM	Man in The Middle
MPDU	MAC Protocol Data Unit
MSDU	MAC Service Data Unit
MSK	Master Session Key
NAM	Network Animator
NIC	Network Interface Card
NS2	Network Simulator 2
OTCL	Object-oriented TCL
PAE	Port Access Entity
PCL	Protocol Composition Logics
PMK	Pairwise Master Key
PN	Packet Number
PSK	Pre-Shared Key
PTK	Pairwise Transient Key
QoS	Quality of Service
RADIUS	Remote Authentication Dial-In User Service
RFC	Request For Comment
RSN	Robust Security Network
RSN IE	RSN Information Element
RSNA	Robust Security Network Association
SSL/TLS	Secure Socket Layer/ Transport Layer Security
STN	Station
TCL	Tool Command Language
TK	Temporary Key
TKIP	Temporal Key Integrity Protocol

TLS	Transport Layer Security
TMK	Temporary MIC Key
TPTK	Temporary PTK
TSC	TKIP Sequence Counter
TSN	Transition Security Network
UDP	User Datagram Protocol
VPN	Virtual Private Network
VSA	Vendor Specific Attributes
WEP	Wired Equivalent Protocol
Wi-Fi	Wireless Fidelity
WPA	Wi-Fi Protected Access
WPA2	Wi-Fi Protected Access 2

CHAPTER 1
INTRODUCTION

Broadcast radio and, later, broadcast TV have defined wireless for two generations. The ability for radio waves and TV signals to go anywhere and be heard and seen by anyone has provided huge benefits to the general public. If we have the receiver, this broadcast capability is very attractive, but sometimes for the sender these broadcast qualities can be a major disadvantage.

The military were the first to address the disadvantage of being heard by everyone. To protect communications over radio, the military adapted secret codes that had for many years been used to protect written messages. Techniques such as spread spectrum transmission were invented to try to prevent unwanted reception. In order to protect wireless communication during the Cold War (1950 to 1980), huge advances were made in secure communications, but the general public did not receive any direct benefits from this work.

Because wireless technology has advanced and dropped in price, now almost everyone uses both radio receivers and transmitters e.g. mobile phones, cordless phones, Wi-Fi LANs, and lots of other equipment. After the technology has gone to the wireless, several organizations have developed the wireless devices. The major problems here is that no two of them were compatible i.e. a computer equip with the brand X radio would not work in a room equipped with a brand Y base station. So, in order to solve this sort of problem, the IEEE committee that standardized the wired LANs was given the task of drawing up the wireless LAN standard. The standard they come up and were given the name 802.11 also known as Wi-*Fi (Wireless Fidelity)* [1].

1.1 Issues And Objectives Of The Study

Wireless networks are by nature insecure as all traffic is essentially broadcasted and

anyone with an antenna can pick it up. So, all the traffics should be encrypted in order to protect the privacy of the users.

IEEE 802.11i [2] addresses almost all of the known weakness in the older network security protocols and if implemented properly, it is assumed to be most secure among the wireless technology. But it also consists of certain drawbacks such as,

- Various DoS (*Denial of Service*) attacks are possible [3, 4].
- IEEE 802.11i keying and authentication procedures are too slow to support the real-time applications such as voice.
- Since IEEE 802.11i uses IEEE 802.1X and *RADIUS*, there is an inheritate problems related to these techniques etc [5].

This study mainly focused on the DoS attacks on IEEE 802.11i and improving IEEE 802.11i which will minimize the DoS attacks. So, the main objective of this study is to study IEEE 802.11i in depth and to provide the secure solution which will minimize the possible DoS attacks.

1.2 Wireless Networks Architecture

According to IEEE 802.11, wireless networks operate in three modes:

- Ad Hoc/Independent Basic Service Set (IBSS) Network
- Infrastructure/Basic Service Set (BSS) Network
- Extended Service Set (ESS) Network

1.2.1 Ad Hoc/Independent Basic Service Set (IBSS) Network

In Ad hoc mode, each computer communicates directly (peer-to-peer network). So, it is also known as independent mode. Here we have independent networks with a BSS, (*Basic Service Set*) on each computer. Each station has same BSS. Ad hoc mode is designed such that only the clients within the transmission range (i.e. within the same

cell) of each other can communicate. If the client in an Ad hoc network wishes to communicate outside of the cell, a member of the cell must operate as a gateway and perform the routing.

1.2.2 Infrastructure/Basic Service Set (BSS) Network

In infrastructure mode, all communication goes through the base station, called an A*ccess Point* (AP). The AP acts as the *Ethernet* bridge and forwards the communications onto the appropriate network, either the wired or the wireless. Thus, regardless of the distance between two communicating devices, communication between then must relayed through the AP.

1.2.3 Extended Service Set (ESS) Network

ESS networks are characterized by multiple APs (tuned to the same channel) with overlapping coverage. The distribution services of ESS APs include co-operative engagement to forward data frames from communicating devices associated with other AP in ESS to communicating device in their own BSS. This makes the ESS appear to external network entities as though it was one large stationary *subnet*. In addition, the APs also control seamless handoffs from one AP to another within the ESS to ensure transparent roaming for the communicating device within the overall coverage area.

1.3 Wireless Network Connection Process

The interaction between S*tations* (STNs) in establishing network connection consists of four phases. So, the STN can occupy one of the three different states as shown in Figure 1.1 in order to define relationship with the access point (AP).

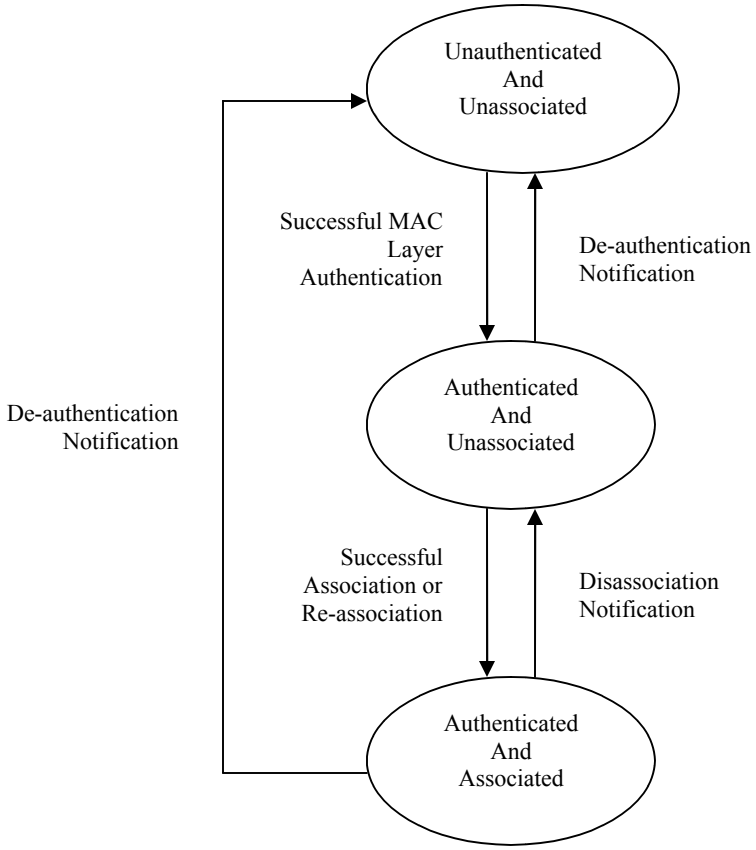


Figure 1.1 Connection States and Services

(Adapted From IEEE 802.11 Handbook, A Designers Companion p. 16)

1.2.1 Unauthenticated And Unassociated

This is the first stage in establishing relationship of the STN with the AP. While establishing the relationship, STN will perform two steps: *Scanning* and *synchronization*. Ad Hoc network do not form complete association connections because they lack mechanism to regulate traffic amongst the participating STNs.

Scanning

Scanning is the process in which STN seeks out other STNs or APs to form connections. It can be done either actively or passively. In *active scanning*, the STN transmits a probe

request management frame in order to get response from other STNs and in *passive scanning* STN listen for a beacon management frame which will be broadcasted by APs.

Synchronization

Synchronization is the process which is used to support *Inter-frame space* (IFS) function to minimize data frame collision. It can be accomplished by transmitting periodic beacon management frames. This function is performed by AP in infrastructure network and in case of Ad Hoc network; it is shared by all STNs.

1.2.2 Authenticated And Unassociated

Authentication is the process of validating the identity of a STN in the network. The process of authentication depends on the protocol used for securing wireless network.

1.2.3 Authenticated And Associated

Association is the final stage in the process linking from STN to an AP. Although the STN may be simultaneously authenticated with numerous APs, it can be associated with only one AP at a time. This prevents confusion in determining which AP is providing service in an EBSS environment.

De-authentication and disassociation management frames enables an AP to downgrade the connection state of one or more STNs.

1.4 Security Principles

Security is collection of tools that were designed to protect data and other resources from third person (hacker). The main goal of security system is to make as though there are no hacker. While designing the security system, we should not concentrate on only one

security issues and also we should not ignore security weakness just because they have no consequences.

Since we are dealing with wireless security, we cannot control one threat; radio jamming. One can prevent communication by transmitting a jamming signal. In other words, the hacker will still be able to demonstrate their presence by blocking communication. But if we design our security protocols correctly, and install them correctly, that is all they can do.

According to the security point of view, we should concentrate on following points which are also known as *security principles*.

1.4.1 We should not talk with the one which we don't know

According to the context of security, this means, we must be 100% certain about the identity of a device or person before we communicate. Security gurus' point out that it is impossible to be 100% certain of anything, but it is the job of security designers to bring as close to 100% as possible.

For a Wi-Fi LAN, it is not enough to verify the identity of the third party. It also has to verify that every message really came from that party. A simple method for this is the use of secret key. This key can be used to establish the identity and same key can also be used while sending each message to ensure message's authentication. The main idea behinds this technique is that, even if enemies knows the valid network address and other information, they cannot send the message because they don't know the secret key, which must be incorporated into every message.

1.4.2 Accept nothing without a guarantee

We know that the sender must prove his identity before we accept his message. But we also need to be sure that the message we have received from the sender has not been

modified, delayed or even replaced with new message.

1.4.3 Treat everyone as an enemy until proved otherwise

We should not give information to anyone until that person has proved the identity. In Wi-Fi LAN, AP advertises itself by transmitting a beacon frame which consists of its identity and other information. So, an enemy may receive this frame and start connecting into the network.

1.4.4 We should not trust our friends for a long time

An enemy may have access the network by some means and also know the secrete key to process the message. So in this case enemy will be treated as friend. In order to avoid this sort of problem we should keep changing secrete key after certain interval of time. Thus, if an enemy knows a secrete key, he will be able to communicate only for short period of time but in the case of friends, they knows when the secrete key is changed and what the secrete key is.

1.4.5 Use well-tried solutions

The security that we are going to used should be well tried one i.e. it should be recognized by security gurus as the trusted one.

If we have developed a new security methods, that should be publicly available, in order to be well trusted. Also our system should be able to gain the trust of security gurus. After gaining the trust, they study on the system and try to break it. But if our system is really strong and can survive for few years, it can be used in any security system.

1.4.6 We should watch the ground on which we are standing for the cracks

We should constantly monitor the system we are using. We can not say that a security system we are using is completely secure. It may be more secure compared to the other security system but there may be some hole to enter for the enemy. It may be too difficult to enter through the hole but if unlimited time and processing power be given, every system is said to be breakable except the Vernam cipher for encryption, which uses pure random data.

1.5 Outline Of The Thesis

Chapter 2 presents the discussion of the wireless protocols in detail. This chapter mainly focus on the latest security protocol namely *WPA 2*, also known as IEEE 802.11i. Besides the discussion of IEEE 802.11i, it also presents the older wireless security protocols; *WEP* and *WPA*.

Chapter 3 presents the security analysis of the wireless protocols. This chapter begins by giving the model for analyzing security protocol which is known as *CIA Model*. It then discuss about the threat model, which is most important while analyzing the security protocols. This chapter also discusses about the known issues of IEEE 802.11i and finally it presents the logics for proving security properties of the network protocol, known as *Protocol Composition Logics (PCL)*.

Chapter 4 presents the DoS attacks on IEEE 802.11i. This chapter mainly discusses about the two important DoS attacks on IEEE 802.11i; DoS attack on 4-Way Handshake and RSN IE Poisoning, and also provides their possible solutions.

Chapter 5 analyzes the IEEE 802.11i by using *PCL* theoretically and practically by using ns2 and proposes its improvements. The proposed solution was also verified by using *PCL* theoretically and practically by using ns2.

Chapter 6 consists of the conclusion of the study and the future work to be done in the field of wireless security.

CHAPTER 2
WIRELESS PROTOCOLS

This chapter presents the detail about the wireless protocols; WEP, WPA and WPA 2 (also known as IEEE 802.11i). This chapter starts with the basic operations in the wireless LAN followed by the detail study of the wireless security protocols. This chapter mainly focuses on the IEEE 802.11i.

2.1 Basic Operations in Wireless LAN

Before discussing about the protocol detail, we need to discuss about the overview of the operation in wireless communication (i.e. the sequence of events used in wireless communication). Suppose AP is already turned on and operating. The AP advertises its presence by transmitting short wireless messages at a regular interval, usually about 10 times a second. These short messages are called beacons and allow wireless devices to discover the identity of the AP.

Now if we turned on the STN, after completion of all the initialization phase, it starts searching for the AP. It may have been configured to look for a particular AP, or it may be prepared to connect to any AP, regardless of identity. There are a number of different radio frequencies (called channels) that could be used, so, the STA must tune into each channel in turn and listen for beacon messages. This process is called *scanning*. The process can be accelerated by *probing*.

The STA may discover several APs in a large network and must decide to which it intends to connect; often this decision is made based on signal strength. When the STA is ready to connect to the AP, it first sends an authenticate request message to the AP. The original IEEE 802.11 standard defined the authenticate messages as part of the security solution, but they are not used for this purpose in Wi-Fi. Because, in this section, we are

not using security, the AP immediately responds to authenticate request by sending an authenticate response indicating acceptance.

Now that the STA has permission to connect to the AP, it must take one more step before the connection is complete. In IEEE 802.11 the concept of "connection" is called *association*. When a STA is associated with an AP, it is eligible to send data to and receive data from the network. The STA sends an association request message and the AP replies with an association response indicating successful connection. After this point, data sent from the STA to the AP is forwarded onto the wired LAN to which the AP is connected. Similarly, data from the wired LAN intended for delivery to the STA is forwarded by the AP.

In IEEE 802.11, there are three types of messages:

- **Control**

  These are short messages that tell devices when to start and stop transmitting and whether there has been a communication failure.

- **Management**

  These are messages that the STA and AP use to negotiate and control their relationship. For example a STA uses a management message to request access to the AP.

- **Data**

  After STN has been associated with the AP, data are transmitted by using this message.

Control messages doesn't concern with the security. So, here only management messages will be described in brief. The original IEEE standard for wireless network describes seven types of management frames;

- Beacon*
- Probe

- Authenticate
- Associate
- Reassociate
- Disassociate*
- Deauthenticate*

Here * means that message is sent out but the response is not expected.

Fixed Fields	Elem 1	Elem 2	Elem 3	……..	Elem n

Figure 2.1 Management Frame Format

Management frames consists of two parts. The first part is the set of fixed field and the second part consists of elements. An element is a self-contained packet of information that may (or may not) be relevant to the receiving device. Also, there may be a number of elements added to the fixed portion of the management message.

The fixed field contains various items of information specific to particular types of management frames. This includes, for example, flag bits that indicate whether optional features are active. The use of elements is a powerful and flexible idea with several benefits such as easily updating standards, providing manufacture specific information etc.

Each element has a similar structure. The first byte identifies the type of element. The second byte indicates the length i.e. how many bytes are in the element and the information in the bytes that follow.

2.2 Wired Equivalent Protocol (WEP)

WEP is a data encapsulation technique used by IEEE 802.11. The primary goal of WEP is to protect confidentiality of the user data from eavesdropping. It is based on RC4

encryption algorithm, with a secrete key of 40 bits or 104 bits, which is combined with 24 bits Initialization Vector (IV) to encrypt the plain text message, M and its checksum i.e. Integrity Check Value (ICV). Thus, this technique consists of two steps:

- Computation of checksum
- And encryption

First of all the integrity checksum value of a plain text message, M, is computed by using CRC32 algorithm. After computation of the check sum value, it is concatenate with the message, M, to obtain the plain text, P. Thus

$$P = M + ChkS(M)$$

Where ChkS(M) is the checksum value of message, M.

After the computation of the checksum value and the plain text, P, it is input to the encryption algorithm. As mention earlier, WEP uses RC4 encryption technique. The RC4 algorithm generates a keystream (i.e. a long sequence of pseudorandom bytes) as the function of IV, v, and secret key, k. This generated keystream is denoted by RC4(v,k). Finally, the cipher text is obtained by performing exclusive OR operation between plaintext and the generated keystream. Thus, if c is the cipher text then, it can be computed as follows:

$$c = P \text{ XOR } RC4(v,k)$$

where, XOR is an exclusive OR operator

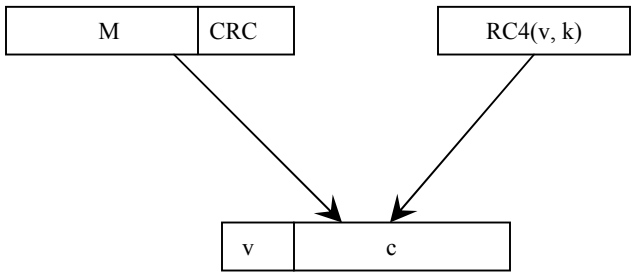


Figure 2.2 WEP Encryption Techniques

The computed cipher text and the IV are then transmitted over the radio wave. In the recipient the encrypted WEP frame is decrypted. The decryption process is just the reverse of the encryption process. So, first the keystream is regenerated by using v and k i.e. RC4(v,k) and then perform the exclusive OR operation with the transmitted cipher text to obtain the plain text.

If c is the cipher text then,

c XOR RC4(v,k)
→(P XOR RC4(v,k)) XOR RC4(v,k)
→P , which is the transmitted plain text

After obtaining the plain text, P, the recipient verifies the checksum in the decrypted plain text. For this first P is splited into M' and c', recomputed the checksum value, ChkS(M') and checking that it matches with the received checksum value.

- P → (M',c')
- if ChkS(M') = c' then
    - valid message and accept it
- else
    - invalid message and reject it

Thus, this verification ensures that only frames with the valid checksum will be accepted by the receiver.

The above description shows that the IV is the key to WEP security. So, to maintain decent level of security, the IV should be incremented for each packet so that subsequent packets are encrypted with the different keys. But for WEP security, the IV is transmitted in the plain text and the 802.11 standard doesn't mandate IV incrementation, which leaves the security measures at the option of the particular wireless terminal (access point or wireless card) implementation.

2.2.1 WEP Goals

WEP is intended to enforce three main security goals:

- Confidentiality

  Prevent from eavesdropping

- Access Control

  Protect access to the wireless network infrastructure

- Data Integrity

  Prevent tempering with the transmitted message

All the above goals depend on the difficulties of discovering the secret key through the *brute-force attack*.

2.2.2 Drawbacks of WEP

Lots of weaknesses have been found by lots of security experts regarding the security provided by WEP. Nikita Borisov, Ian Goldberg and David Wagner [6] claim that all the goals proposed by the WEP protocol was not fulfill i.e. all of the claimed goals can easily be broken. They have shown various drawbacks of the WEP protocol such as keystream reuse, the use of linear function of message by the checksum use of unkeyed function of message by the WEP checksum, authentication spoofing etc. Due to keystream reuse number of attacks can be occurred such as if the plain text of one of the message is known then the plain text of other can easily be obtained. The WEP protocol uses linear function of messages for its checksum, so, the message can easily be modified. Also due to use of the unkeyed function of the message in the checksum, the WEP does not provide secure access control (i.e. message injection).

According to Guillaume Lehembre [7], the WEP protocol was not created by the expert of security or cryptography. Fluher, Mantin and Shamir in [8] show two vulnerabilities in the RC4 encryption algorithm: “Invariance Weaknesses” and “known IV attacks”. Both attacks are based on the fact that for certain key values it is possible for bits in the initial

bytes of the keystream depend on just the few bits of the encryption key. Also, the standards committee for 802.11 left many of the difficult security issues such as key management and a robust authentication mechanism as open problems [9]. Thus, the deployment of a wireless network opens a back door into the internal network that permits an attacker access beyond the physical security perimeter of the organization.

As mentioned earlier, the integrity checksum used in the WEP protocol is the CRC32 algorithm, which also suffers from the serious problems. CRC32 is commonly used for the error detection but was never considered as the cryptographically secure due to its linearity.

2.2.3 WEP Hacking Tools

There are also many tools available which exploit the vulnerabilities of the WEP. Among them one of the most popular one is AirSnort. This tool allows WEP keys to be recovered by analyzing a sufficient amount of traffic. This type of attack can be conducted successfully on a busy network with the reasonable timeframe. Other popular tools are:

- FMS attack and its optimized version by h1kari
- Aircrack (WEP key cracker making use of collected unique IVs, created by French Security researcher Chrisristophe Devine)
- WepLab
- Airodump (wireless sniffing tool used to discover WEP enabled networks)
- Aireplay (injection tool to increase traffic)

2.2.4 Summary on WEP Security

The insecurity of WEP is due to following:

- RC4 algorithm weaknesses within the WEP protocol due to key construction.
- IVs are too short (24 bits) which requires less than 5000 packets for a chance of collision.

- Allowing IV reuse due to which there is no protection against message replay.
- No proper integrity check (CRC32 is used for error detection which is not considered as cryptographically secure due to its linearity).
- No built-in method for updating keys.

Table 2.1 shows the timeline of the death of WEP.

Date	Description
September 1995	Potential RC4 vulnerability (Wagner)
October 2000	First publication on WEP weakness : Unsafe at any key size; An analysis of the WEP encapsulation (Wagner)
May 2001	An inductive chosen plain text attack against WEP/WEP2 (Arbaugh)
July 2001	CRC bit flipping attack – Intercepting Mobile Communications: The insecurity of 802.11 (Borisov, Goldberg, Wagner)
August 2001	FMS attacks – Weaknesses in the key scheduling algorithm of RC4 (Fluhrer, Mantin, Shamir)
August 2001	Release of AirSnort
February 2002	Optimized FMS attack by h1kari
August 2004	KoreK attack (unique IVs) – release of chopchop and chopper
July/August 2004	Release of Aircrack (Devine) and WepLab (Sanchez) implementing KoreK attacks.

Table 2.1 Timeline of WEP Death [7]

2.3 Wi-Fi Protected Access (WPA)

WPA [1] is an intermediate standard framework developed in order to have a more secure WLAN solution until the IEEE 802.11i standard was finished. WPA is based on parts from the early IEEE 802.11i drafts and is considered secure if property used.

WPA solves two major problems associated with the earlier WEP security mechanism:

- It uses encryption technique for authentication, which helps in preventing unauthorized clients from becoming part of the wireless network.
- It uses the constantly changing key instead of the single shared key used for encryption by WEP.

WPA uses 802.1X and *Extensible Authentication Protocol* (EAP) [10] for authentication and a protocol called *Temporal Key Integrity Protocol* (TKIP) for traffic encryption. WPA also includes an integrity checksum based on the network packet that can detect whether a packet is originating from the valid network user or an intruder who is attempting to crack the key used by network.

2.3.1 Types of WPA

There are of two types of WPA:

- WPA Personal
- WPA Enterprise

WPA Personal

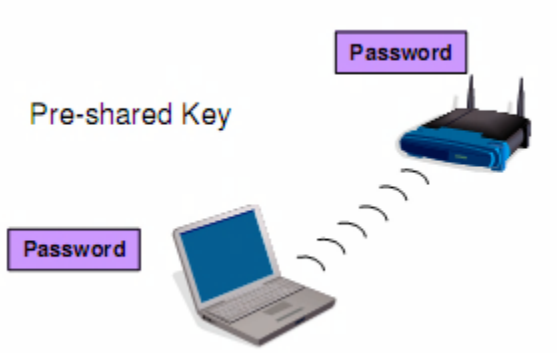


Figure 2.3 WPA Personal

WPA-Personal is generally intended for the home user and the small organizations, which doesn't consist of any authentication server. In this type of network, *Per-Shared Key* (PSK) is used as an encryption key. The PSK must be provided by the network client

before it can log into a WPA based network. The original PSK is encrypted using a process known as TKIP, which changes the key repeatedly during a connection to keep the connection secure. Hence, WPA-Personal is also known as WPA-PSK.

WPA Enterprise

Figure 2.4 WPA Enterprise

WPA-Enterprise is generally used in the large organization where a more secure means of communication are required. It is more powerful version of WPA and uses the separate server for the authentication. *Remote Authentication Dial-In User Service* (RADIUS) [11] or *Authentication, Authorization and Accounting* (AAA) [12] server is generally used to authenticate individual user and while authentication it uses the protocol namely IEEE 802.1X [13]. WPA also supports the *Extensible Authentication Protocol* (EAP).

2.4 Wi-Fi Protected Access 2 (WPA2)

In order to improve the security issues in wireless networks and to improve the security issues in WEP protocol, IEEE launched a new standard in July 2004, which is known as IEEE 802.11i [2]. It was designed to provide more secure alternative to WEP and WPA while still retaining backward compatibility to WPA devices. It is also called as WPA 2 which is what *Wi-Fi Alliance* calls their approved interoperable implementation of 802.11i.

The IEEE 802.11i architecture consists of following components:

- IEEE 802.1X for authentication
- RSN for keeping track of associations
- And CCMP for providing confidentiality

2.4.1 IEEE 802.11i Authentication

In order to provide better authentication and confidentiality in IEEE 802.11 networks than WEP, the standard defines a *Robust Security Network Association* (RSNA) based on IEEE 802.1X [14] authentication. The authentication process involves three entities:

- Supplicant
- Authenticator
- Authentication Server

A successful authentication means that the supplicant and authenticator verify each other's identity and generate some shared secret for subsequent secure data transmissions. The authentication server can be implemented either in a single device with the authenticator or through separate server, assuming the link between the authenticator and authentication server are physically secure.

The complete process of authentication of IEEE 802.11i authentication consists of handshake between:

- Supplicant and authenticator (using 802.1X [14] protocol)
- Authenticator and authentication server (de facto RADIUS [3, 15])
- And between supplicant and authentication server (de facto EAP-TLS [16])

After these handshakes, the supplicant and the authentication server have authenticated each other and generate a common secret which is known as *Master Session Key* (MSK). The supplicant uses MSK to generate *Pairwise Master Key* (PMK). Authenticator also generates PMK after receiving AAA key material from the server. The supplicant and authenticator may be configured using static *Pre-Shared Key* (PSK) for the PMK. The PSK is generated from the passphrase (from 8 to 63 characters) or 256 bits string and

provide a solution for home networks and small enterprises that have no authentication server.

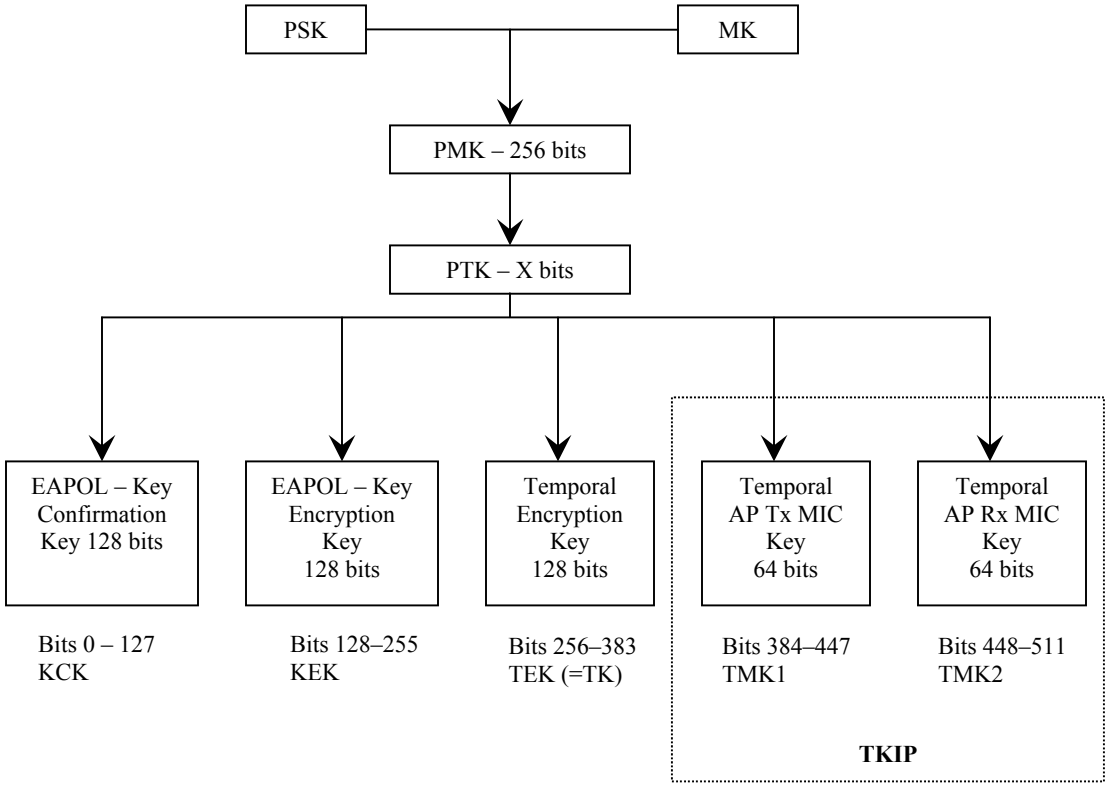


Figure 2.5 Pairwise Key Hierarchy

The PMK itself is never be used for encryption or integrity checking but it is used to generate a temporary encryption key; for unicast traffic this is the *Pairwise Transient Key* (PTK). The length of the PTK depends on encryption protocol i.e. 512 bits for TKIP and 384 bits for CCMP. The PTK further consists of following keys:

- **KCK (*Key Confirmation Key* – 128 bits):** This key is used for authenticating messages (MIC) during 4-Way handshake and Group Key Handshake.
- **KEK (*Key Encryption Key* – 128 bits):** This key is used for ensuring data confidentiality during the 4-Way Handshake and Group Key Handshake.
- **TK (*Temporary Key* – 128 bits):** This key is used for data encryption which is used by TKIP and CCMP.

- **TMK (*Temporary MIC Key* – 2*64 bits):** This key is used for data authentication which is used by Michael with TKIP. A dedicated key is used for each side of the communication.

Also during re-association, a cached PMK can be used directly in order to reduce the computational load on the authentication server during repeated authentication requests from the same server.

Regardless of whether the PMK is derived from the EAP/802.1X/RADIUS handshake, or based on PSK, or reused from a cached PMK, a 4-Way Handshake must execute for successful RSNA establishment.

2.4.2 4-Way Handshake

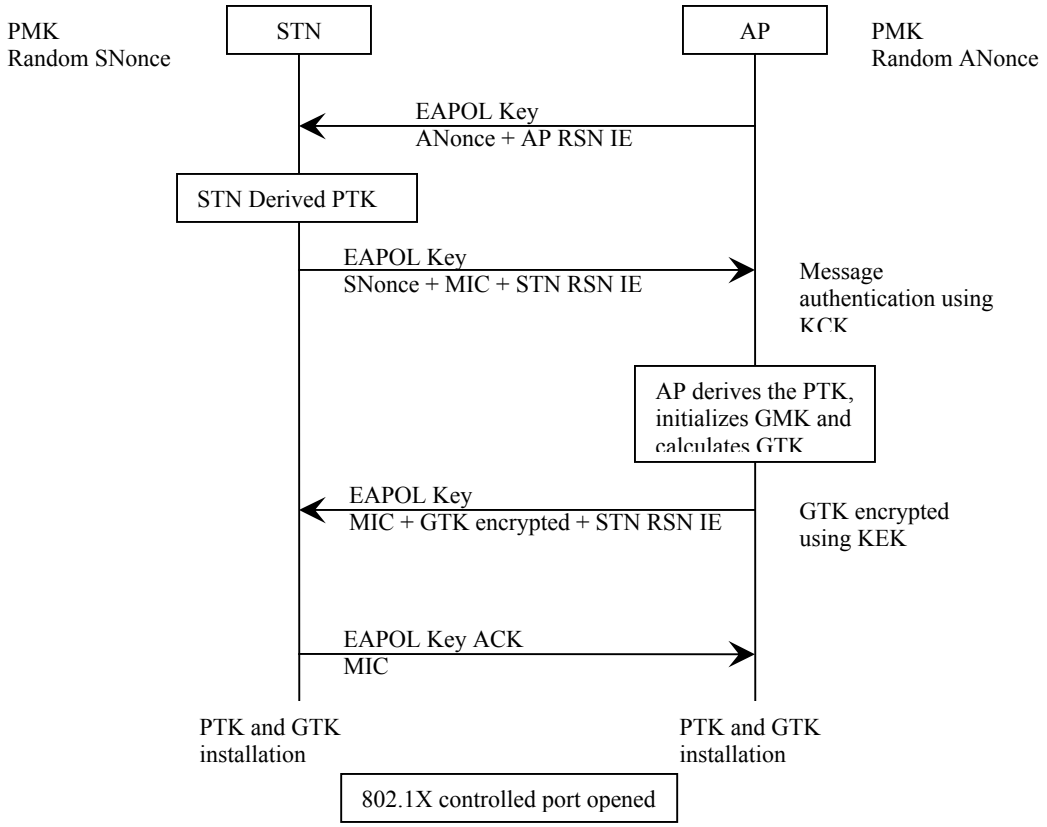


Figure 2.6 4-Way Handshake

In 4-Way Handshake four EAPOL-key messages are ex-changed between the supplicant and the access point. It is initiated by access point.

The PTK is derived from PMK, a fixed string, the MAC address of client and two random numbers (ANonce and SNonce, generated by the authenticator and supplicant respectively). The access point initiates the first message by selecting the random number ANonce and sending it to the supplicant, without encrypting the message. The supplicant generates its own random number called SNonce and can now calculate the PTK and derived temporary keys. So, it sends SNonce and the MIC key which is calculated by using KCK key. When the authenticator receives the second message, it can extract SNonce (because the message is not encrypted) and calculate the PTK and derived temporary keys. Now it can verify the value of MIC in the second message and thus be sure that the supplicant knows the PMK and has correctly calculated the PTK and derived temporary keys.

The third message send by the authenticator to the supplicant contains the GTK (encrypted with the KEK key), which is derived from the random GMK and GNonce, along with an MIC calculated from the third message using KCK key. When the supplicant receives this message, the MIC is checked to ensure that the authenticator knows the PMK and has correctly calculated the PTK and derived temporary keys.

The last message acknowledges completion of the whole handshake and indicates that the supplicant will now install the key and start encryption. Upon receiving the last message, the authenticator installs its key after verifying the MIC value. So, the STN and the AP have obtained, computed and installed encryption keys and are now able to communicate over the secure channel for unicast and multicast traffic.

Multicast traffic is protected by another key which is known as GTK (*Group Transient Key*). GTK is generated from the master key called GMK (*Group Master Key*), a fixed string, the MAC address of the access point and a random number called GNonce. The

length of GTK depends on the encryption protocol i.e. 256 bits for TKIP and 128 bits for CCMP. GTK is divided into following temporary keys:

- GEK (*Group Encryption Key*): This key is used for data encryption.
- GIK (*Group Integrity Key*): This key is used for data authentication which is used only by *Michael* with TKIP).

2.4.3 Group Key Handshake

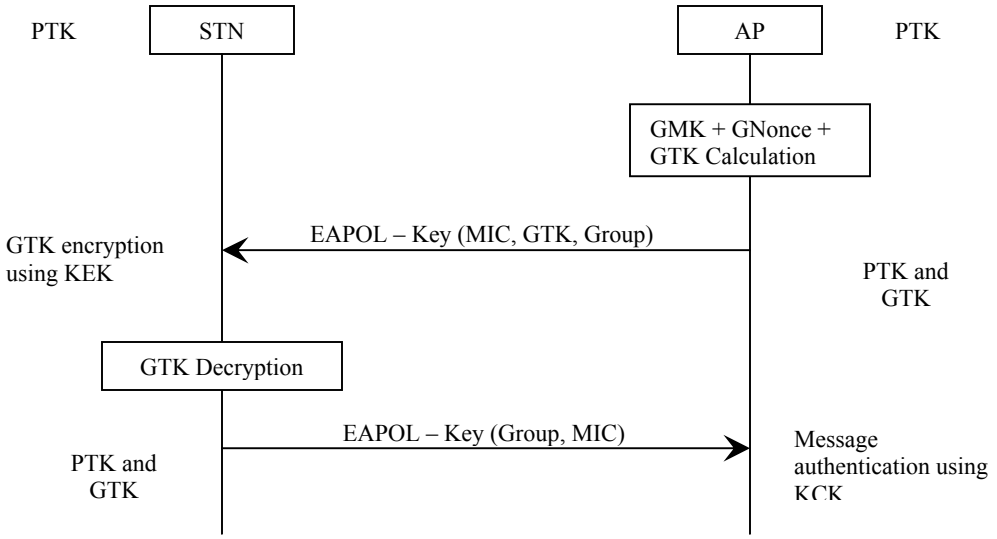


Figure 2.7 Group Key Handshake

During group key handshake, two EAPOL-key messages are exchanged between the client and the access point. This handshake makes use of temporary keys generated during 4-Way Handshake.

The group key handshake is only used to disassociate a host and to renew the GTK at a client's request. The authenticator initiates the first message by choosing the random number called GNonce and calculate the new GTK. The GTK is encrypted by using KEK. It then sends the encrypted GTK, the GTK sequence number and the MIC calculated form this message using KCK to the supplicant. When the message is received by the supplicant, the MIC is verified and the GTK can be decrypted.

The second message acknowledges the completion of the Group Key Handshake by sending the GTK sequence number and the MIC calculated on this second message. Upon receipt, the authenticator first verifies the MIC and then installs the new GTK.

2.4.4 802.1X Authentication

The IEEE 802.1X [14] authentication protocol which is also known as *Port-Based Network Access Control*, is a framework originally developed for the wired networks for providing authentication, authorization and key distribution mechanism, implementing access control for user joining the network.

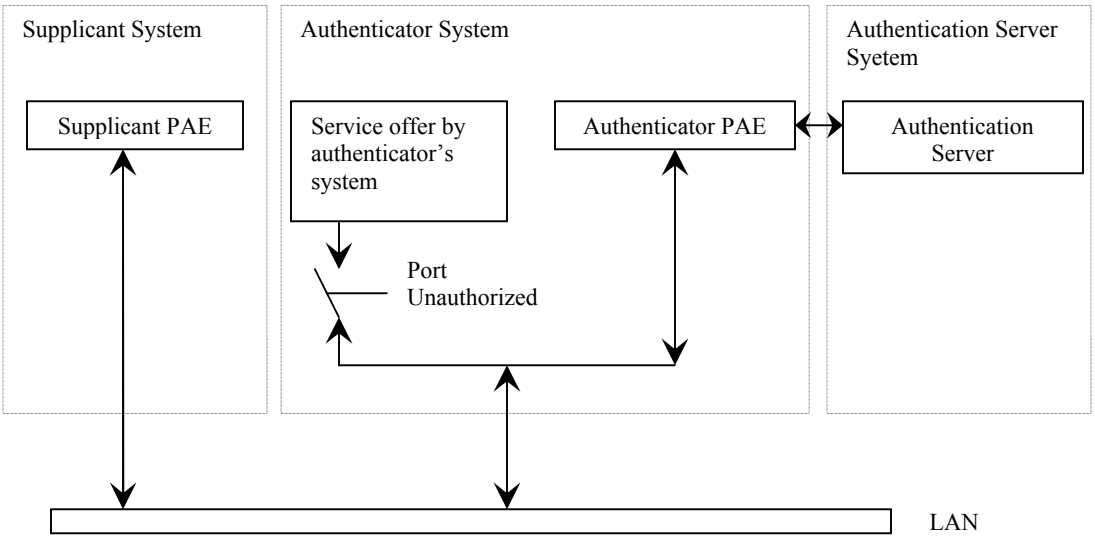


Figure 2.8 IEEE 802.1X Model From IEEE 802.1X Specification

The architecture of IEEE 802.1X also made up of three functional entities:

- The supplicant joining the network.
- The authenticator providing access control.
- The authentication server making the authentication decisions.

For IEEE 802.11i, the access point takes the role of authenticator and the client card, the role of supplicant. But in the system using IBSS, the client card takes the role of both the supplicant and the authenticator. In 802.1X, the authenticator enforces authentication.

The authenticator doesn't need to do authentication but the authenticator exchanges the authentication traffic between the supplicant and the authentication server.

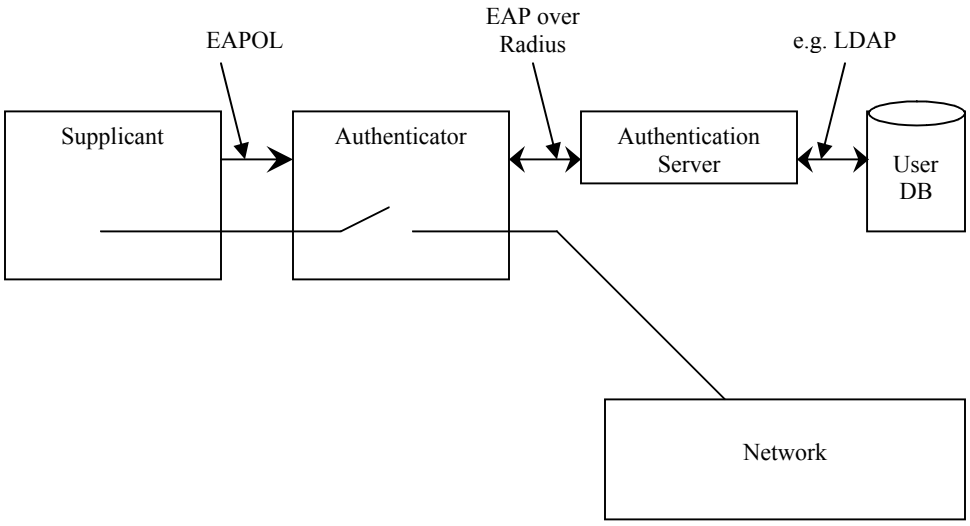


Figure 2.9 Before Authentication

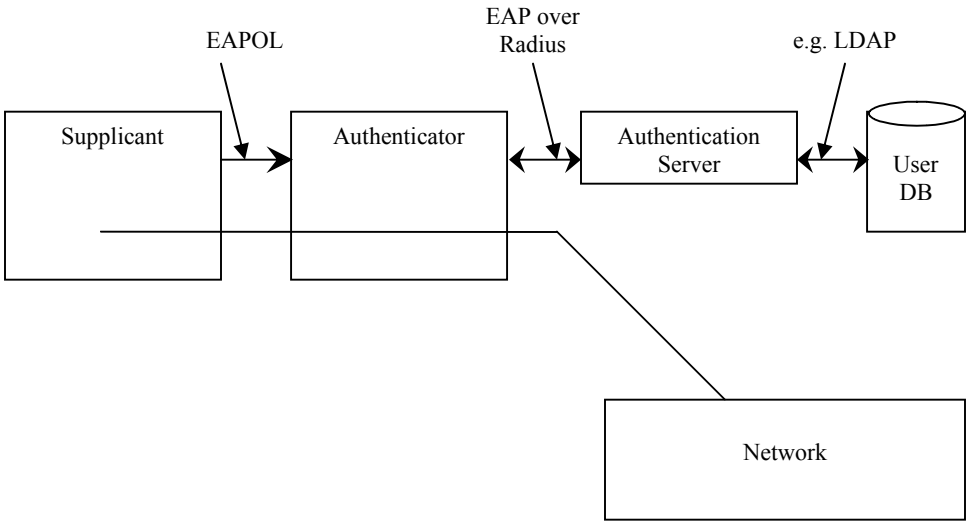


Figure 2.10 After Authentication

In 802.1X, each physical port[1] is divided into two logical ports, making up the PAE (*Port Access Entity*).

[1] Virtual port in wireless network

- Authentication PAE
- Service PAE

The authentication PAE is always open and allows authentication frames to pass through but the service PAE is only opened upon successful authentication (i.e. in an authorized state) for a limited time (3600 seconds by default). The decision to allow access is usually made by authentication server which may be either a dedicated RADIUS or in home networks, a simple process running on the access point.

The IEEE 802.11i standard makes small modifications to IEEE 802.1X for wireless networks to account for the possibility of identify stealing i.e. message authentication has been incorporated to ensure that both the supplicant and the authenticator calculate their secret keys and enable encryption before accessing the network.

The supplicant and the authenticator communicate by using EAP [10, 17] based protocol. The role of authenticator is essentially passive i.e. it may simply forward all messages to the authentication server.

EAP is a framework for the transport of various authentication methods which allow only a limited number of messages to flow i.e. Request, Response, Success and Failure. Other intermediated messages are dependent on the selected authentication method e.g. EAP-TLS, EAP-TTLS, PEAP, Kerberos, V5, EAP-SIM etc. When all the process of authentication is completed both, the supplicant and the authenticator have a secret master key. Communication between the authenticator and the authentication server is done by using EAPOL (EAP over LAN). EAPOL is used in the wireless networks to transport EAP data using higher layer protocol such as RADIUS.

2.4.5 RADIUS Protocol

Remote Authentication Dial-In User Service (RADIUS) is an industry standard protocol described [16] and [18]. RADIUS is used to provide authentication, authorization, and accounting services. A RADIUS client (typically a dial-up server, VPN server, or

wireless access point) sends user credentials and connection parameter information in the form of a RADIUS message to a RADIUS server. The RADIUS server authenticates and authorizes the RADIUS client request, and sends back a RADIUS response message. RADIUS clients also send RADIUS accounting messages to RADIUS servers. Additionally, the RADIUS standards also support the use of RADIUS proxies. A RADIUS proxy is a computer that forwards RADIUS messages between RADIUS-enabled computers.

RADIUS messages are sent as *User Datagram Protocol* (UDP) messages. UDP port 1812 is used for RADIUS authentication messages and UDP port 1813 is used for RADIUS accounting messages. Some network access servers might use UDP port 1645 for RADIUS authentication messages and UDP port 1646 for RADIUS accounting messages. By default, IAS supports receiving RADIUS messages destined to both sets of UDP ports. (For information about changing the UDP ports that are used by IAS, see Configure IAS port information.) Only one RADIUS message is included in the UDP payload of a RADIUS packet.

According to [16] and [18] RADIUS messages are of following types:

- **Access-Request**

  This message is sent by a RADIUS client to request authentication and authorization for connection attempt.

- **Access-Accept**

  This message is sent by a RADIUS server in response to an Access-Request message. This message informs the RADIUS client that the connection attempt is authenticated and authorized.

- **Access-Reject**

  This message is sent by a RADIUS server in response to an Access-Request message. This message informs the RADIUS client that the connection attempt

is rejected. A RADIUS server sends this message if either the credentials are nto authentic or the connection attempt is not authorized.

- **Access-Challenge**
  This message is sent by a RADIUS server in response to an Access-Request message. This message is a challenge to the RADIUS client that requires a response.

- **Accounting-Request**
  This message is sent by a RADIUS client to specify accounting information for a connection that accepted.

- **Accounting-Response**
  This message is sent by the RADIUS server in response to the Accounting-Request message. This message acknowledges the successful receipt and processing of the Accounting-Request message.

A RADIUS message consists of a RADIUS header and zero or more RADIUS attributes. Each RADIUS attribute specifies a piece of information about the connection attempt. For example, there are RADIUS attributes for the user name, user password, type of service requested by the user, and the IP address of the access server. RADIUS attributes are used to convey information between RADIUS clients, RADIUS proxies, and RADIUS servers, e.g. the list of attributes in the Access-Request message includes information about the user credentials and the parameters of the connection attempt. In contrast, the Access-Accept message includes information about the type of connection that can be made, connection constraints, and any vendor-specific attributes (VSAs).

2.4.6 RSNA Data Confidentiality And Integrity

[6, 9] shows that IEEE 802.11 entity authentication (*Open System Authentication* and *Shared Key Authentication*) are completely insecure. So, IEEE 802.11i defines the notion

of a *Robust Security Network* (RSN).

RSN is the network that only allows *Robust Security Network Association* (RSNA). Two devices can establish a RSNA if they use the 4-Way Handshake to authenticate the association. Robust security is however, is not achieved unless all the devices within the network use RSNAs. An ESS advertises its RSN capabilities via the RSN-IE (*RSN Information Element*). Older security solution such as WPA and WEP are collected under *Transition Security Network* (TSN).

802.11i RSNA establishment procedure consists of 802.1X authentication and key management protocols. Generally, a successful authentication means that the supplicant and authenticator verifies each other's identity and generate some shared secret for subsequent key derivations. Based on this shared secret, the key management protocols compute and distribute usable keys for data communication sessions. RSNA supports following protocols for data confidentiality and integrity:

2.4.6.1 Temporal Key Integrity Protocol (TKIP)

TKIP is the protocol which is used for the traffic encryption in the WPA. It is designed as a software patch to upgrade WEP in already deployed equipment. Thus, we can say that TKIP is just a WEP in a suit of armor that takes care of the current problems with WEP. So, no separate hardware is required because it uses RC4 not the AES (*Advance Encryption Standard*), which is used in CCMP. TKIP is required for WPA certification and is also included as the part of RSN 802.11i as an option.

TKIP is designed to address all the known weakness of the WEP:

- Prevent frame forgeries
- Prevent replay
- Correct WEP's misuse of the encryption
- Never reuse keys

In TKIP, a new MIC, developed by the cryptographer Niels Ferguson, called Michael was used. Michael does not consist of multiplication. It just consists of shift and adds operations, so, it can be implemented in the software running on slow microprocessors. It also fits in all the existing APs. Michael is equivalent to 20-bits of security. So, the brute force attack is still exists in Michael. Due to this defect, TKIP also implement countermeasures. The counter measure bound the probability of the successful forgery and amount of information an attacker can learn about a key.

Replay Protection

TKIP uses a per-packet sequence counter called TSC (*TKIP Sequence Counter*) for the replay protection. If the receiver receives a packet out of order, it will drop that packet. Thus TKIP simply ignore those messages that has TSC less than or equal to the last message. Sequence number for different MPDUs (*MAC Protocol Data Unit*) of the same MSDU (*MAC Service Data Unit*) must be sequential, otherwise the fragmentation attack is said to be enabled. MSDU and MPDU both refer to a single packet of data, but MSDU represents data before fragmentation, while MPDUs are the multiple data units after fragmentation. MIC in TKIP is calculated from the MSDU but in CCMP it is calculated from the MPDU. In order to protect from replay, TKIP works with the 802.11e (QoS). QoS intentionally reorder the packets while transmitting.

Key Mixing

In order to overcome the weakness in IV of the WEP, following improvement has been done in the IV of TKIP:

- Size is increased from 24 bits to 48 bits.
- IV is used as the sequence counter to avoid the replay attacks.
- IV is constructed in such a way that it avoids certain kinds of weak key attack.

If IV is incremented by 1 each time then for 24 bits, an IV will be reuse after 2^{24} packets. Thus, for a device sending 10000 packets per second, 24 bits IV takes half an hour to

rollover but 48 bits of IV takes 900 years.

In WEP, per packet key is 24 bits + 104 bits, i.e. 128 bits which can easily be handled. But in the case of TKIP, per packet key is 48 bits + 104 bits, i.e. 152 bits. So, we need special attention in order to handle this sort of odd circumstances, which can be done by using "key mixing".

The key mixing consists of two phases. Both of these phases uses a cryptographic mixing function called S-Box, consisting of 512 word entries, which involve only shift, add and XOR operations.

- **Phase I**

  This phase takes following inputs and produces 80 bits output values:
  - 128 bits session keys
  - The upper 32 bits of the IV
  - And the MAC address
- **Phase II**

  This phase takes following inputs and produces 128 bits encryption key:
  - Output of phase I
  - The lower 16 bits of the IV
  - And 128 bits session keys

The output of phase II and part of the extended IV plus a dummy byte are the input for RC4 which generates a keystream that is XOR-ed with the plaintext MPDU, the MIC calculated form the MPDU and the old ICV from WEP.

2.4.6.2 Counter-Mode/Cipher Block Chaining Message Authentication Code Protocol (CCMP)

CCMP is the encryption protocol in IEEE 802.11i. It defines the set of rules that uses the *Advance Encryption Standard* (AES) block cipher for encryption and integrity protection. Thus, AES is not a security protocol; it is a block cipher. In RSN the security protocol

built around AES is called *Counter Mode–CBC MAC Protocol*, or CCMP. CCMP was developed from scratch, so, it was supposed to solve all the known weakness in wireless network, where as TKIP compromises certain issues such as it uses weaker security primitives i.e. Michael for calculating MIC in order to accommodating the existing hardware.

CCMP uses two modes of operations; Counter Mode and Counter Mode with CBC-MAC (CCM).

Counter Mode

Counter mode does not use the AES block cipher directly to encrypt the data. But it encrypts the value of an arbitrary value called the counter and then XORs the result with the data to produce ciphertext. The counter is generally incremented by 1 for each successive block processed; that's why it was named as counter mode.

The counter might start at any arbitrary value and might increment by some other value or pattern. The important thing is that the receiving party who wants to decrypt the message must know the starting value of the counter and the rules for advancing it. So, even if two blocks of data were identical, they would be combined with a different counter value to produce different ciphertext.

Decryption is exactly the same process as encryption because XORing the same value twice takes back to the original value. The other useful feature, for some applications, is that the encryption can be done completely in parallel. Because all the counter values are known at advanced, we could have a bank of AES encryption devices and encrypt an entire message in a single parallel operation. Another useful property is that there is no problem if the message doesn't break into an exact number of blocks. We simply take the last (short) block and XOR it with the encrypted counter output using only the number of bits we need. Therefore, the length of the ciphertext can be exactly the same as the length

of the input message. Because each block operation depends on the state of the counter from the previous block, counter mode is essentially stream cipher.

CCM (Counter Mode + CBC – MAC)

CCM mode was created especially for use in IEEE 802.11i RSN, but it is applicable to other system as well and has been submitted to NIST as a general mode for use with AES. It has also been submitted to the IETF for use in IP security. CCM was invented by three of the cryptographers participating in the IEEE 802.11i standards group: Doug Whiting, Russ Housley, and Niels Ferguson. It builds on top of counter mode.

CCM uses counter mode in conjunction with a message authentication method called cipher block chaining (CBC). CBC is used to produce a message integrity code (MIC). The MIC is called a message authentication code by the cryptographic community, leading to the name CBC-MAC.

CBC-MAC can be computed as follows:

- Take the first block in the message and encrypt it using AES (or any block cipher).
- XOR the result with the second block and then encrypt the result.
- XOR the result with the next block and encrypt that…and so on.

The result is a single block (128 bits in our case) that combines all the data in the message. CBC-MAC is simple but cannot be parallelized; the encryption operations must be done sequentially. Furthermore, it should be noted that, by itself, CBC-MAC can only be used on messages that are an exact number of blocks.

CCMP encrypts data at the MPDU level. MPDU corresponds to the frames that actually get transmitted over the radio link. There is one MPDU for each frame transmitted, and the MPDU itself might be the result of fragmenting larger packets passed from a higher layer, called MSDUs.

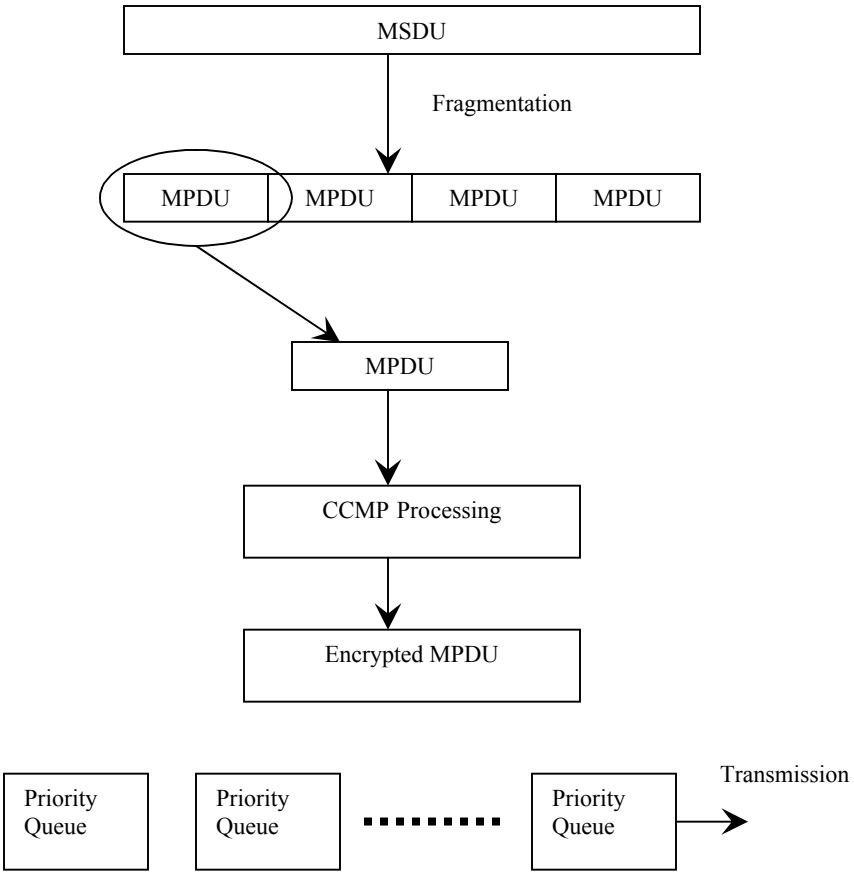


Figure 2.11 Flow of Frames through CCMP

The data arrives as an MSDU and are broken down into fragments. Each fragment is formed into an MPDU and assigned its own IEEE 802.11 header containing source, destination addresses and other information. At this point, each MPDU is processed by the CCMP algorithm to generate a new encrypted MPDU. Only the data part is encrypted, not the header. However, CCMP does more than just encrypt portions of the MPDU. It also inserts extra fields, causing the resulting encrypted MPDU to be 16 bytes longer than the original.

CCMP Processing

Following are the steps involved in CCMP processing:

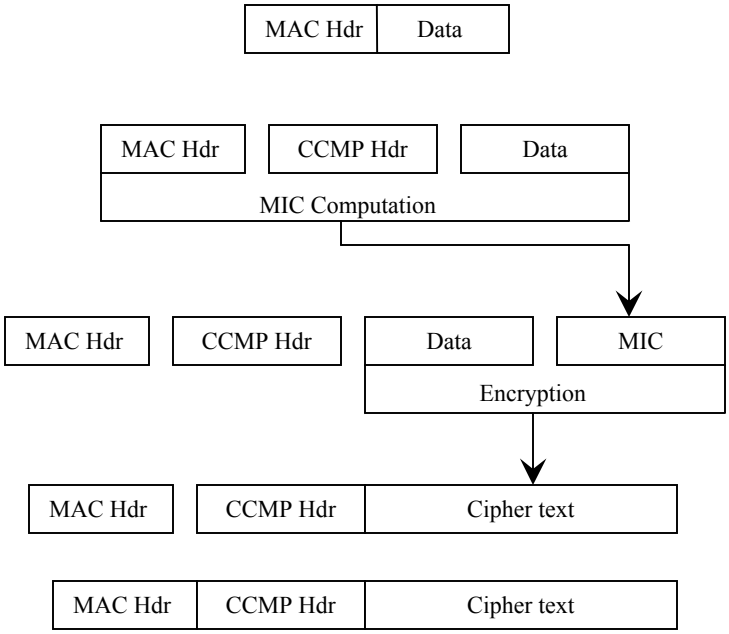


Figure 2.12 Processing of CCMP

- The processing starts with the unencrypted MPDU which consists of IEEE 802.11 MAC header. The header consists of source address, destination address and other fields.
- The MAC header is separated from the MPDU and put aside. Information from the header is extracted and used for creating the 8 byte MIC value. At this stage the 8 byte CCMP header is constructed for later inclusion into the MPDU.
- The MIC value is now computed so as to protect the CCMP header, the data, and parts of the IEEE 802.11 header. Liveness is ensured by the inclusion of a nonce value. The MIC is appended to the data.
- The combination of data and MIC is encrypted. After encryption the CCMP header is prepended.
- Finally the MAC header is restored onto the front of the new MPDU and the MPDU is ready to the queued for transmission. The transmission logic need not have to know about the CCMP header. From here until transmission, only the MAC header will be updated.

The encrypted MPDUs are placed on a queue prior to transmission. There might be several queues waiting their turn based on some priority policy. This allows for later extension to accommodate different traffic classes under IEEE 802.11e. Immediately prior to transmission, some of the fields of the IEEE 802.11 header are updated to meet transmission rules. Those fields that are subject to such changes are called mutable fields and are excluded from the MIC computation.

CCMP Header

The CCMP header must be prepended to the encrypted data and tramsmitted in the plaintext. The CCMP header has two purposes. First, it provides the 48 bit packet number (PN) that provides replay protection and enables the receiver to derive the value of the nonce that was used in the encryption. Second, in the case of multicasts, it tells the receiver which group key has been used.

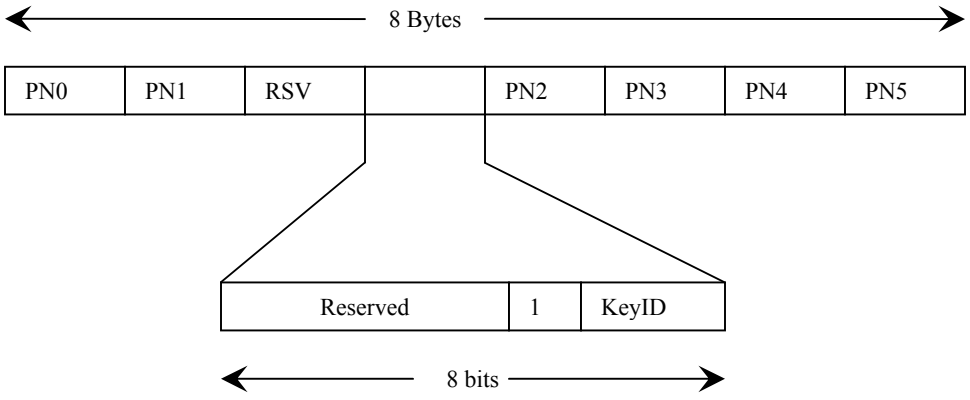


Figure 2.13 CCMP Header

The format of CCMP header is similar to that of the TKIP header. Six bytes are used for the 48-bit PN value, 1 byte is reserved, and the remaining byte contains the KeyID bits. The KeyID is used for specifying which group key has been used. The bit next to the KeyID bits is set to 1, corresponding to the Extended IV bit in TKIP. This value indicates that the frame format is RSN rather than the earlier WEP format.

CCMP Encryption

The implementation of CCMP must keep the sequence counter called the packet number (PN), which is incremented for each packet processed. This prevents an attacker trying to reuse a packet that has previously been sent. The PN is 48 bits long, which is large enough to ensure that it never overflows. There will never be two packets sent with the same sequence value. If the power is down and restarts, the PN will be reset, but this will be with a different key value and hence does not create a threat.

The encryption process in CCMP occurs in two steps:

- MIC is calculated and appended to the MPDU.
- And the entire MPDU (including MIC) is encrypted to produce the result.

After the MIC has been calculated and appended to the plaintext data, it is ready to encrypt the MPDU. The encryption occurs using counter mode and starting with the data immediately following the CCMP header in the template. The encrypted data replaces the original data for the entire data portion and the MIC value, resulting in a complete encrypted MPDU ready to be queued for transmission.

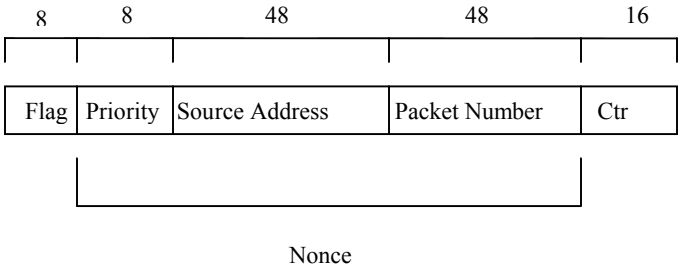


Figure 2.14 Constructing the Counter for CCMP AES Counter Mode

An essential step in counter mode is to initialize the counter in a way that avoids ever using the same start value twice. Therefore the counter is constructed from a nonce in an almost identical way to that for the MIC. The nonce value used is identical to that of the MIC and includes the sequence counter, source MAC address, and priority fields. This

value is then joined with two fields: Flag and Counter ("Ctr"), as shown in following figure.

Once the counter is initialized, encryption can proceed. Each successive value of the counter is encrypted using the secret key and XORed with the template data to produce the encrypted data.

CCMP Decryption

When the encrypted MPDU is delivered to the receiver, the first job is to get the right key for decryption. The correct pairwise keys are selected based on the source MAC address in the MAC header. There are a number of steps that the receiver must take to extract and check the validity of the received data. Decryption is only one step and this process is more generally called decapsulation.

The PN is sent unencrypted in the CCMP header. The first thing the receiver does is to read the PN and compare to the last frame received. If the sequence number is lower or equal to the last one, it should be discarded as a replay of an old message. In this case the receiver goes no further with the MPDU. If the PN matches, the next step is to prepare for decryption using AES/counter mode. This requires the computation of the starting value for the counter, which must match the value used in encryption. All the information is available in the received frame. The sequence number can be combined with the source MAC address and priority to create the nonce. This is then combined with the known flag value and the start ctr value to create the initial counter. Any attacker can compute the same value. However, it is of no use unless the secret key is also known. Decryption proceeds as for encryption. Successive values of the counter are encrypted and XORed with the received MPDU to restore the unencrypted data and the MIC value.

The next stage is to verify that the MIC value is correct. The MIC value is recalculated across the same data as the original MPDU at the sender. The mutable fields in the header are masked out and the computation performed over the whole MPDU, excluding the

MIC. If the data is unaltered from when it was sent, and we have the right secret key, the same result will be obtained. This can be compared to the MIC value sent with the frame. A match means the frame is valid. If it is not matched the frame will be discarded.

Once the MPDU is decoded, the MIC and CCMP header can be removed, and the remaining data is passed up for reassembly with other received fragments to reform the MSDU.

2.5 Summery

Wireless devices (STN) can connect to AP by using two methods; Scanning and Probing. Scanning is the process of choosing the appropriate frequency and listen to the beacon message for connection with the AP and probing is the process by which STN directly sends message to the AP for connection. According to IEEE 802.11, three types of messages are exchanged between STN and AP; Control, Management and Data.

Wireless protocols are of three types; WEP, WPA and WPA2 (IEEE 802.11i). WEP is the data encapsulation technique used in initial IEEE 802.11. The main goal of WEP is to protect data confidentiality from eavesdropping. It uses RC4 as encryption algorithm. The encryption process of WEP consists of two steps; Computation of checksum and Encryption.

Given a message, first checksum is calculated by using CRC32 algorithm, which is combined with message to obtain plaintext. In second step, first of all keystream is generated by using RC4 algorithm by using IV and secret key. The generated keystream is XORed with plaintext to generate cipher text. The cipher text and IV is then transmitted over the radio waves. The decryption process in recipient end is just opposite to the encryption process.

WPA is an intermediate standard developed to secure wireless LAN until IEEE 802.11i standard was completed. It also uses RC4 as encryption algorithm. So, it doesn’t require

any new hardware. Hence we can say that WPA is a software patch to upgrade WEP in already deployed equipment.

WPA uses 802.1X and EAP for authentication and TKIP for data confidentiality. It addresses all the known weakness in WEP but since it was not developed from scratch, there are some compromises in security issues such as it uses weaker algorithm for MIC (Michael) and also it uses weaker encryption algorithm, RC4.

IEEE 802.11i was developed from scratch. So, it addresses all the known weakness in the wireless environment. It is designed as a long term solution to secure wireless network. It uses CCMP for data confidentiality, integrity and replay protection and uses AES as data encryption algorithm. IEEE 802.11i also consists of backward compatibility to older wireless protocols.

IEEE 802.11i architecture consists of three parts:

- 802.1X for authentication
- RSNA for keeping track of association
- CCMP for providing confidentiality

For authentication process IEEE 802.11i uses 802.1X protocol. IEEE 802.1X is port based network access control which is initially designed for wired network. It consists of three entities; Supplication, Authenticator, and Authentication server (generally RADIUS server).

The supplicant and authenticator generate MSK after successful authentication. MSK is used to generate PMK. After generation of PMK, the authenticator and supplicant perform 4-Way handshake. The 4-Way handshake uses PMK to derive PTK, which is used for actual encryption. Thus after successful of 4-Way handshake, both supplicant and authenticator will calculate PTK. After 4-Way handshake group key handshake will be performed, which is use to generate GTK. GTK is used for encryption in group communication. And after group key handshake, actual communication will begin.

CHAPTER 3
SECURITY ANALYSIS

This chapter provides the model for analyzing the security of the protocol followed by the threat model, which is essential to design while analyzing the security protocols. This chapter also provides the known issues of 802.11i and the end this chapter presents the logics for proving security properties of the network protocol, which is known as *Protocol Composition Logics* (PCL).

3.1 CIA Model

The CIA model is the generally accepted model for assessing risks in a system. This model has three components:

- **Confidentiality**
  Information should only be available to those with the proper authorization.
- **Integrity**
  Information should not be altered in any way except by those authorized to do so.
- **Availability**
  Information should be accessible to the authorized users any time that are needed.

The most common way to ensure confidentiality is to encrypt the data but other methods like physically protecting the link over which the information is sent are also valid. But in the wireless, anyone with an antenna can listen to the traffic. So, the only feasible way of ensuring confidentiality is through encryption.

Integrity in the other hand is very hard to ensure. So, the most common solution is to detect it rather than enforce it. Detecting integrity failure is done through *Message Integrity Checksum* (MIC).

Availability is probably the hardest one of all. There is little to do but to have a good redundancy and robust system. E.g. load balancing servers can be used on traffic intense systems to alleviate availability problems.

3.2 Threat Model

In order to understand whether security objectives have been met or not and to evaluate alternative proposals, a threat model is required. As already mentioned, there are three types of frames: Management Frames, Control Frames and Data Frames. Any manipulation of these frames that directly or indirectly affects data confidentiality, integrity, mutual authentication and availability will be considered as threat. Following are the threats in IEEE 802.11i:

3.2.1 Passive Eavesdropping/Traffic Analysis

In wireless communication an adversary can easily sniff and store all the traffic. Even when messages are encrypted, it is important to consider whether an adversary may learn partial or complete information from certain messages. This possibility exists if common message fields are predictable or redundant. Also encrypted messages may be generated upon the request from the adversary itself.

3.2.2 Active Eavesdropping/Message Injection

An adversary may be capable to insert message into the wireless network. *Network Interface Cards* (NICs) may limit the interface for composing packets to the 802.11 standards but an adversary is still able to control any field of the packet using some known techniques. Hence it is reasonable that the adversary can generate any kinds of packets, modify content of the packet and completely control the transmission of the packet. The adversary can also insert a replayed packet, if there is no replay protection.

3.2.3 Message Deletion and Interception

It should be assumed that an adversary is able to do message deletion i.e. an adversary is capable of removing packet from the network before it reaches its destination. This could be done by interfering the packet e.g. by causing CRC errors so that the receiver drops the packet.

Message interception means that an adversary can completely control the connection i.e. the adversary can capture the packet before the receiver actually receives it and decides whether to delete the packet or forward it to the receiver. It differs from the eavesdropping and message replaying because the receiver does not get the packet before the adversary forward it. Note that it is not necessary for an adversary to perform a *Man in the Middle* (MITM) attack in order to intercept packets.

3.2.4 Masquerading and Malicious AP

Since plaintext MAC addresses are included in the packet while transmitting it through the wireless links, an adversary can learn about the MAC addresses by eavesdropping. The adversary is also capable of modifying its MAC address to any value because most firmware provides the interface to do so. If a system uses MAC address as the only identification of the wireless devices, the adversary can masquerade as any wireless station by spoofing its MAC address. It can also masquerade as an AP by spoofing its MAC address and functioning appropriately through appropriate freeware e.g. HostAP.

3.2.5 Session Hijacking

An adversary may be able to hijack the legitimate session after the wireless devices have finished authenticating themselves successfully. In this type of attack, the adversary is able to receive all packets destined to the hijacked device and send out packets on behalf of the hijacked device. If data confidentiality and integrity protocols are used, the

adversary must break them in order to read encrypted traffic and send out valid packets. So, this attack can be prevented by using sufficiently powerful data confidentiality and integrity mechanisms.

3.2.6 Man in the Middle

This type of attack is different from message interception because the adversary must participate in communication continuously. If there is already connection between STN and the AP, the adversary must first break this connection. After that the adversary masquerades as the legitimate STN to associate the AP. And finally, the adversary must masquerade as the AP to fool the STN to associate with it.

3.2.7 Denial of Service

Wireless LAN systems are quite vulnerable to DoS attacks. An adversary is capable of making the whole BSS unavailable or disrupting the connection between legitimate peers. Using characteristics of wireless networking, an adversary may launch DoS attack in several ways. E.g. forging the unprotected management frames, exploiting some protocol weaknesses, jamming of frequency band etc.

3.3 Known Issues

This section will discuss some of the major issues related to the protocols used in 802.11i.

3.3.1 802.1X

[19] shows some of the design faults in 802.1X and its combination with 802.11. The authors claim that when 802.1X is used in combination with 802.11, it fails to provide strong access control and authentication. They encounter two major attack i.e. *Man in the*

Middle (MITM) attack and session hijacking. They perform these attacks by using tools developed as the part of Open1x project. The attack was successful because of several design flaws in 802.1X, EAP and 802.11.

There are also many issues in IEEE 802.1X pre-authentication as pointed out in [15]. These includes advertisement of capabilities, integration between 802.1X and 802.11 state machines, encapsulation of 802.1X pre-authentication data frames, secure cipher suite negotiation, authentication and integrity protection of management frames and key establishment.

3.3.2 RADIUS

The RADIUS implementation has a number of weaknesses as pointed out by Joshua Hill in his analysis of the RADIUS protocol [25]. His findings are summarized below:

- **Flawed Password Protection**
  RADIUS uses a Keyed Hash (MD5) as a stream cipher primitive, something which is not designed for. Hill enumerates four different attacks based on this vulnerability.

- **Flawed MIC generation**
  RADIUS uses MD5 to generate a MIC for all response messages. As MD5 has weaknesses, this is not a good idea.

- **Many client create insufficiently unpredictable nonces**
  Much of the security of RADIUS depends in the generation of the nonce called Request Authenticator. The RFC does not emphasize the importance of this enough and the result is a plethora of the poor implementation.

- **Bad shared secret hygiene**
  The RADIUS standard specifically permits use of the same shared secret by

many clients. This is a bad idea as a single flawed client will allow for several others to be compromised. This is possible as RADIUS provides no protection for client or server address. Many implementations also artificially limit the shared secret entropy by only allowing ASCII inputs and limiting the length to 16 characters. This greatly reduces the key space an attacker has to go through in order to find the shared secret.

As mentioned in RFC 2869, RADIUS protocol is also vulnerable to connection hijacking i.e. an attacker could inject packets into a conversation between the AP and RADIUS. This is possible as not all packets are integrity protected. Since RADIUS doesn't provide end-to-end security, MITM attacks are possible where an attacker could alter EAP packets in transit. Authentication method downgrading is also a threat if the AS accepts low-security methods such as EAP-MD5.

3.3.3 WPA And 802.11i

802.11i addresses most of the security issues in 802.11; most significantly it offers strong confidentiality and robust authentication. There are, however, still a few threats to the availability of the system.

The most practical vulnerability is the attack against PSK key. As already mentioned, the PSK provides an alternative to 802.1X PMK generation using authentication server. It is the string of 256 bits or a passphrase of 8 to 63 characters used to generate such a string. The PTK is derived from the PMK using 4-Way handshake and all information used to calculate its value is transmitted in the plaintext. So, the strength is PTK relies only on the PMK value, which for PSK effectively means the strength of the passphrase. As indicated by Robert Moskowitz in [32], the second message of 4-Way handshake could be subjected to both dictionary attack and brute force offline attack. The cowpatty utility [30] was created to exploit this flaw and its source code was used and improved by Christophe Devine in Aircrack [31] to allow PSK dictionary and brute force attack on WPA.

The other main weaknesses of WPA and 802.11i are a possibility of DoS attack during 4-Way handshake. According to [3], the first message of the 4-Way handshake isn't authenticated and each client has to store every first message until they receive a valid third message, leaving the client potentially vulnerable to memory exhaustion. By spoofing the first message send by the access point, an attacker can perform DoS on the client if it is possible for several simultaneous sessions to exist. Besides the DoS attack on 4-Way handshake [4] shows two new DoS attacks; RSN-IE (RSN Information Element) Poisoning and 4-Way handshake Blocking.

The Michael Message Integrity Code used in WPA has also known weaknesses resulting form the design. The security of Michael depends on the fact that the communication is being encrypted. But cryptographic MICs are usually designed to resists plaintext attacks. Michael is vulnerable to such type of attacks because it is invertible. Given a single known message and it MIC, it is possible to discover the secret MIC key. So, keeping the MIC value secret is critical.

3.4 PCL Proof System

Protocol Composition Logic (PCL) [40] is the logic for proving security properties of network protocols that uses public and symmetric key cryptography. PCL is a Floyd-Hoare style logic that supports axiomatic proofs of protocol properties. The logic is design around the process calculus with action for each protocol steps. The semantics of the logics is based on the set of traces of protocol execution, following the standard symbolic model of protocol execution and attack. Security proof involves local reasoning about properties guaranteed by individual actions and global reasoning about the actions of honest principals who faithfully follow the protocol. PCL supports compositional reasoning about complex security protocols such as SSL/TLS, IEEE 802.11i, Kerberos V5, etc.

3.4.1 Cord Calculus

One important part of security analysis involved understanding the way honest agents running a protocol will response to the messages from the malicious attacker. The common arrow-and-messages are generally insufficient because it only presents the executions (traces) of the protocol that occur when there is no attack. In addition, the protocol logics requires more information about a protocol than the set of protocol executions obtained from the honest and the malicious parties i.e. we need a high level description of the program executed e.g. by each agents performing each protocol role so that we know not only which actions occur in the run but why they occur.

Cord calculus was originally introduced in [41, 42]. Cord form an action structure based on *Π-Calculus* and *Spi-Calculus*. The basic idea of *Π-calculus* [43, 44] is to represent communication by term reduction, so that the communication link can be created dynamically. And the idea of spi is to add to Π the suitable constructors for encryption and decryption, and analyze secure communication in terms of bi-simulations and process equivalences.

Example of Cord Calculus

We can represent challenge-response as follows:

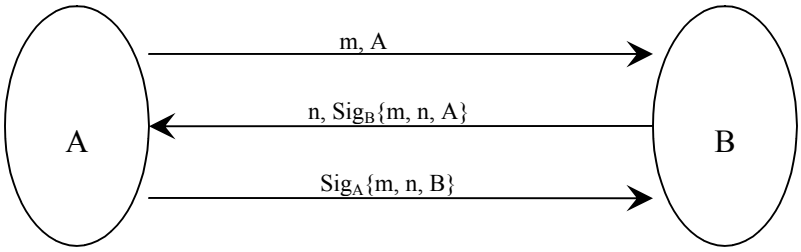


Figure 3.1 Challenge-Response Protocol

Here, Alice reason: if Bob is honest, then

- Only Bob can generate his signature.

- If Bob generates a signature of the form $Sig_B\{m, n, A\}$;
  - He sends it as the part of msg2 of the protocol.
  - He must have received msg1 from Alice.

According to the cord calculus, the challenge- response can be represented as follows:

InitCR(A, X) = [

New(m);

Send(A, X, m, A);

Receive(X, A, x, $Sig_X\{m, x, A\}$);

Send(A, X, $Sig_A\{m, x, X\}$)

]

RespCR(B) = [

Receive(Y, B, y, Y);

New(n);

Send(B, Y, n, $Sig_B\{y, n, Y\}$);

Receive(Y, B, $Sig_Y\{y, n, B\}$)

]

3.4.2 Protocol Logics

This section presents the syntax and semantics of the protocol logics.

Syntax

Action Predicates:

a ::= Send(P, m) | Receive(P, m) | Verify(P, t)|

Decrypt(P, t) | New(P, t)

Formulas:

ϕ ::= a | Has(P, t) | Fresh(P, t) | Honest(N) | Contains(t1,t2) | $\phi \wedge \phi$ | $\neg\phi$ |

$\exists x.\phi$ | $\diamond\phi$ | $\Theta\phi$ | Start(P)

Modal Formulas:

$\Psi ::= \phi\rho\phi$

Table: 3.1 Syntax of the logics

The formula for the logics is given by the above table. Here ρ may be given any role which is written using the notation of cord calculus. t and P denotes the term and thread respectively. A thread is the sequence of actions by principal executing an instance of a role e.g. Alice executing the initiator of the protocol. According to the notation convention $\hat{X}$ is the principal executing the thread X. ϕ and Ψ are used to indicate predicate formulas and m is used to indicate the generic term known as "message". A message consists of source, destination, protocol-identifier and content. The source field of a message may not identify the actual sender of the message. This is because the intruder can spoof the source address. Similarly, the principal identified by the destination field may not receive the message since an intruder can intercept message. Nevertheless, the source and destination fields in the message may be useful for starting and proving authentication properties while the protocol identifier is useful for proving properties of the protocols.

Most protocol proof uses formulas of the form $\theta[P]_X \phi$, which means that if X starts from a state in which θ holds, and executing the program P, then in the resulting state the security property ϕ is guaranteed to hold irrespective of the actions of an attacker and other honest agents. Many protocol properties are naturally expressible in this form.

The formula Has(X, x) means that the principal $\hat{X}$ possesses information x in the thread X. This is "possesses" in the limited sense of having either generated the data or received it under encryption where the decryption key is known. The formula Send(X, m) means that the last action in a run of the protocol corresponds to the principal $\hat{X}$ sending message m in the thread X. Receive(X, m), New(X, t), Decrypt(X, t) and Verify(X, t) are similarly associated with the receive, new, decrypt and signature verification actions of the protocol. Fresh(X, t) means that the term t generated in X is "fresh" in the sense that no one else has seen any term containing t as a sub-term. Generally, a fresh term will be a nonce and freshness will be used to reason about the temporal ordering of actions in runs of a protocol. The form of reasoning is useful in proving authentication properties of the

protocol. The formula Honest($\hat{X}$) means that $\hat{X}$ is acting honestly i.e. the actions of every thread of $\hat{X}$ precisely follow some role of the protocol. Contains(t_1, t_2) means t_2 is sub-term of t_1. This predicate helps us to find the components of the message. There are two temporary operators; $\diamond$ and Θ. $\diamond\phi$ means that in some state in the past ϕ holds. And $\Theta\phi$ means that in the previous state ϕ holds. Start(X) means that the thread X did not perform any actions in the past.

The formalization of authentication is based on the notion of matching the records of runs which requires that $\hat{A}$ and $\hat{B}$ accept each other's identities at the end of a run, their records of the run should match i.e. each message the $\hat{A}$ send was received by $\hat{B}$ and vice versa, each events send happened before the corresponding receive event and moreover the message send by each principal ($\hat{A}$ or $\hat{B}$) appear in the same order in both the records. Including the source and destination fields in the message allows us to match up send-receive actions. Since we reason about correctness of the protocol in an environment in which other protocols may be executing concurrently, it is important that when $\hat{A}$ and $\hat{B}$ accept each other's identities, they also agree on which protocol they have successfully completed with the other. One way to extend the matching histories characterization to capture this requirement is by adding protocol identifiers to messages. Now if $\hat{A}$ and $\hat{B}$ have matching histories at the end of a run, not only do they agree on the source, destination and content of each message, but also on which protocol this run is an instance of.

Semantics

The formula may be true or false at the run of a protocol. The main semantics relation is Q, R |= ϕ, which may be read as "formula ϕ may be hold for run R of protocol Q". In this relation, R may be complete run, with all sessions or an incomplete run with some principals waiting for additional messages to complete one or more sessions. If Q be the

protocol then let $\bar{Q}$ be the set of all initial configurations of protocol Q, each including the possible intruders. Let Run($\bar{Q}$) be the set of all runs of the protocol Q with intruder. If ϕ has free variable then Q, R |= ϕ, if we have Q, R |= $\sigma\ \phi$ where all the substitutions σ will eliminate all the free variables in ϕ. We can also write Q |= ϕ if Q, R |= ϕ for all R Є Run($\bar{Q}$).

Because a run is a sequence of reaction steps, each step resulting from a principal executing an action, it is possible to assert whether a particular action occurred in a given run and also to make assertions about the temporal ordering of the actions. An alternative view similar to the execution model is to think of a run as a linear sequence of states. Transition from one state to the next is effected by an action carried out by some principal in some role. A formula is true in a run if it is true in the last state of that run. So, an action formula a is true in a run if it is the last action in that run. Also a past formula ⟡a is true if in the past the action formula a was true in some state i.e. if the action has occurred in the past.

3.5 Summery

For accessing risks in a system, we generally use CIA Model. CIA Model consists of three components; Confidentiality, Integrity and Availability. In wireless technology, confidentiality can be achieve by using encryption, integrity by using MIC and availability by using load balancing server. In order to analyze the security of a system, it is also necessary to develop a Thread Model. With the help of thread model we can understand whether security objectives have been met of not and to evaluate alternative proposals.

The components used in IEEE 802.11i consist of various drawbacks. When 802.1X is combined with 802.11, two major types of attacks are possible; Man in the Middle attack, and Session hijacking. There are also many issues related to 802.1X pre-authentication

such as, advertisement of capabilities, integration between 802.1X and 802.11 state machine, secure cipher suite negotiation etc.

There are also various flaws found in RADIUS which include flawed password protection, flawed MIC generation, many clients create insufficiently unpredictable nonces and bad shared secret hygiene. RADIUS protocol is also vulnerable to connection hijacking i.e. an attacker can inject packets into conversation between AP and RADIUS server.

WPA and 802.11i also suffers from various attacks. WPA uses Michael for MIC calculation which is invertible. So, given a single known message and MIC, it is possible to discover the secret key of MIC. PSK is vulnerable to dictionary attack and brute force attack. Besides these vulnerabilities, DoS vulnerability is the major vulnerability in IEEE 802.11i. Because of the unauthentication of first message in 4-Way Handshake, there is a DoS vulnerability. Also there are two major vulnerabilities in IEEE 802.11i, namely RSN-IE poisoning and 4-Way Handshake block.

PCL is the logics for proving security properties of network protocols that uses public and symmetric key cryptography. It can be expressed by using modal formula $\theta[P]_X \phi$, which means if X starts from a state in which θ holds, and executing the program P then in the resulting state the security property ϕ is guaranteed to hold irrespective of the actions of an attacker.

CHAPTER 4
DoS ATTACK ON 802.11i

This chapter will discuss two main DoS attacks on IEEE 802.11i; DoS attack on 4-Way Handshake [3, 4] and RSN IE [4]. It also presents the possible solution to these attacks.

4.1 DoS Attack on 4-Way Handshake

The 4-way handshake is an essential component of the RSNA establishment. Its purpose is to confirm the possession of the shared PMK in the authenticator and the supplicant and to derive the fresh PTK for the subsequent data communication. In the handshake the authenticator and the supplicant generates its own nonces and send them to each other. The PTK is derived from the shared PMK, the nonces, and the MAC address of both the supplicant and the authenticator. Message 1 and 3 carry the nonce generated by the authenticator, Message 2 carries the nonce generated by the supplicant and Message 4 is an acknowledgement to indicate that the handshake is successfully completed.

Message 2, 3 and 4 are authenticated by the fresh PTK but Message 1 is unprotected. So, there will be PTK inconsistency. The attacker forged Message 1 and send to the supplicant after Message 2 of 4-Way Handshake. The supplicant then calculates new PTK corresponding to the nonces for the newly received Message 1. This causes supplicant handshake to be blocked because PTK is different from the authenticator. In order to prevent an adversary from affecting the PTK through forging Message 1, IEEE 802.11i adopts a Temporary PTK (TPTK) to store newly generated PTK until Message 3 is verified. But this approach doesn't prevent DoS attack on Message 1.

The supplicant must accepts all the Message 1s it receives in order to ensure that the handshake can complete in the case of packet loss and retransmission. If the supplicant received Message 3, it uses PTK corresponding to the nonce in the message to verify MIC. This approach does overcome the DoS vulnerability but it suffers from memory

exhaustion attack. So, an adversary is able to launch a memory DoS attack by sending out numerous forged Message 1s. This attack is serious because it is simple for the adversary to perform and a successful attack will cancel all the efforts in the previous authentication process.

4.2 Solution to DoS Attack on 4-Way Handshake

In order to prevent from DoS vulnerability and memory DoS vulnerability, [3, 4] proposes the nonce reuse method. This approach eliminates the intermediate states on the supplicant side i.e. the supplicant doesn't update its nonce responding to each received Message 1 until Message 3 is received and verified. This approach does remove memory DoS vulnerability but it uses more computation on supplicant i.e. PTK should be calculated twice for each received nonce (when supplicant received Message 1 and Message 3). Besides the computation, when attacker forge Message 1 and send it repeatedly to the supplicant, it received same Message 2 for a longer period of time. So, it has more time to analyze about the packet and can generate various other attacks. Detail discussion of this problem and its solution has been done in 5.2.2.

4.3 RSN IE Poisoning

RSN IE verification mechanism also introduces the possibility of DoS attack on IEEE 802.11i.

Element ID	Length
Version	
Group Key Cipher Suite	
Pairwise Key Cipher Suite Count	
Pairwise Key Cipher Suit List	
Authentication and Key Management Cipher Suite Count	
Authentication and Key Management Cipher Suite List	
RSN Capabilities	

PMKID Count
PMKID List

Figure 4.1 RSN IE Format

RSN IE contains authentication and pairwise key cipher suite selectors, a single group key cipher suite selector, RSN capability field, the PMKID count and PMKID list. The authenticator should insert the supported RSN IEs in the Beacon and Probe Response and the supplicant should insert its chosen RSN IE in the re-association request. The authenticator and the supplicant should use the negotiated security suites to perform the authentication and key management protocol and used the negotiated cipher suites to encrypt data communication. In order to confirm the authenticity of the RSN IEs, the supplicant should include the same RSN IE in Message 2 of the 4-Way Handshake. The authenticator is also required to include the same RSN IE in Message 3 of the 4-Way Handshake as in the Beacon or Probe Response. After receiving the Message 2, the authenticator will check the RSN IE by using bit-wise comparison with the one it receive in the re-association request from the supplicant, in order to confirm that they are exactly the same. The supplicant will also perform the bit-wise comparison for RSN IE in Message 3 with the one it receives in Beacon or Probe Response. If RSN IEs are not exactly the same, the supplicant and the authenticator will deauthenticate each other. This confirmation procedure introduces the possibility of DoS attacks.

In Message 2 of the 4-Way Handshake, the authenticator verifies the MIC before the RSN IE, which is in the correct order; but in the Message 3, the supplicant checks the RSN IE before the MIC verification and abort if the RSN IE is unmatched. An adversary can easily modify the RSN IE in Message 3 to cause the handshake to fail. Even if the order is in correct form, there is another fundamental attack which causes the RSN IE confirmation process to fail.

An adversary can easily eavesdrop on the Beacon frames of an authenticator and can modify several bits in the frame that are insignificant i.e. the modification of these bits

doesn't effect the validity of the frame and the selection of the authentication cipher suites. For example, the reserved bit and the replay counter bits in the RSN capabilities fields are insignificant. The supplicant then broadcasts this forged Beacon to poison the knowledge of RSN IEs in the supplicants. Because this forged Beacon only consists of modified insignificant bits, the supplicant and the authenticator are still able to continue the authentication and key management using the effective security suites. But the 4-Way handshake will never be succeeding because the RSN IE confirmation will fail.

RSN IE poisoning is different from the security level role back attack because the adversary doesn't aim to establish the successful connection. It simply blocks the protocol execution. This attack is considered to be harmful because it is quite easy for an adversary to implement and it will affect all the supplicants simultaneously when the forged Beacon is allowed. Also because the supplicant and the authenticator are unaware of the RSN IE poisoning, they might continue to execute considerable number of message until 4-Way Handshake fails. This wastes the resources of the authenticator and the supplicant; also the adversary will have more time to periodically repeat its attacks.

This weakness is because of the following three reasons:

- The management frames like Beacon, Probe Response and Re-association Request are not protected.
- There are a number of message exchanges between the RSN IE negotiation and confirmation, which consume resources and leave more time for the adversary.
- The bit-wise comparison in 4-Way Handshake might be unnecessarily strict to confirm RSN IE.

4.4 Solution to RSN IE Poisoning

RSN IE poisoning can be removed by authenticating the management frames. But in some situations this might not be acceptable to authenticate the Beacon or Probe Response frames. This is because the authenticator and the supplicant do not share secret

at the beginning. So, an acceptable approach is to do RSN IE confirmation as soon as possible to avoid wasting of the message exchanges and makes the attack less destructive.

This attack can also be minimized by loosening the condition of the RSN IE confirmation i.e. the authenticator and the supplicant can ignore the differences of the insignificant bits in the corresponding RSN IEs, while keeping the negotiation secure. If an adversary does not change the authentication and key management suite selector, the RSN IE could be accepted because the correct authentication has been executed. And hence the authenticator and the supplicant can use the authenticated RSN IE in the 4-Way Handshake for the subsequent data encryptions. If the adversary modifies the authentication and key management suite selector, this can be detected at the beginning of the association. The association fails and the supplicant retries quickly without continuing message exchanges. In the worse case, this modification can be prevented in the 4-Way Handshake.

4.5 Summery

The first message of 4-Way Handshake is not authenticated due to which there are DoS vulnerable in 4-Way Handshake. Suppose authenticator send first message and supplicant received and derived PTK corresponding to that message. It then send second message. In the mean while if an attacker send forged first message and supplicant received it before third message of authenticator, it will recalculate PTK based on the first message of attacker. So, when third message is received from authenticator, supplicant simply rejects that because PTK has been different.

In order to solve this issue, the supplicant must accept all the first messages and store it until a valid third message is received. This process will remove the DoS vulnerability but suffers from memory exhaustion attack. So, John C. Mitchell and Anapum Datta proposed a nonce reuse method in supplicant side. This process will remove both DoS vulnerability and memory exhaustion attacks but it violet the use of nonce, according to the original IEEE 802.11i specification. This problem was addressed in section 5.2.2.

Another major vulnerability is RSN IE poisoning. RSN IE poisoning is because of the three reasons:

- Management frames are not protected.
- Between RSN IE negotiation and confirmation, many messages are exchanges between supplicant and authenticator.
- RSN IE confirmation is done by bit-wise comparison.

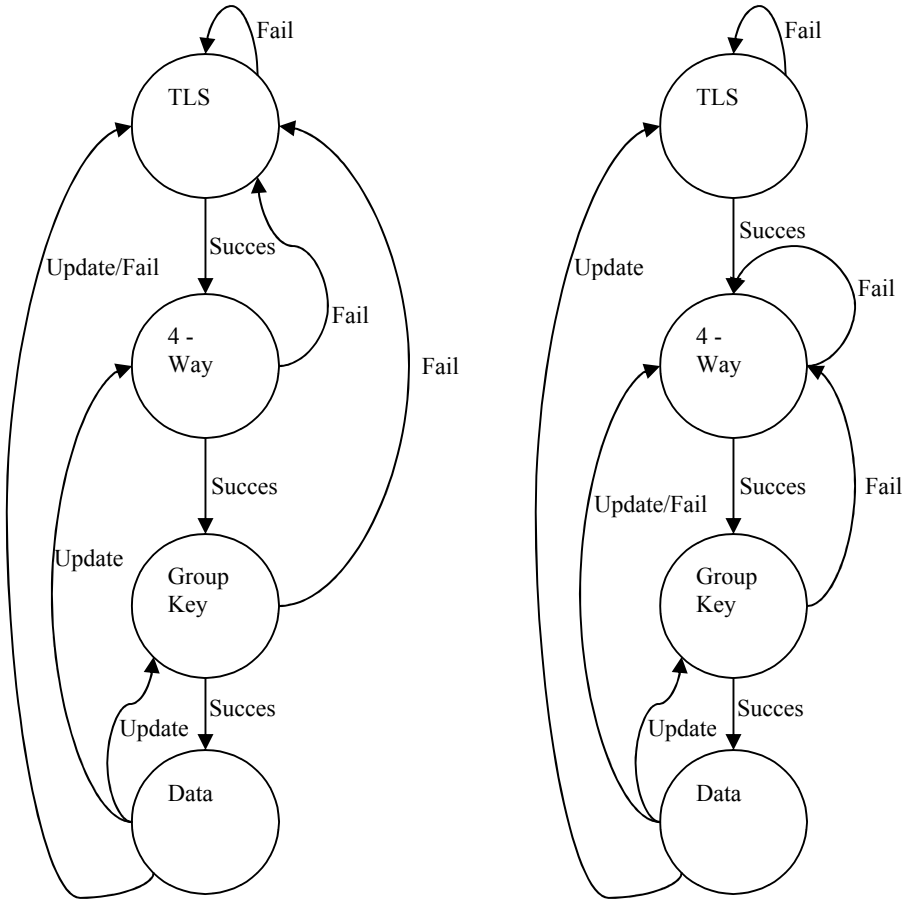
RSN IE poisoning can be minimized by adopting one of the following ways:

- Authenticating management frames.
- Confirming RSN IE as soon as possible.
- Ignoring the insignificant bits in RSN IE while confirmation.

CHAPTER 5
IMPROVEMENTS IN 802.11i AND ITS CORRECTNESS PROOF

This chapter presents the analysis of existing IEEE 802.11i by using PCL and also proposes its improvements. The proposed solution was also verified by using PCL theoretically and practically by using ns2 and xgraph.

5.1 Failure Recovery



a. Original Failure Recovery **b. Improved Failure Recovery**

Figure 5.1 IEEE 802.11i Flow Control

The IEEE 802.11i wireless network protocol provide mutual authentication between a network access point and user device prior to user authentication. The protocol consists of several parts; 802.1X authentication phase using TLS over EAP, 4-Way handshake to establish a fresh session key and an optional group key handshake for group communication. Failure in any one component also leads to failure in other possible execution sequences. In the original IEEE 802.11i specification, the entire sequence is restarted if one of the components fails. According to [4] this failure recovery is very inefficient, which may be improved as shown in Figure 5.1.

5.2 4-Way Handshake

The 4-Way Handshake generates the *Pairwise Temporary Key* (PTK) for data confidentiality protocols using a pre-established secret shared between the authenticator and the supplicant. The pre-established secret, which is known as *Pairwise Master Key* (PMK), may be set up via mutual authentication protocols (de facto EAP-TLS), or it may be pre-configured as a *Pre-Shared Key* (PSK). During the handshake, the authenticator and the supplicant generate fresh nonce, and then derive a fresh PTK based on the shared PMK, the nonce and their MAC addresses. The authenticator and supplicant's roles of the 4-Way handshake are described formally using PCL are as follows:

4-Way : AUTH = (X, $\hat{Y}$, pmk)

[New(x); Send($\hat{X}$, $\hat{Y}$, x, msg1);

Receive($\hat{Y}$, $\hat{X}$, z); Match(z/y, msg2, mic1);

Match($HASH_{pmk}$(x,y)/ptk));

Match(mic1/$HASH_{ptk}$(y,msg2));

Send($\hat{X}$, $\hat{Y}$, x, msg3, $HASH_{ptk}$(x, msg3);

Receive($\hat{Y}$, $\hat{X}$, w);

Match(w/msg4, mic2); Match(mic2/$HASH_{ptk}$(msg4))]x

4-Way : SUPP = (Y, pmk)

[Receive($\hat{X}$, $\hat{Y}$, z); Match(z/x, msg1);
New(y); Match($HASH_{pmk}$(x,y)/ptk);

Send($\hat{Y}$, $\hat{X}$, y, msg2, $HASH_{ptk}$(y, msg2));

Receive($\hat{X}$, $\hat{Y}$, w);
Match(w/x, msg3, mic); Match(mic/$HASH_{ptk}$(x, msg3);

Send($\hat{Y}$, $\hat{X}$, msg4, $HASH_{ptk}$(msg4))]y

Table 5.1 4-Way Handshake Program

4-Way : AUTH program has three inputs; the authenticator identity, the supplicant identity and the PMK represented by the variable pmk. The first action executed by the authenticator X involves generating a fresh nonce, x, using the action New. Then X sends out Message 1 to Y, which contains the nonce, x, and the string msg1. Authenticator X waits to receive a response from Y and then checks whether the received message is the expected Message 2, using the Match action, which verify the Message Integrity Code (MIC) based on the derived ptk. If Message 2 is valid, X sends out message 3 including the nonce x, the string msg3, and the MIC, and wait for response. If a valid Message 4 is received and verified, X completes the 4-Way handshake and subsequently uses the derived ptk. According to the IEEE 802.11i specification, the PTK is divided into three parts; KCK (*Key Confirmation Key*) for computing MIC, KEK (*Key Encryption Key*) for encrypting the group key, and TK (*Temporary Key*) for protecting data packets. For simplicity, ptk is used to refer to all these three parts.

The program 4-Way : SUPP consists of 2 inputs; the supplicant identity and *Pairwise Master Key* (PMK), which is denoted in program by pmk. It waits for the first message send by the authenticator. After receiving the first message it then check weather the received message is correct one or not by using the action called Match. If it has received the correct message then, it generates the nonce, y, and then sends it to authenticator along with the Message 2, which is encrypted by using ptk and nonce. After Message 2 has been send, it waits for the valid Message 3 from authenticator. If the valid Message 3

has been received which will be verified by using Match action, which verify the MIC based on ptk. And finally Message 4 is send by encrypting it using ptk.

5.2.1 4-Way Handshake Security Properties

According to the IEEE 802.11i specification, 4-Way handshake consists of following security properties:

a. PMK is present in both supplicant and the authenticator.
b. The generated PTK is fresh.
c. Synchronize the installation of session keys into the MAC.
d. Transfer the GTK from the authenticator to the supplicant.
e. Confirm the selection of the cipher suites.

This section covers two main security properties; key secrecy and session authentication. a, b, c and e are covered by session authentication and also session authentication can be obtained only when the key secrecy is guaranteed. d will be discussed in section 5.3.

The formula, $\phi_{4way,\ sec}$, which is stated below shows that only authenticator and the supplicant possess the PTK i.e. the 4-Way Handshake is said to provide key secrecy if $\phi_{4way,\ sec}$ holds:

$$\phi_{4way,\ sec} = \text{Honest}(\hat{X}) \wedge \text{Honest}(\hat{Y}) \supset$$
$$(\text{Has}(\hat{Z}, \text{ptk}) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{Y}) \wedge$$
$$\text{Has}(\hat{X}, \text{ptk}) \wedge \text{Has}(\hat{Y}, \text{ptk})$$

Also the 4-Way handshake is said to provide session authentication for authenticator if $\phi_{4way,\ Auth}$ holds.

$$\phi_{4way,\ Auth} = \text{Honest}(\hat{X}) \wedge \text{Honest}(\hat{Y}) \supset$$
$$\exists .\text{ActionsInOrder}($$
$$\text{Send}(X, \hat{X}, \hat{Y}, \text{Message 1}),$$
$$\text{Receive}(Y, \hat{X}, \hat{Y}, \text{Message 1}),$$

Send(Y, $\hat{Y}$, $\hat{X}$, Message 2),

Receive(X, $\hat{Y}$, $\hat{X}$, Message 2),

Send(X, $\hat{X}$, $\hat{Y}$, Message 3),

Receive(Y, $\hat{X}$, $\hat{Y}$, Message 3),

Send(Y, $\hat{Y}$, $\hat{X}$, Message 4),

Receive(X, $\hat{Y}$, $\hat{X}$, Message 4))

5.2.2 Improved 4-Way Handshake

As mentioned earlier, the 4-Way Handshake suffers from DoS vulnerability. The vulnerability is due to the lack of any authentication in Message 1, which allows the attacker to block the supplicant role. A modification that involves nonce reuse has been discusses in [3] and [4] and has been adopted by the IEEE 802.11i standard committee. The modified supplicant program in pseudo-code is given below but the authenticator program is same as the original specification:

New(y); //generate nonce
repeat
    Receive($\hat{X}$, $\hat{Y}$, z); //Receive message
    if Match(z, msg1) then //check if the received message is MSG1
        Send($\hat{Y}$, $\hat{X}$, y, msg2, $HASH_{ptk}$(y, msg2)); //send MSG2
    end if
until Match(z, msg3, $HASH_{ptk}$(x, msg3)); //check the validity of MSG3
Send($\hat{Y}$, $\hat{X}$, msg4, $HASH_{ptk}$(msg4)); //send MSG4

Modified Supplicant Program with nonce reuse

The modified program doesn't require storing the state information by the supplicant. So,

it prevents memory exhaustion problem but it violates the use of nonce i.e. nonces are used to guarantees the freshness. An adversary may eavesdrop and get information about the nonce because same nonce is seen for long period. So, an attacker has more time to analyze it. We can preserve the freshness property by storing certain number of Message 1 in the supplicant. Suppose n number of messages can be stored by the supplicant. If (n +1)th message is received, the supplicant will randomly pick one message and drop it, so that it won't suffer form memory exhaustion and also freshness property is guaranteed.

According to this improvement, the authenticator program need not to be modified i.e. it should be same as the original IEEE 802.11i specification but the supplicant program should be modified as below:

```
Receive(X̂ , Ŷ , z);              //Receive message
if Match(z, msg1) then
        if (IsLess(buffer_size ,Count(message_seen))) then
                //drop one message by randomly picking it
                Drop(message_seen, Random(buffer_size));
        end if
        if NotFound(message_seen, msg1) then
                //add message in the message list
                Append(message_seen, msg1);
        end if
        New(y);                  //generate nonce
        Send(Ŷ , X̂ , y, msg2, HASH_ptk(y, msg2));    //send MSG2
else if Match(z, msg3, HASH_ptk(x, msg3)) then
        Send(Ŷ , X̂ , msg4, HASH_ptk(msg4));          //send MSG4
end if
```

Modified Supplicant Program without nonce reuse[2]

[2] Appendix A presents ns2 simulation of the complete program and its NAM simulation

The modified program for supplicant without nonce reuse prevents from DoS attack due to unauthentication of Message 1 in 4-Way Handshake. If an attacker, forge Message 1 and send it repeatedly, the supplicant does accept it and store in the message_seen list. But if the message_seen list is full, it will pick message randomly and drop it. Thus, the supplicant won't suffer from DoS attack but in worse condition, it may drop message from the access point. So, the authentication may get some delay but this drawback will be less costly than the DoS attack and the problem generated by nonce reuse.

5.2.3 Practical Analysis of the Improvement in 4-Way Handshake

Implementation of the nonce reuse in supplicant program and proposed modification in supplicant program has been done in ns2 (*Network Simulator 2)* [46, 47, 48, 49] and the graphical analysis is done by using xgraph [50]. The analysis was done on two set of data. In first set attacker and authenticator both send 10 *Message 1* each to the supplicants. Authenticator starts sending message after 0.1 sec of the simulation time but the attacker start sending only after 0.2 sec of the simulation time. Both authenticator and the attacker send packets at an interval of 0.1 sec. In the second set of data 20 *Message 1* are send by both attacker and the authenticator. The packet transmission and interval is same as in the first case.

5.2.3.1 Analysis of Packet Received and Processed

Packets received and processed by the supplicant were analyzed by using xgraph [50]. The graphical analysis shows that the packet received and processed by supplicants in modified supplicant program is almost identical to that of the packet received and processed by the nonce reuse program. Figure 5.2 and 5.3 shows the packet received by supplicant in modified program without nonce reuse and program with nonce reuse respectively in which 10 *Message 1* are send by both attacker and authenticator.

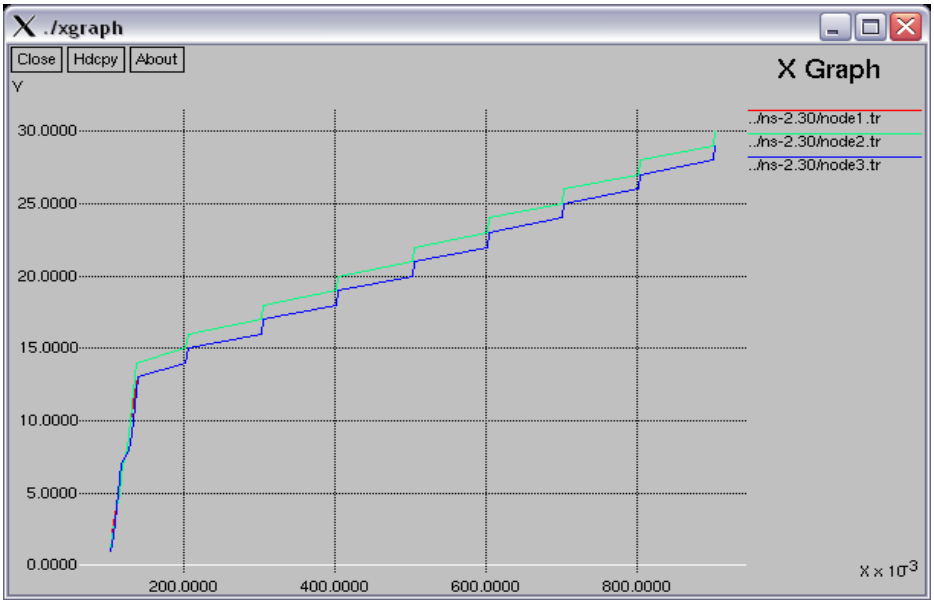


Figure 5.2 Packet Received By Supplicants (Without Nonce Reuse, 10 Packets)

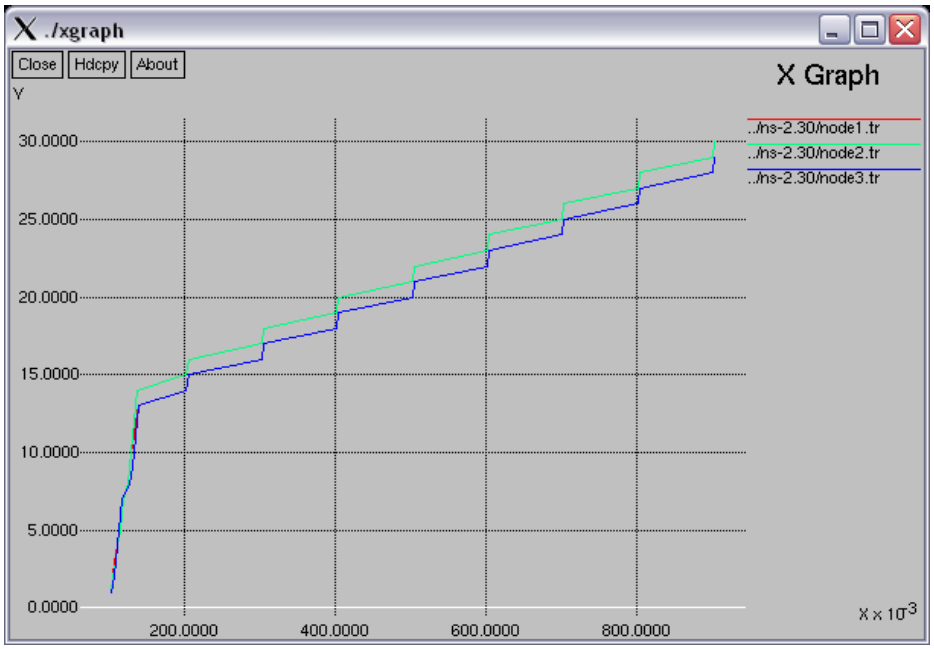


Figure 5.3 Packet Received By Supplicants (Nonce Reuse, 10 Packets)

Figure 5.4 and 5.5 shows the packet received by supplicant in modified program without nonce reuse and program with nonce reuse respectively in which 20 *Message 1* are send by both attacker and authenticator.

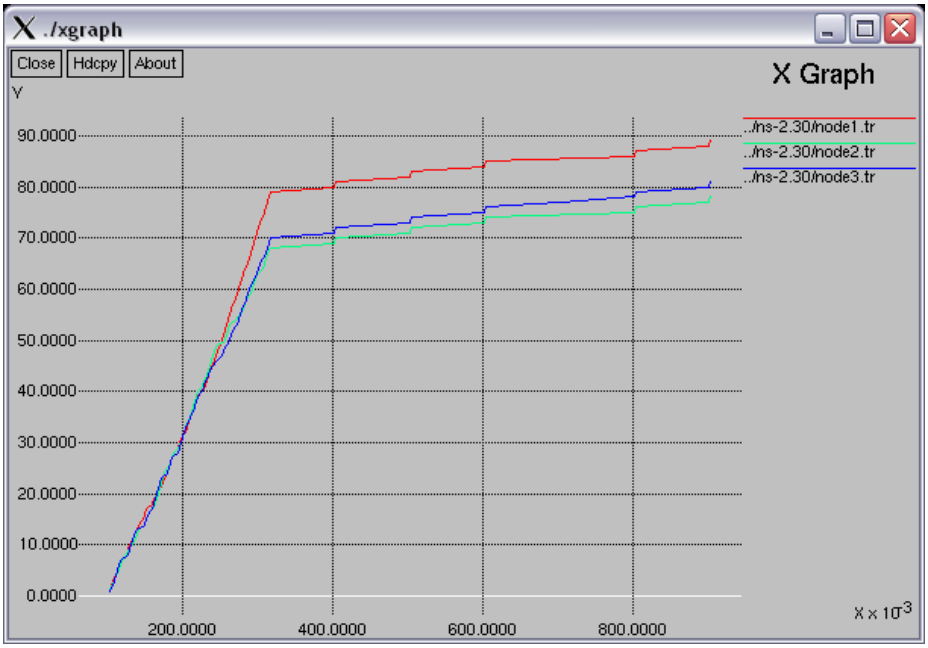


Figure 5.4 Packet Received By Supplicants (Without Nonce Reuse, 20 Packets)

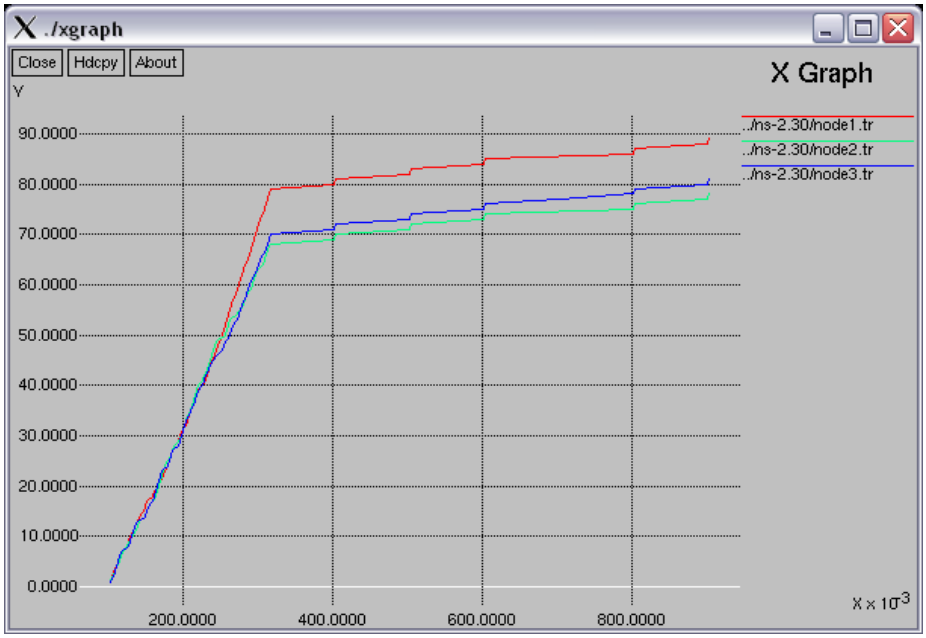


Figure 5.5 Packet Received By Supplicants (Nonce Reuse, 20 Packets)

Figure 5.6 and 5.7 shows the packet processed by supplicant in modified program without nonce reuse and program with nonce reuse respectively in which 10 *Message 1* are send by both attacker and authenticator.

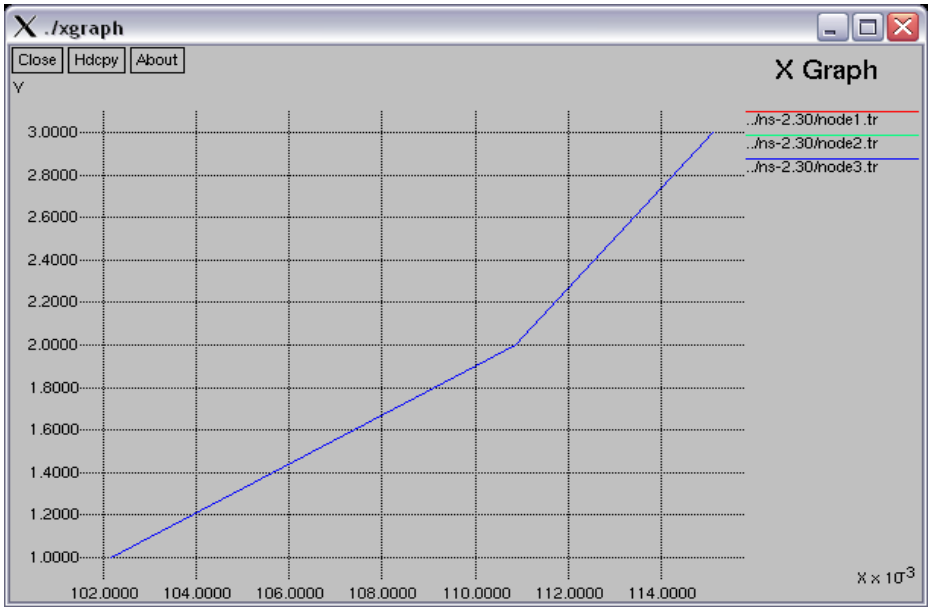


Figure 5.6 Packet Processed By Supplicants (Without Nonce Reuse, 10 Packets)

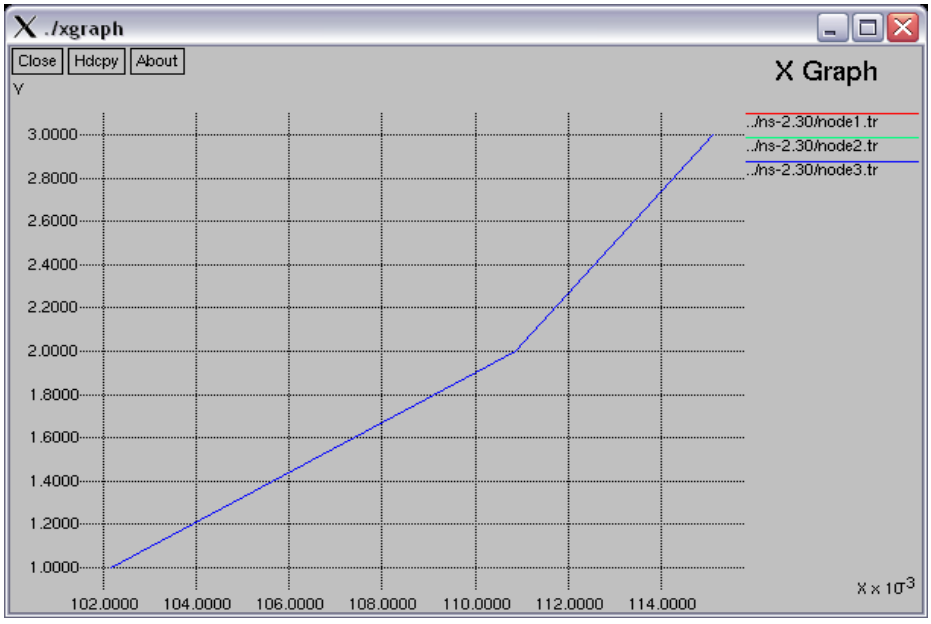


Figure 5.7 Packet Processed By Supplicants (Nonce Reuse, 10 Packets)

Figure 5.8 and 5.9 shows the packet processed by supplicant in modified program without nonce reuse and program with nonce reuse respectively in which 20 *Message 1* are send by both attacker and authenticator.

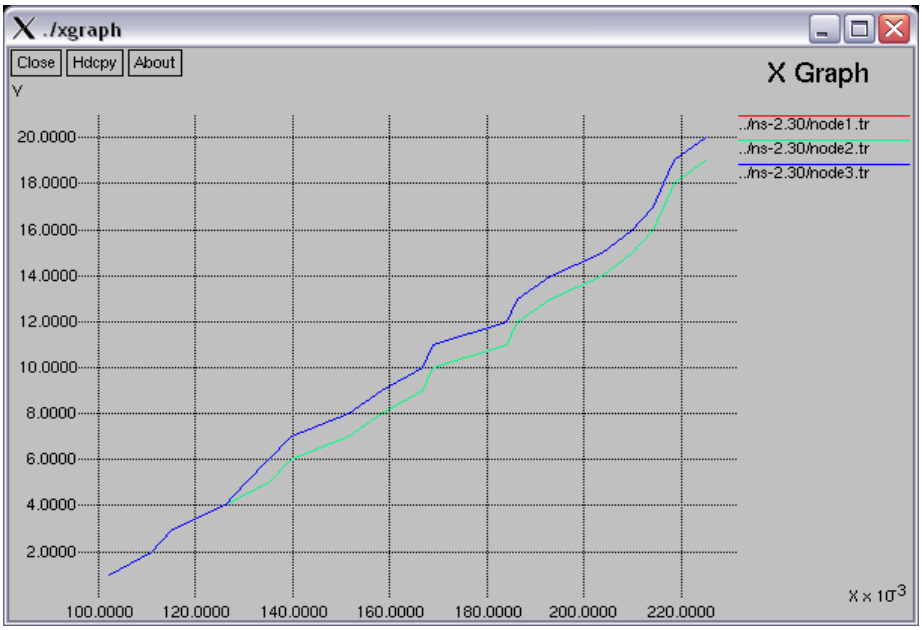


Figure 5.8 Packet Processed By Supplicants (Without Nonce Reuse, 20 Packets)

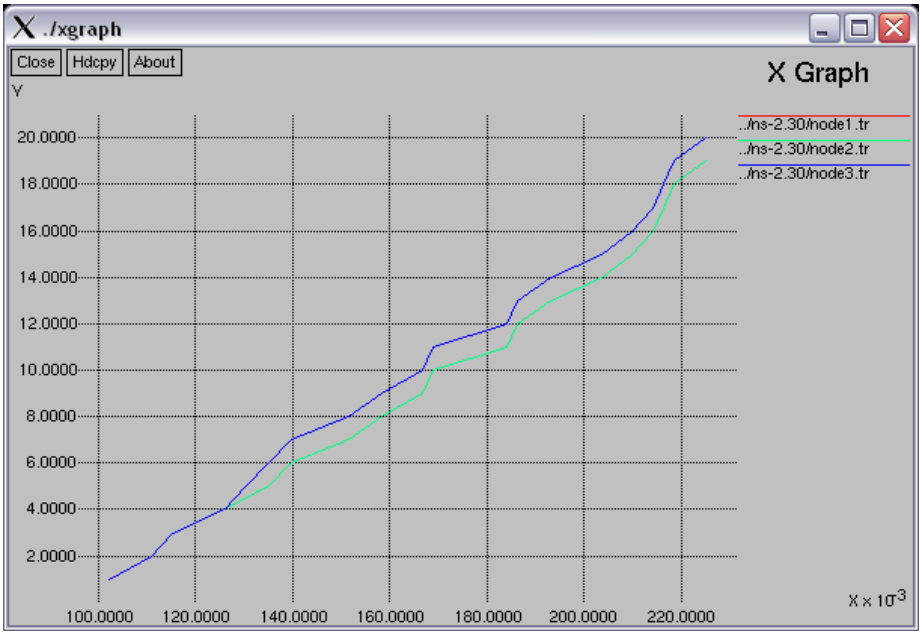


Figure 5.9 Packet Processed By Supplicants (Nonce Reuse, 20 Packets)

The above graphical analysis shows that both the modified supplicant program with and without nonce reuse overcome the DoS vulnerability in 4-Way Handshake.

5.2.3.2 Analysis of Packet Received by Attacker

The packet received by the attacker was also analyzed by using xgraph [50]. The graphical analysis shows that attacker got same packet longer period of time in nonce reuse; due to which it got more time to analyze about the packet and may launch various other attacks. But in the case of modified program without nonce reuse, each time attacker got a different packet. So, it is more difficult for the attacker to analyze the packets in this situation. Figure 5.10 and 5.11 shows the graphical representation of packet received by attacker in modified program without nonce reuse verses program with nonce reuse in which 10 *Message 1* and 20 *Message 1* are send by both attacker and authenticator respectively.

In figure 5.10 and 5.11, green (lower) line represent the graphical representation of packets received by program with nonce reuse and red (upper) line represent the graphical representation of the packets received by program without nonce reuse. Figure 5.10 shows that attacker received same packets from 119 millisecond to 127 millisecond and also same packet was received in between 131 and 133 milliseconds. Also we can see various intervals in figure 5.11 in which same packets are received. Thus, this graphical analysis shows that modified program without nonce reuse is more secure than the modified program with nonce reuse.

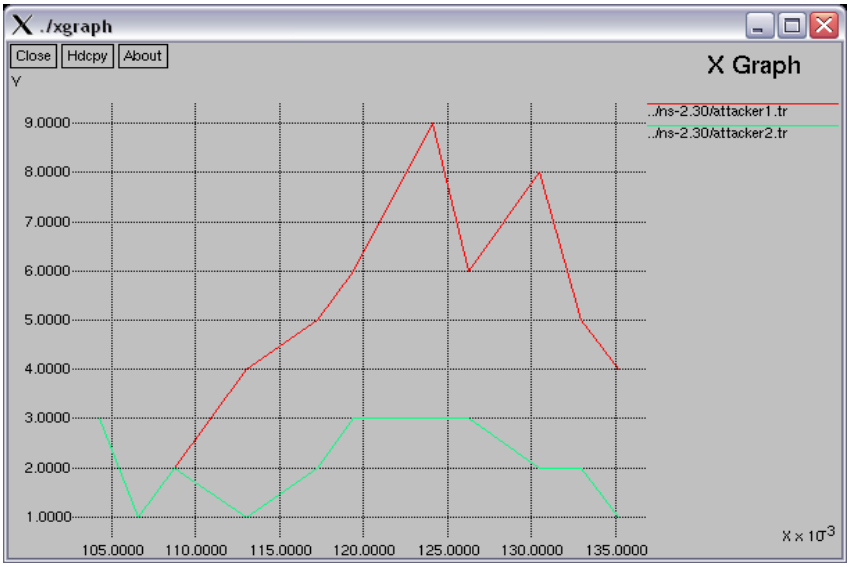


Figure 5.10 Packet received by attacker (10 packets)

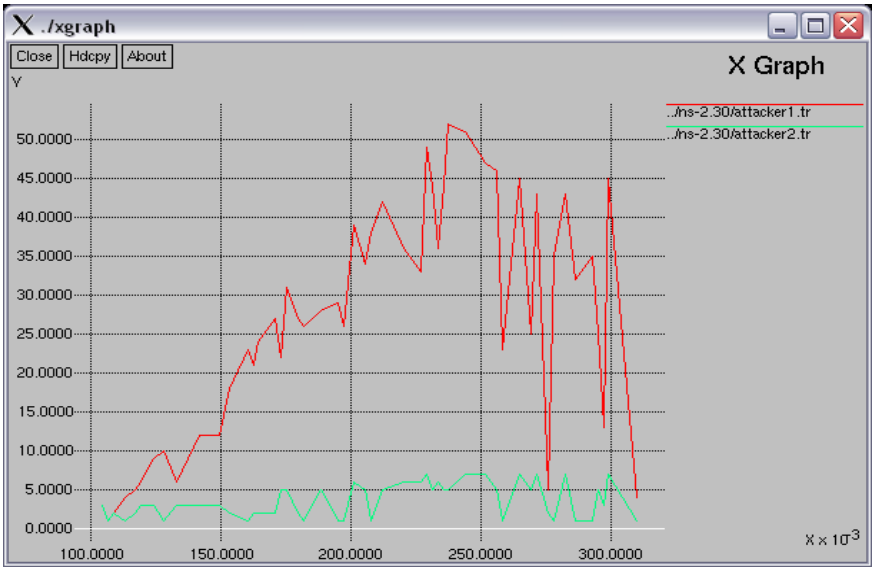


Figure 5.11 Packet received by attacker (20 packets)

5.3 Group-Key Handshake

In multicast applications, the authenticator distributes a *Group Temporary Key* (GTK) to the supplicants. Messages 3 and 4 of 4-Way handshake are used to setup this key distribution. The authenticator then runs Group Key Handshake protocol periodically to update the GTK.

Following table shows the program for Group Key Handshake:

GK : AUTH = (X, $\hat{Y}$, CurrSeqNo, ptk, gtk)[

Match(Succ(CurrSeqNo)/NewSeqNo);

Send($\hat{X}$, $\hat{Y}$, NewSeqNo, grp1, ENC_{ptk}(gtk));

$HASH_{ptk}$(NewSeqNo, grp1, ENC_{ptk}(gtk));

Receive($\hat{Y}$, $\hat{X}$, z);

Match(z/NewSeqNo, grp2, $HASH_{ptk}$(NewSeqNo, grp2))]x

GK : SUPP = (Y, $\hat{X}$, OldSeqNo, OldGTK, ptk)[

Receive($\hat{X}$, $\hat{Y}$, z);
Match(z/NewSeqNo, grp1, ENC_{ptk}(gtk),
$HASH_{ptk}$(NewSeqNo, grp1, ENC_{ptk}(gtk));
IsLess(OldSeqNo, NewSeqNo);
Send($\hat{Y}$, $\hat{X}$, NewSeqNo, grp2,
$HASH_{ptk}$(NewSeqNo, grp2);]y

Table 5.2 Group Key Handshake Program

The authenticator sends GrpMessage1 containing the GTK, and the supplicant confirms receipt of the GTK by sending GrpMessage2. The authenticator encrypts the GTK under the Key Encryption Key (KEK)[3] and sends it to the supplicant. The authenticator monotonically increases the sequence number for every key exchange message sent to prevent replay attacks. MICs are used to provide authentication and message integrity. The sequence number comparison IsLess(a, b) is used by the supplicant to check weather a < b and the expression Succ(a) represents a number greater than a.

5.4.1 Security Properties of Group Key Handshake

Group Key Handshake should satisfied following two properties

- **Key Ordering Property**
  The supplicant should received the GTK in the current Group Key Handshake should be send by the authenticator and authenticator should generate that GTK after the GTK that the supplicant holds from the previous Group Key Handshake.

- **Key Secrecy Property**
  The supplicant with the knowledge of the GTK must have executed 4-Way

[3] In the program it was represented by ptk

Handshake with the authenticator.

For the supplicant $\hat{Y}$, the Group Key Handshake is said to provide key ordering if $\phi_{gk,ord}$ holds, where

$$\phi_{gk,ord} = \text{Honest}(\hat{X}) \supset$$
$$(\text{Send}(X, \hat{X}, \hat{Y}, \text{SeqNo1}, \text{ENC}_{ptk}(gtk1)) \wedge$$
$$\text{Send}(X, \hat{X}, \hat{Y}, \text{SeqNo2}, \text{ENC}_{ptk}(gtk2)) \wedge$$
$$\text{IsLess}(\text{SeqNo1}, \text{SeqNo2}) \supset$$
$$\text{FirstSend}(X, \hat{X}, \hat{Y}, \text{SeqNo1}, \text{ENC}_{ptk}(gtk1)) <$$
$$\text{FirstSend}(X, \hat{X}, \hat{Y}, \text{SeqNo2}, \text{ENC}_{ptk}(gtk2)))$$

For an authenticator $\hat{X}$, the Group Key Handshake is said to provide key secrecy if $\phi_{gk,sec}$ holds, where

$$\phi_{gk,sec} = \text{Honest}(\hat{Z}_1) \wedge \text{Honest}(\hat{Z}_2) \wedge \ldots\ldots\ldots\ldots \wedge \text{Honest}(\hat{Z}_n) \supset$$
$$((\text{Has}(\hat{Z}, gtk) \wedge \hat{Z} \neq \hat{X}) \supset$$
$$\hat{Z} = \hat{Z}_1 \vee \hat{Z} = \hat{Z}_2 \ldots\ldots\ldots \vee \hat{Z} = \hat{Z}_n)$$

5.4 Summery

IEEE 802.11i consists of four parts; 802.1X authentication, 4-Way Handshake, group key handshake and actual communication. According to the original specification, when there is a failure in any one of the steps, whole process should be repeated which is not efficient. This can be improved by just repeating current step or parent step or whole steps according to where the failure occurs.

Due to unauthentication of first message in 4-Way Handshake, there is a DoS vulnerability in 4-Way Handshake. This DoS vulnerability was addressed in [3, 4] by

reusing the nonce. According to the IEEE specification nonce is used to guarantee freshness. So, nonce reused method violet the use of nonce. Also when an attacker is able to capture the packet, it will get same packet (second message) for a longer period of time. So, an attacker got lots of time to analyze about the packet. Thus, if he/she is successful, he may know how to calculate PTK, which is a key for encryption. In order to solve this problem the proposed solution, without nonce reuse, maintain a queue in the supplicant side. The supplicant stores first message and its corresponding PTK into the queue until valid third message is received. When the queue is full, it will pick one random item and drop it. After receiving valid third message, it will drop all the items from the queue and the valid key will be installed.

The practical analysis of proposed solution by using ns2 shows that it removes the DoS attack due to unauthentication of the first message and also removes the problem found in nonce reuse method. The proposed solution has two drawbacks. First certain memory is required in the supplicant for maintaining the queue. The size of queue depends on the supplicant used. Second, when a queue is full, one random item will be dropped. In the worst case the randomly picked item may be the item generated from the actual authenticator. In this situation, the authentication process will be delayed for some time.

CHAPTER 6
CONCLUSION AND FUTURE WORK

There are three types of wireless protocols; WEP, WPA and WPA 2 (IEEE 802.11i). WEP is the data encapsulation technique used by initial IEEE 802.11. There are lots of weaknesses in WEP. So, it was absolute in very short period of time. In order to overcome the drawback of WEP, WPA was developed. WPA was aimed to use instead of WEP until IEEE 802.11i was fully developed. So, it is designed as the software patch to upgrade WEP in already deployed equipment. Since WPA also uses RC4 as encryption algorithm, so, it doesn't required separate hardware. WPA does address weaknesses of WEP but it also have some drawbacks such as it uses weaker message integrity called *Michael*.

In order to provide long term solution to the wireless security IEEE 802.11i was developed. It addresses all the weaknesses of WEP and WPA. IEEE 802.11i uses AES as encryption algorithm. The specification of IEEE 802.11i defines two classes of security algorithms; RSNA and Pre-RSNA. Pre-RSNA security consists of WEP and 802.11 entity authentication. RSNA provides two data confidentiality protocols, called the TKIP and CCMP and RSNA establishment procedure, including 802.1X authentication and key management protocols.

IEEE 802.11i, if implemented correctly, is assumed to be the most secure protocol among wireless family but it does suffer from various DoS attacks such as RSN IE Poisoning, 4-Way Handshake blocking etc. RSN IE Poisoning is due to the defect in checking the RSN IE i.e. bit wise comparison and also the verification of RSN IE is done during 4-Way Handshake. So, if an adversary eavesdrops and modifies some bits in nonce, the STN and AP continue to exchange message until 4-Way Handshake. 4-Way Handshake block is because *Message 1* in 4-Way Handshake is not authenticated. So, there is PTK inconsistency. In order to solve this problem, STN stores all the received nonce and derived PTKs, until it finishes the handshake. This solution solved the DoS vulnerability

but it suffer from *Memory Exhaustion*. Again in order to solve the memory exhaustion problem, nonces were reused in supplicant program. The nonce reuse method also suffer from other vulnerability because same packet were received by the attacker for longer period of time i.e. it violets the freshness property as guaranteed by IEEE 80.11i.

This study presents a formal correctness proof for the IEEE 802.11i using PCL. The PCL proof demonstrates the need for separate keys for supplicant and authenticator to prevent a reflection attack. It also shows that the reduction in DoS vulnerability by using nonce reuse in 4-Way Handshake violets the freshness properties as guaranteed by IEEE 802.11i, due to which other types of attacks are possible. This is because, every time same nonce is used so, an attacker can eavesdrop and identify the nonce because it has sufficient time to analyze it. In order to remove this drawback, this study presents the solution which maintains a list for storing messages (Message 1) in supplicant until Message 3 is received. If the number of stored message in the list exceed than total size of the list, it will randomly drop one message from the list. When valid Message 3 is received, it will drop all the items stored in the list and install the key corresponding to the valid message. This prevents DoS vulnerability by maintaining freshness property as guaranteed by IEEE 802.11i.

As the future work, it is aimed to analyze other higher level protocols such as IKEv2, IEEE 802.11e, IEEE 802.11r, Kerberos and Mobile IPv6 by using PCL. Also it is aimed to develop for incremental protocol construction i.e. starting from the simple components and extending them by applying a sequence of protocol transformation operations.

For proving security properties by using PCL, a practical tool has not been developed yet. So, it is also aimed to develop a practical tool for implementing PCL.

REFERENCES

[1] Wi-Fi Allance: Wi-Fi Protected Access. Version 2.0, April 2003. Available through http://www.wi-fi.org/

[2] IEEE standard 802.11i™-2004. IEEE standard for Information Technology – Telecommunication and Information Exchange between systems – Local and Metropolitan Area Networks – Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Medium Access Control (MAC) Security Enhancements, 2004. Available at http://www.ieee.org

[3] Changhua He, John C Mitchell. "*Analysis of The 802.11i 4-Way Handshake*", Electrical Engineering and Computer Science Departments, Stanford University.

[4] Changhua He, John C Mitchell. "*Security Analysis and Improvements for IEEE 802.11i*", Electrical Engineering and Computer Science Departments, Stanford University.

[5] Magnus Falk. "*Fast and Secure Roaming in WLAN*", Linkoping University.

[6] Nikita Borisov, Ian Goldberg, David Wagner. "*Intercepting Mobile Communications: The Insecurity of 802.11*", Seventh Annual International Conference on Mobile Computing And Networking, July, 16-21,2001.

[7] Guillaume Lehembre (2005), "*Wi-Fi Security-WEP,WPA and WPA2*". Available at http://www.hakin9.org

[8] Scott Fluhrer, Itsik Mantin and Adi Shamir. "*Weakness in Key Scheduling Algorithm in RC4*". Available at http://www.drizzle.com/~aboba/IEEE/rc4_ksaproc.pdf

[9] William A. Arbaugh, Narendar Shankar And Y.C. Justin Wan. "*Your 802.11 Wireless Network has No Clothes*", Department of Computer Science, University of Maryland, March 30, 2001.

[10] Bernard Aboba, Larry J. Blunk, John Vollbrecht, James Carlson, and Henrik Levkowetz. "*Extensible Authentication Protocol*", June 2004. Available at http://www.ietf.org/rfc/rfc3748.txt

[11] P. Congdon, B. Aboba, A. Smith, G. Zorn and J. Roese. "*IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guide*". Available at http://www.ietf.org/rfc/rfc3580.txt

[12] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege and D. Spence. "*AAA Authorization Framework*". Available at http://www.ietf.org/rfc/rfc2904.txt

[13] Kamat, Narendra, Lee, Harold, Li, Bo, Menchaca and Daniel. "*Evolution of Wireless LAN Security Standards*". Available at http://users.ece.utexas.edu/~wireless/EE381K11_Spring03/projects/8.3.pdf

[14] IEEE Standard 802.1X-2001, "*IEEE Standard for Local and Metropolitan Area Networks – Port-Based Network Access Control*", June 2001. Available at http://www.ieee.org

[15] Bernard Aboba, "*IEEE 802.1X Pre-Authentication*", Microsoft Corporation, June 17, 2002.

[16] C. Rigney, "*RADIUS Accounting*". Available at http://www.ietf.org/rfc/rfc2866.txt

[17] L. Blunk and J. Vollbrecht, "*PPP Extensible Authentication Protocol (EAP)*". Available at http://www.ietf.org/rfc/rfc2284.txt

[18] C. Rigney, S. Willens, A. Rubens and W. Simpson, "*Remote Authentication Dial-In User Service (RADIUS)*". Available at http://www.ietf.org/rfc/rfc2865.txt

[19] Arunesh Mishra, William A. Arbaugh, "*An Initial Security Analysis of The IEEE 802.1X Standard*", Department of Computer Science, University of Maryland, Feb 6, 2002.

[20] 3e Technology International, "*FIPS 140-2, IEEE 802.11i and WPA2 For Security, Interoperability and Affordability"*.

[21] Jesse Walker, "*IEEE 802.11i Standard Improves Wireless LAN Security*", Communication Technology Lab, Intel Corporation, 2005.

[22] National Institute of Standards And Technology (NIST), "*Wireless Network Security, 802.11, Bluetooth and Handheld Devices*".

[23] Andrea Bittau, "*The Fragmentation Attack in Practice*", University College London, September 17, 2005.

[24] Wi-Fi Allance, "*Securing Wi-Fi Wireless Networks with Today's Technologies*", February 6, 2003. Available through http://www.wi-fi.org/

[25] Joshua Hill, InfoGard Laboratories, "*An Analysis of the RADIUS Authentication Protocol*", 2001. Available at http://www.untruth.org/~josh/security/radius/

[26] Xinliang Zheng, Chuming Chen, Chin-Tser Hung, Manton M. Matthews, Naveen Santhapuri, "*A Dual Authentication Protocol for IEEE 802.11 Wireless LANs*", Department of Computer Science And Engineering, University of South Carolina.

[27] Bernard Aboba and Pat R. Calhoun, "*RADIUS (Remote Authentication Dial-In User Service) Support For Extensible Authentication Protocol (EAP)*", Department of Computer Science And Engineering, University of South Carolina, September 2003. Available at http://www.ietf.org/rfc/rfc3579.txt

[28] Anupam Datta, "*Security Analysis of Network Protocols: Compositional Reasoning And Complexity-Theoretic Foundations*", Stanford University PhD thesis, September 2005.

[29] Bernard Aboba and D. Simon, "*PPP EAP TLS authentication protocol*", RFC 2716, October, 1999. Available at http://www.ietf.org/rfc/rfc2716.txt

[30] http://new.remote-exploit.org/images/5/5a/Cowpatty-2.0.tar.gz, "*Cowpatty WPA-PSK Cracking tools"*.

[31] Christophe Devine, "*Aircrack, the wireless cracking tools*". Available at http://www.cr0.net:8040/code/network/aircrack/

[32] Robert Moskowitz, "*WPA-PSK weakness*". Available at http://www.Wi-Finetnews.com/archieves/002452.html

[33] Dazhi Chen, Jing Deng and Pramod K. Varshney, "*Protecting Wireless Network Against a Denial of Service Attack Based on Virtual Jamming*", EECS Department, Syracuse University, Syracuse, NY.

[34] Matthew J. Braun, "*A Method for Mitigating Denial of Service Attacks on Differentiated Services Networks*", Naval Postgraduate School, Monterey, California, September 2002.

[35] Donna M. Gregg, William J. Blackert, David V. Heinbuch and Donna C. Furnanage, "*Analyzing Denial of Service Attack Using Theory and Modeling and*

Simulation". Available at http://www.itoc.usma.edu/Workshop/2001/Authors/Submitted_Abstracts/paperW1A1(27).pdf

[36] Joan Daemen and Vincent Rijmen, "*The Rijndael Block Cipher*", AES Proposal, 1999.

[37] Anupam Datta, Ante Derek, John C. Mitchell, "*Secure Protocol Composition*", Computer Science Department, Stanford University.

[38] Anupam Datta, Ante Derek, John C. Mitchell and Dusko Pavlovic, "*A derivation for security protocols and its logical formalization*", Computer Science Department, Stanford University.

[39] Changhua He, Mukund Sundararajan, Anupam Dattam Ante Derek and John C. Mitchell, "*A Modular Correctness Proof of IEEE 802.11i and TLS*", Electrical Engineering and Computer Science Department, Stanford University.

[40] Anupam Datta, Ante Derek, John C. Mitchell and Arnab Roy, "*Protocol Composition Logic (PCL)*", Computer Science Department, Stanford University.

[41] Nancy Durgin, John C. Mitchell and Dusko Pavlovic, "*A Compositional Logic For Protocol Correctness*", Computer Science Department, Stanford University.

[42] Nancy Durgin, John C. Mitchell and Dusko Pavlovic, "*A compositional logics for proving security properties of protocols*", Computer Science Department, Stanford University.

[43] Jerome Maye and Gregory Theoduloz, "*Implementation of Π-Calculus*", 1st June 2005.

[44] Jeannette M. Wing, "*FAQ on Π-Calculus*", Carnegie Mellon University, 27th December 2002.

[45] T. Dierks and C. Allen, "*The TLS Protocol*", Carnegie Mellon University, 27th December 2002. Available at http://www.ietf.org/rfc/rfc2246.txt

[46] Ke Liu, "*Understanding the Implementation of IEEE MAC 802.11 standard in NS-2*". Available at http://www.cs.binghamton.edu/~kliu/research/ns2code/note.pdf

[47] Yue Wang, MobiTec Lab, CUHK, "*A Tutorial of 802.11 Implementation on NS-2*". Available at http://www.winlab.rutgers.edu/~zhibinwu/pdf/tr_ns802_11.pdf

[48] Paul Meeneghan and Declan Delaney, "*An Introduction to NS, Nam and OTcl Scripting*", National University of Ireland, Maynooth, Co. Kildare, Ireland. Available at http://www.cs.nuim.ie/research/reports/2004/nuim-cs-tr-2004-05.pdf

[49] "*The ns Manual*", Available at http://www.isi.edu/nsnam/ns/ns-documentation.html

[50] "*XGRAPH, The General Purpose 2D Plotter*", http://www.xgraph.org

APPENDIX A

This section presents ns2 (*Network Simulator 2*) detail of DoS attack, namely 4-Way Handshake Blocking, and its solution by using random drop policy. The simulation result was analyzed by using xgraph and graphically the simulation was visualized by using NAM (*Network Animator*).

A.1 Physical Layer Implementation

Channel (channel.cc)
The function of class channel is to deliver packets from a wireless node to its neighbors within the sensing range.

NetIF (wireless-phy.cc)
The function of class WirelessPhy is to send packets to the channel and receive packet from channel.

A.2 MAC Implementation

MAC (mac-802_11.cc)
There are two functions of class Mac802_11. On sending, it uses *CSMA/CA* medium access mechanism and on receiving, it uses SIRT (*SIR Threshold; SIR - Susceptible Infectious Removal*) based reception (Capture).

CSMA/CA
For outgoing packets, it will call send method which is the entry point of CSMA/CA.

```
void send(Packet *p, Handler *h)
{
        ..............................
```

```
        if(mhBackoff_.busy() == 0)
        {
                if(is_idle())
                {
                        if (mhDefer_.busy() == 0)
                        {
                                /*If we are already deferring, there is no need to
                                reset the Defer timer.*/
                                rTime = (Random::random() % cw_)
                                * (phymib_.getSlotTime());
                                mhDefer_.start(phymib_.getDIFS() + rTime);
                        }
                }
                else
                {
                        /*If the medium is NOT IDLE, then we start
                        the backoff timer.*/
                        mhBackoff_.start(cw_, is_idle());
                }
        }
}
```

Capture Model

The capture model is simple; when multiple packets collide at the receiver, only the first packet will be successfully received, if Rx power is larger than any of the other packets by at lest CPThresh (i.e. 10dB).

```
void recv(Packet *p, Handler *h)
{
        .....
        //Handle incoming packets
```

```
        //When there is no packet reception, log receiving p at pktRx_
        if(rx_state_ == MAC_IDLE)
        {
                setRxState(MAC_RECV);
                pktRx_ = p;
                mhRecv_.start(txtime(p));// schedule the reception of this packet in txtime
                //setRxState(MAC_IDLE) again after reception.
        }
        /*When there is already a packet reception (in pktRx_), calculate the inference*/
        else
        {
                //Simplified SIR calculation (Comparison of two signals)
                if(pktRx_->txinfo_.RxPr / p->txinfo_.RxPr >= p->txinfo_.CPThresh)
                {
                        capture(p);//pktRx_ can be correctly received;
                        //recalculate when the channel will be idle
                }
                else
                {
                        collision(p);//stop receving pktRx_ (i.e. mhRecv.stop() )
                        //recalculate when the channel will be idle
                }
        }
}
```

A.3 OTcl Coding

DoS_4_Way.tcl

```
Mac/Simple set bandwidth_ 1Mb

set MESSAGE_PORT 42
```

```
set BROADCAST_ADDR -1

#---------------------------------------------------------------------
# we use 5 nodes. 1 AP, 3 STN and 1 attacker
# Node 1 : AP
# Node 2 - 4 : STN
# Node 5 : Attacker
#---------------------------------------------------------------------
set num_nodes 5

set val(chan)           Channel/WirelessChannel    ;#Channel Type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)          Phy/WirelessPhy            ;# network interface type

set val(mac)            Mac/802_11                 ;# MAC type

set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         50                         ;# max packet in ifq

# DumbAgent, AODV, and DSDV work.  DSR is broken
set val(rp) DumbAgent
#set val(rp)            DSDV
#set val(rp)            DSR
#set val(rp)                AODV

# size of the topography
set val(x)              500
set val(y)              500
```

```
set authenticated_nodes ""
set STN_nonce_counter {0}

#Open the output files
set f1 [open node1.tr w]
set f2 [open node2.tr w]
set f3 [open node3.tr w]
set fa [open attacker1.tr w]

set ns [new Simulator]

set f [open DoS_4_Way.tr w]
$ns trace-all $f
set nf [open DoS_4_Way.nam w]
$ns namtrace-all-wireless $nf $val(x) $val(y)

$ns use-newtrace

# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

# Create God
create-god $num_nodes

set chan_1_ [new $val(chan)]

$ns node-config -adhocRouting $val(rp) \
                -llType $val(ll) \
                -macType $val(mac) \
```

```
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propType $val(prop) \
            -phyType $val(netif) \
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace OFF \
            -macTrace ON \
            -movementTrace OFF \
            -channel $chan_1_

# subclass Agent/MessagePassing to make it do flooding
Class Agent/MessagePassing/Flooding -superclass Agent/MessagePassing

Agent/MessagePassing/Flooding instproc recv {source sport size data} {
    $self instvar messages_seen node_
    global ns BROADCAST_ADDR authenticated_nodes f1 f2 f3 fa

    #---------------------------------------------------
    # extract message ID from message
    #---------------------------------------------------
    set message_id [lindex [split $data ":"] 0]
    set message [lindex [split $data ":"] 1]

    #---------------------------------------------------
    #buffer size of the STN
    #---------------------------------------------------
    set buffer_size 9

    #---------------------------------------------------
```

```
# Process message according to the node
#-----------------------------------------------------
if {[$node_ node-addr] == 0} {
	#-----------------------------------------------------
	# AP
	# check whether the received message is MSG2 or MSG4
	# other MSG will be negeleted
	#-----------------------------------------------------
	if {$message_id == 2 } {
		#-----------------------------------------------------
		# MSG2 is received
		# so we send MSG3
		#-----------------------------------------------------
		puts "\nMSG2 is Received By Node [$node_ node-addr]"
		# check the validity of MSG2
		if {[checkMSG2Validity $message] == 1} {
			puts "\nMSG3 is Send From Node [$node_ node-addr]"
			$self sendto $size "3:$message" $BROADCAST_ADDR $sport
		} else {
			puts "\nInvalid MSG2"
		}
	} elseif {$message_id == 4} {
		lappend authenticated_nodes $message
		puts "\nNode $message is authenticated"
	}
} elseif {[$node_ node-addr] == 4} {
	#-----------------------------------------------------
	# Attacker.
	#-----------------------------------------------------
	if {$message_id >1 && $message_id <4} {
		set now [$ns now]
```

```
                puts $fa "$now [getNonceNumber $message]"
        }
} else {
        #-----------------------------------------------------
        # STN
        # check for memory
        #-----------------------------------------------------
        $self instvar msgCounter
        if {$msgCounter > $buffer_size} {
                #-----------------------------------------------------
                # Memory is not sufficient
                # So, randomly drop the stored message
                #-----------------------------------------------------
                set randomMSG [generateRandomNo 1 $buffer_size]
                set MSGs [split $messages_seen " "]
                set messages_seen ""
                for {set i 0} {$i < [llength $MSGs]} {incr i} {
                        if { $i !=$randomMSG} {
                                lappend messages_seen [lindex $MSGs $i]
                        }
                }
                set msgCounter [expr ($msgCounter-1)]
        }
        # Record for graph
        $self instvar packet_processed
        incr packet_processed
        set now [$ns now]
        if {[$node_ node-addr]==1} {
                puts $f1 "$now $packet_processed"
        } elseif {[$node_ node-addr]==2} {
                puts $f2 "$now $packet_processed"
```

```
} else {
        puts $f3 "$now $packet_processed"
}
#-----------------------------------------------------
# check for message
# only MSG1 and MSG3 will be proceed
#-----------------------------------------------------
if {$message_id == 1 } {
        #-----------------------------------------------------
        # check weather the node has been authenticated or not
        #-----------------------------------------------------
        if {[lsearch $authenticated_nodes [$node_ node-addr]] == -1} {
                #-----------------------------------------------------
                # Record for graph
                #-----------------------------------------------------
                #$self instvar packet_processed
                #incr packet_processed
                #set now [$ns now]
                #if {[$node_ node-addr]==1} {
                #       puts $f1 "$now $packet_processed"
                #} elseif {[$node_ node-addr]==2} {
                #       puts $f2 "$now $packet_processed"
                #} else {
                #       puts $f3 "$now $packet_processed"
                #}
                puts "\nMSG$message_id Received By Node [$node_ node-addr]"
                #-----------------------------------------------------
                # check the existance of the nonce in the already stored list
                #-----------------------------------------------------
                if {[lsearch $messages_seen $message] == -1} {
                        # nonce has not been stored yet
```

```
                        incr msgCounter
                        lappend messages_seen $message
                    }
                    # send MSG2 to AP
                    puts "\nMSG2 Send From Node [$node_ node-addr]"
                    $self sendto $size "2:[$node_ node-
addr];$message;[createFreshNonceSTN]" $BROADCAST_ADDR $sport
                }
        } elseif {$message_id == 3 } {
                #-----------------------------------------------------
                # check if the valid message
                #-----------------------------------------------------
                if {[lindex [split $message ";"] 0] == [$node_ node-addr]} {
                    puts "\nMSG$message_id Received By Node [$node_ node-addr]"
                    #-----------------------------------------------------
                    # Remove MSG1 from the list if exit otherwise we neglete it
                    #-----------------------------------------------------
                    set RSNIE [lindex [split $message ";"] 1]
                    if {[lsearch $messages_seen $RSNIE] != -1} {
                        set MSGs [split $messages_seen " "]
                        set messages_seen ""
                        for {set i 0} {$i < [llength $MSGs]} {incr i} {
                            if {[lindex $MSGs $i] != $RSNIE} {
                                lappend messages_seen [lindex $MSGs $i]
                            }
                        }
                    }
                    puts "\nMSG4 Send From Node [$node_ node-addr]"
                    $self sendto $size "4:[$node_ node-addr]"
$BROADCAST_ADDR $sport
                }
```

```
            }
        }
    }

#-----------------------------------------------------
# procedure for generating random number
#-----------------------------------------------------
proc generateRandomNo {min max} {
    set sizeRNG [new RNG]
    $sizeRNG next-substream
    set size_ [new RandomVariable/Uniform]
    $size_ set min_ $min
    $size_ set max_ $max
    $size_ use-rng $sizeRNG
    return [expr round([$size_ value])]
}

#-----------------------------------------------------------------------
# procedure for generating fresh nonce for an STN
#-----------------------------------------------------------------------
proc createFreshNonceSTN {} {
        global STN_nonce_counter
        incr STN_nonce_counter
        return STNn$STN_nonce_counter
}

proc checkMSG2Validity {msg} {
        set dat [lindex [split $msg ";"] 1]
        if {$dat == "RSNIE_AP"} {
                return 1
        } else {
```

```
                return -1
        }
}

proc getNonceNumber {msg} {
        set nonce [lindex [split $msg ";"] 2]
        set nonceNo [lindex [split $nonce "n"] 1]
        return $nonceNo
}

Agent/MessagePassing/Flooding instproc send_message {size message_id data port} {
   $self instvar messages_seen node_
   global ns MESSAGE_PORT BROADCAST_ADDR

   #$ns trace-annotate "[$node_ node-addr] sending message $message_id"
   $self sendto $size "$message_id:$data" $BROADCAST_ADDR $port
}

#---------------------------------------------------------------------
# create a bunch of nodes
#---------------------------------------------------------------------
for {set i 0} {$i < $num_nodes} {incr i} {
   set n($i) [$ns node]
   $n($i) set X_ [generateRandomNo 100 300]
   $n($i) set Y_ [generateRandomNo 100 300]
   $n($i) set Z_ 0.0
   $ns initial_node_pos $n($i) 20
}

#---------------------------------------------------------------------
```

```
# attach a new Agent/MessagePassing/Flooding to each node on port
$MESSAGE_PORT
#---------------------------------------------------------------------
for {set i 0} {$i < $num_nodes} {incr i} {
   set a($i) [new Agent/MessagePassing/Flooding]
   $n($i) attach  $a($i) $MESSAGE_PORT
   $a($i) set messages_seen {}
   $a($i) set msgCounter 0
   $a($i) set packet_processed {0}
}

#---------------------------------------------------------------------
# now set up some events
#---------------------------------------------------------------------
# AP send MSG1 with its RSN IE in every 0.2 second
# Here we assumed that the nonce only consists of RSN IE
#---------------------------------------------------------------------
$ns at 0.1 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
$ns at 0.2 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
$ns at 0.3 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
$ns at 0.4 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
$ns at 0.5 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
$ns at 0.6 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
$ns at 0.7 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
$ns at 0.8 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
$ns at 0.9 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
$ns at 0.10 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
#$ns at 0.11 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
#$ns at 0.12 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
#$ns at 0.13 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
#$ns at 0.14 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
```

```
#$ns at 0.15 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
#$ns at 0.16 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
#$ns at 0.17 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
#$ns at 0.18 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
#$ns at 0.19 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
#$ns at 0.20 "$a(0) send_message 200 1 {RSNIE_AP}  $MESSAGE_PORT"
#---------------------------------------------------------------------
# simulate the existance of the attacker
# attacker also sends message in every 0.2 seconds but it start sending
# after 0.3 seconds of the simulation time
#---------------------------------------------------------------------
$ns at 0.2 "$a(4) send_message 200 1 {RSNIE_1}  $MESSAGE_PORT"
$ns at 0.3 "$a(4) send_message 200 1 {RSNIE_2}  $MESSAGE_PORT"
$ns at 0.4 "$a(4) send_message 200 1 {RSNIE_3}  $MESSAGE_PORT"
$ns at 0.5 "$a(4) send_message 200 1 {RSNIE_4}  $MESSAGE_PORT"
$ns at 0.6 "$a(4) send_message 200 1 {RSNIE_5}  $MESSAGE_PORT"
$ns at 0.7 "$a(4) send_message 200 1 {RSNIE_6}  $MESSAGE_PORT"
$ns at 0.8 "$a(4) send_message 200 1 {RSNIE_7}  $MESSAGE_PORT"
$ns at 0.9 "$a(4) send_message 200 1 {RSNIE_8}  $MESSAGE_PORT"
$ns at 0.10 "$a(4) send_message 200 1 {RSNIE_9}  $MESSAGE_PORT"
$ns at 0.11 "$a(4) send_message 200 1 {RSNIE_10}  $MESSAGE_PORT"
#$ns at 0.12 "$a(4) send_message 200 1 {RSNIE_1}  $MESSAGE_PORT"
#$ns at 0.13 "$a(4) send_message 200 1 {RSNIE_2}  $MESSAGE_PORT"
#$ns at 0.14 "$a(4) send_message 200 1 {RSNIE_3}  $MESSAGE_PORT"
#$ns at 0.15 "$a(4) send_message 200 1 {RSNIE_4}  $MESSAGE_PORT"
#$ns at 0.16 "$a(4) send_message 200 1 {RSNIE_5}  $MESSAGE_PORT"
#$ns at 0.17 "$a(4) send_message 200 1 {RSNIE_6}  $MESSAGE_PORT"
#$ns at 0.18 "$a(4) send_message 200 1 {RSNIE_7}  $MESSAGE_PORT"
#$ns at 0.19 "$a(4) send_message 200 1 {RSNIE_8}  $MESSAGE_PORT"
#$ns at 0.20 "$a(4) send_message 200 1 {RSNIE_9}  $MESSAGE_PORT"
#$ns at 0.21 "$a(4) send_message 200 1 {RSNIE_10}  $MESSAGE_PORT"
```

```
$ns at 10.0 "finish"

proc finish {} {
        global ns f nf val f1 f2 f3 fa
        $ns flush-trace
        close $f
        close $nf
        close $f1
        close $f2
        close $f3
        close $fa
    exit 0
}

$ns run
```

A.4 Result of Running OTcl code

```
MSG1 is Received By Node 2
MSG2 is Send From Node 2
MSG1 is Received By Node 1
MSG2 is Send From Node 1
MSG1 is Received By Node 3
MSG2 is Send From Node 3
MSG2 is Received By Node 0
Invalid MSG2
MSG2 is Received By Node 0
Invalid MSG2
MSG2 is Received By Node 0
Invalid MSG2
```

MSG1 is Received By Node 2
MSG2 is Send From Node 2
MSG1 is Received By Node 1
MSG2 is Send From Node 1
MSG1 is Received By Node 3
MSG2 is Send From Node 3
MSG2 is Received By Node 0
MSG3 is Send From Node 0
MSG1 is Received By Node 2
MSG2 is Send From Node 2
MSG1 is Received By Node 1
MSG2 is Send From Node 1
MSG1 is Received By Node 3
MSG2 is Send From Node 3
MSG2 is Received By Node 0
MSG3 is Send From Node 0
MSG2 is Received By Node 0
MSG3 is Send From Node 0
MSG2 is Received By Node 0
Invalid MSG2
MSG3 is Received By Node 3
MSG4 is Send From Node 3
Node 3 is authenticated
MSG2 is Received By Node 0
Invalid MSG2
MSG3 is Received By Node 1
MSG4 is Send From Node 1
MSG3 is Received By Node 2
MSG4 is Send From Node 2
Node 1 is authenticated
Node 2 is authenticated

A.5 Screen Shot of NAM

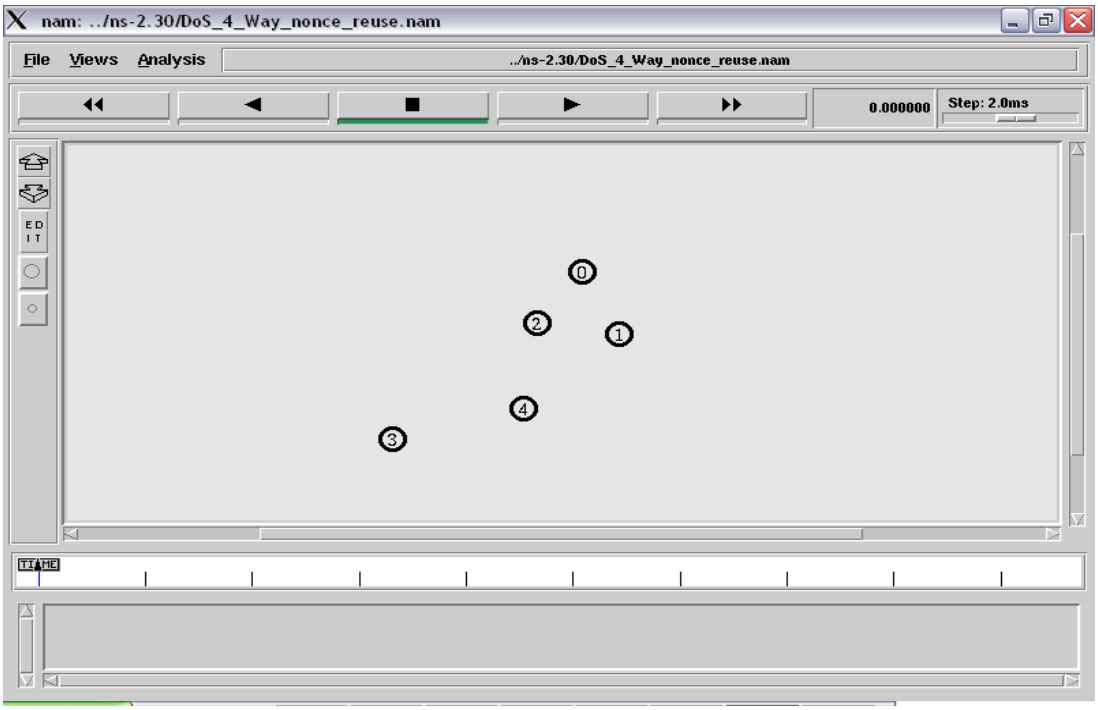


Figure A.1 Placement of Nodes in NAM (Nonce Reuse)

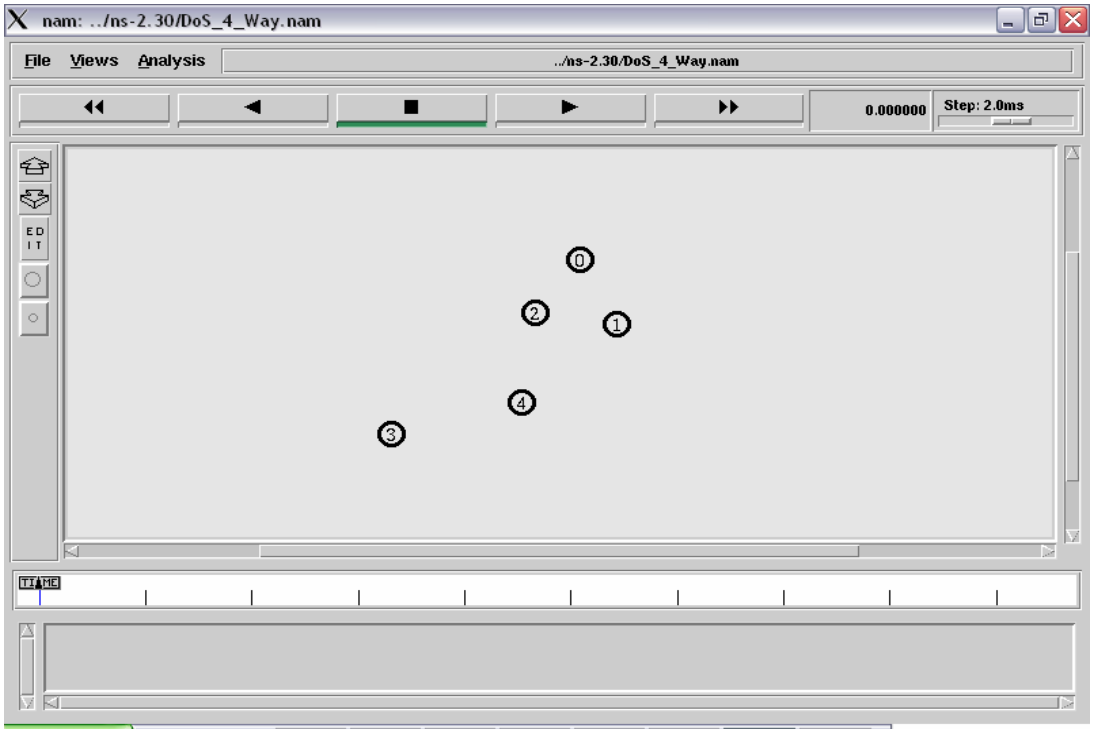


Figure A.2 Placement of Nodes in NAM (Without Nonce Reuse)

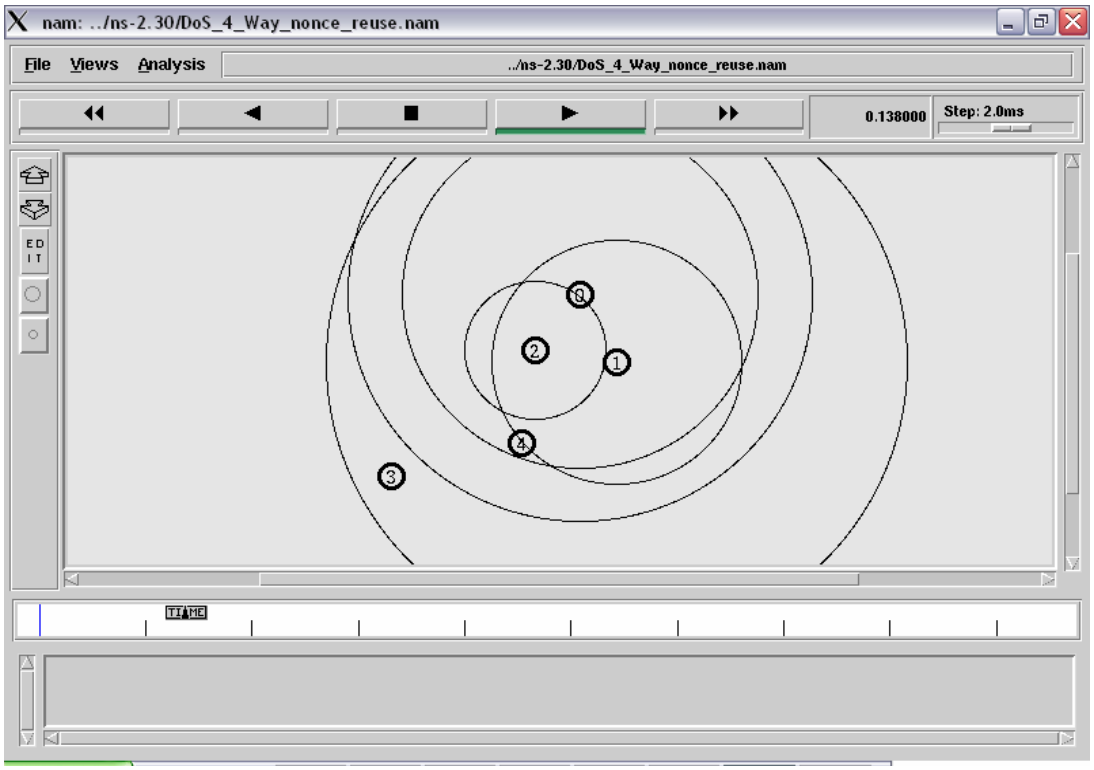


Figure A.3 NAM Simulation 1

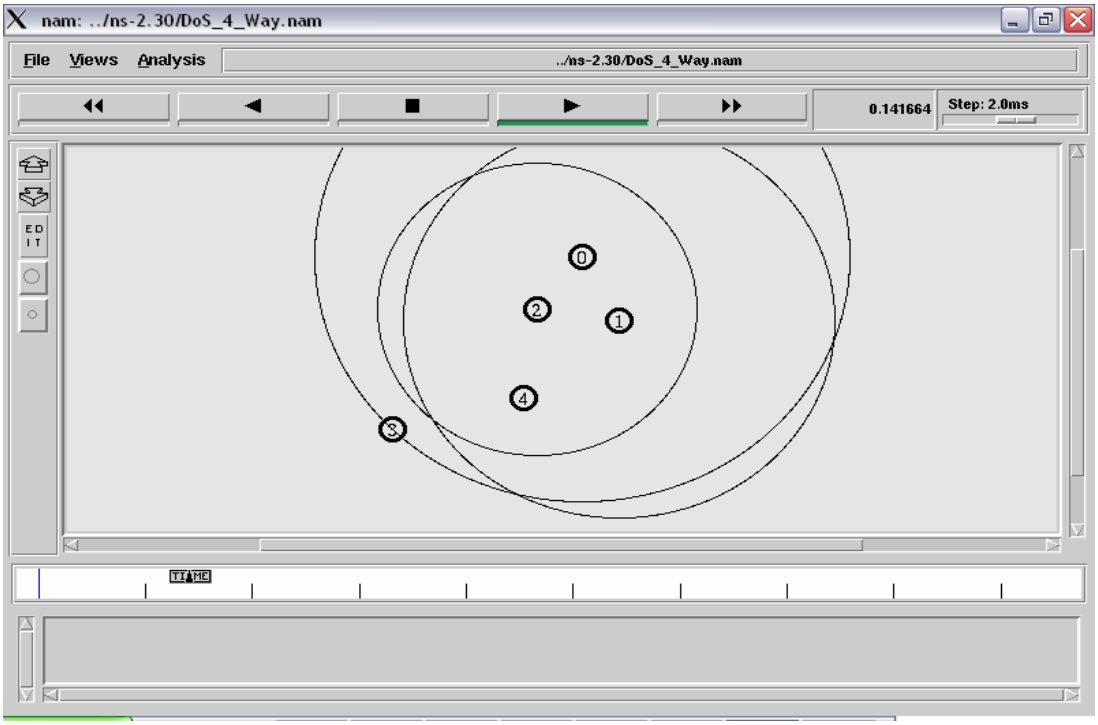


Figure A.4 NAM Simulation 2

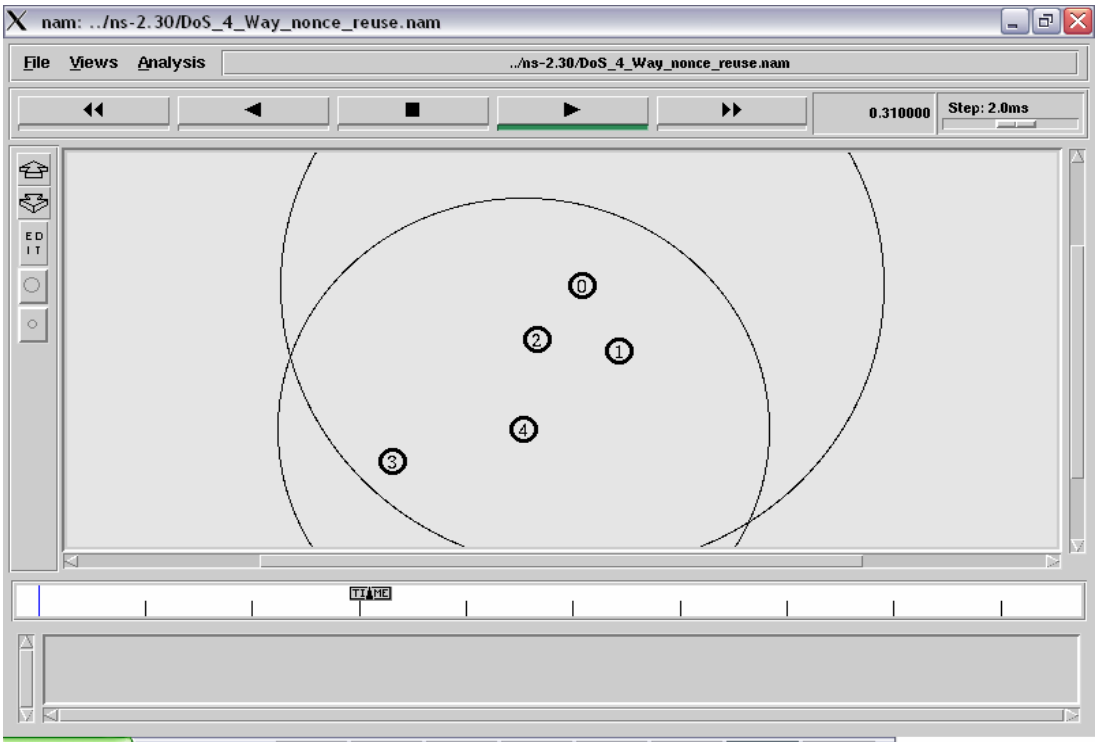


Figure A.5 NAM Simulation 3

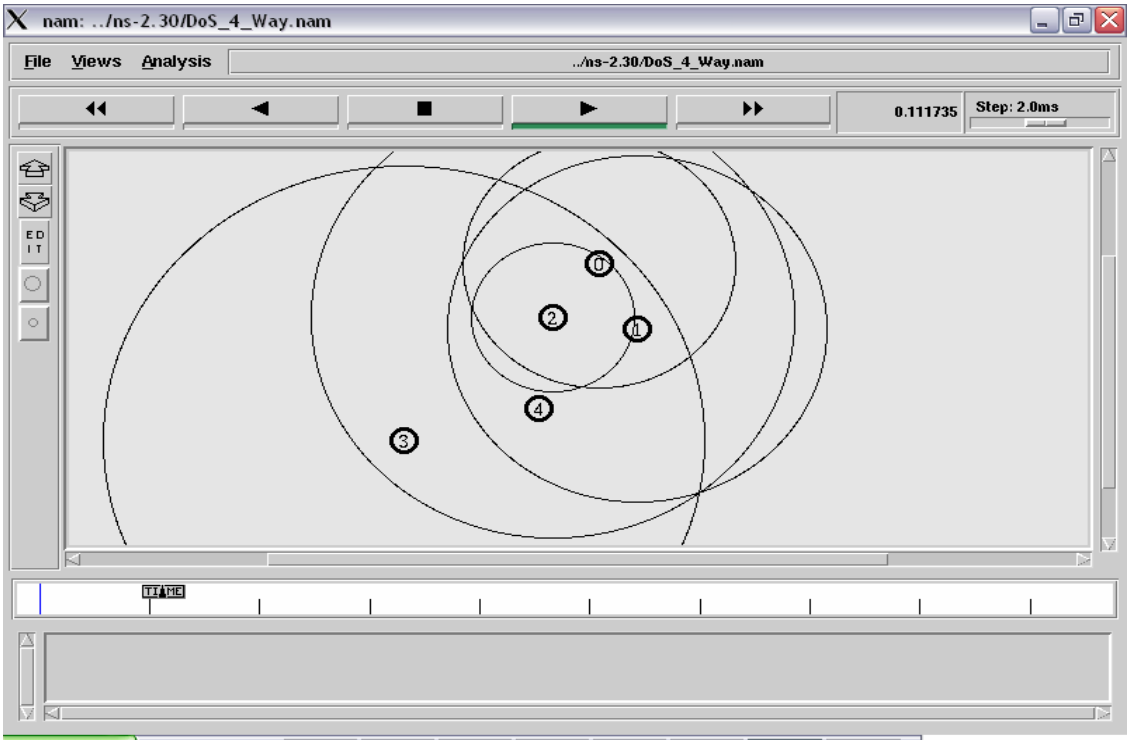


Figure A.5 NAM Simulation 4