2007

Developing Strand Space Based Models And Proving The Correctness Of The Ieee 802.11i Authentication Protocol With Restricted Sec

Zeeshan Furqan
University of Central Florida

DEVELOPING STRAND SPACE BASED MODELS AND PROVING THE CORRECTNESS OF THE IEEE 802.11I AUTHENTICATION PROTOCOL WITH RESTRICTED SECURITY OBJECTIVES

by

ZEESHAN FURQAN
B.S. Wichita State University., 2001
M.S. University of Central Florida., 2004

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the School of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2007

Major Professor: Ratan K. Guha

ABSTRACT

The security objectives enforce the security policy, which defines what is to be protected in a network environment. The violation of these security objectives induces security threats. We introduce an explicit notion of security objectives for a security protocol. This notion should precede the formal verification process. In the absence of such a notion, the security protocol may be proven correct despite the fact that it is not equipped to defend against all potential threats. In order to establish the correctness of security objectives, we present a formal model that provides basis for the formal verification of security protocols.

We also develop the modal logic, proof based, and multi-agent approaches using the Strand Space framework. In our modal logic approach, we present the logical constructs to model a protocol's behavior in such a way that the participants can verify different security parameters by looking at their own run of the protocol. In our proof based model, we present a generic set of proofs to establish the correctness of a security protocol. We model the 802.11i protocol into our proof based system and then perform the formal verification of the authentication property. The intruder in our model is imbued with powerful capabilities and repercussions to possible attacks are evaluated. Our analysis proves that the authentication of 802.11i is not compromised in the presented model. We further demonstrate how changes in our model will yield a successful man-in-the-middle attack.

Our multi-agent approach includes an explicit notion of multi-agent, which was missing in the Strand Space framework. The limitation of Strand Space framework is the assumption that all the information available to a principal is either supplied initially or is contained in messages received

by that principal. However, other important information may also be available to a principal in a security setting, such as a principal may combine information from different roles played by him in a protocol to launch a powerful attack. Our presented approach models the behavior of a distributed system as a multi-agent system. The presented model captures the combined information, the formal model of knowledge, and the belief of agents over time. After building this formal model, we present a formal proof of authentication of the 4-way handshake of the 802.11i protocol.

To my parents and my fiancée

ACKNOWLEDGEMENTS

I would like to thank Allah for blessing me with courage and resources to pursue my objectives. I would like to acknowledge my mother Shama Furqan and my father Muhammad Furqan for their love, support, sacrifice, and prayers throughout this work. My parents have always considered my education as their first priority and have always provided me with the best. It is through their efforts, support, encouragement, and prayers that I have made it through all the tests to reach this point in life and I could not have done it without them. I would also like to acknowledge my fiancée Sonia and her family (especially her mom Nuzhat) for their continued love, prayers, and delicious meals. They have always been there supporting me and always understood the importance of my work. Sonia is an important part of my life and I thank her for never losing trust in me. Together, my parents and my fiancée have made valuable contributions and it is to them that I dedicate this dissertation.

I am deeply indebted to Professor Ratan K. Guha who has been my advisor throughout this work and is the chair of my dissertation committee. His contributions, detailed comments, and insights have been of great value to me. I owe him lots of gratitude for showing me this methodology of research. He has kept an eye on the progress of my work and has always given me valuable advices. I would also like to thank the other members of my dissertation committee, Professor Mostafa Bassiouni, Dr. Joohan Lee, and Dr. Mainak Chatterjee, who monitored my work and took effort in reading this thesis.

I would also like to mention the contributions of my colleagues at Network and Security lab at UCF. I wish them all success and a great career in their research areas. In particular, I would

like to acknowledge Shahabuddin Muhammad who has worked with me on the same projects. Shahabuddin and I have spent countless nights working on the formal verification, 802.11i, Strand Space, and other research areas. His contributions, support, advices, and company have been very significant. I would like to thank him for his help, trust, kindness, and sharing many experiences and thoughts with me throughout the last five years.

During my time as a student, I have been blessed to have been surrounded by many wonderful friends. Every one of them contributed in one way or another. Some were responsible for giving me a push in the right direction in life, whereas others helped me out when I needed them, made delicious meals for me, or simply cheered me up with their sense of humor. I am thankful to all of them and wish them success and happiness. I would also like to extend my appreciations for all of my family members and thank them for their support, faith, and prayers.

In the end, I would like to acknowledge my funding agencies. This work was partially supported by NSF under Grant EIA 0086251 and ARO under grant DAAD19-01-1-0502. The views and conclusions herein are those of the author and do not represent the official policies of the funding agencies or the University of Central Florida.

TABLE OF CONTENTS

LIST OF FIGURES xiii

LIST OF TABLES xiv

CHAPTER 1 INTRODUCTION 1

1.1 Problem 1

1.1.1 Formal Verification 2

1.2 Our Contributions 2

1.2.1 Explicit Notion of Security Objectives 4

1.2.2 Formal Verification of 802.11i 5

1.2.3 A Multi-agent Framework for Security protocol Verification 6

1.3 Organization of this Thesis 7

CHAPTER 2 FORMAL APPROACHES AND SECURITY OBJECTIVES 9

2.1 Logic Based Approaches 9

2.1.1 Rangan's approach 10

2.1.2 BAN Logic 12

2.1.3 Moser's Approach 16

2.1.4 CKT5 and KPL 17

2.1.5 GNY Logic 19

2.1.6 Coffey and Saidha's Approach 21

2.1.7 Other Logic Based Approaches 22
2.2 Theorem Proving 24
2.2.1 Paulson's Inductive Approach 24
2.2.2 Other Theorem Proving Approaches 26
2.3 Model Checking 27
2.3.1 Longley-Rigby Tool 28
2.3.2 Interrogator 29
2.3.3 NRL Protocol Analyzer 30
2.3.4 FDR 32
2.3.5 MurΦ 36
2.3.6 Brutus 38
2.3.7 Athena 40
2.4 Other Approaches 42
2.4.1 Dolev and Yao 42
2.4.2 Kemmerer's Work 44
2.4.3 Type Checking 45
2.4.4 Strand Space Model 46
2.5 Summary 48

CHAPTER 3 SECURITY OBJECTIVES IN WLANS 50
3.1 Security Policy and Security Objectives 51
3.1.1 Importance of using Security Objectives 51

3.2 Security objectives in 802.11 52
3.2.1 Authentication and Access Control 54
3.2.2 Confidentiality 56
3.2.3 Integrity 58
3.2.4 Availability 59
3.3 Security Objectives in 802.11i 60
3.3.1 Authentication 60
3.3.2 Key Management 62
3.3.3 Confidentiality and Integrity 64
3.3.4 Availability 66
3.4 Summary 68

CHAPTER 4 DEVELOPING FORMAL MODELS FOR SECURITY PROTOCOLS 69

4.1 A Modal Logic Approach Based on SSM 69
4.1.1 Syntax of the Language 71
4.1.2 Propositional Constructs 72
4.1.3 Model of Computation 74
4.1.4 Semantics 75
4.1.5 The Proposed Logic 78
4.1.6 Soundness 80
4.2 Formal Model of Security Objectives for 802.11i 82

4.3 Building Lemmas 84

4.4 Summary 89

CHAPTER 5 FORMAL VERIFICATION OF 802.11i 91

5.1 Authentication in 802.11i 91

5.1.1 Discovery 92

5.1.2 Open System and EAP-802.1X Authentication 92

5.1.3 Key Generation and Distribution 95

5.1.4 Secure Data Communication 98

5.1.5 Connection Termination 98

5.2 Modelling 802.11i using Strand Spaces 98

5.2.1 Modelling EAP-802.1X Authentication 98

5.2.2 Modelling the 4-way Handshake 100

5.3 Authentication Proofs 100

5.3.1 Assumptions 102

5.3.2 Defining the Authentication Security Objective 103

5.3.3 Proving EAP-802.1X 104

5.3.4 Proving 4-Way Handshake 112

5.4 Summary 117

CHAPTER 6 A MULTI-AGENT APPROACH FOR SECURITY PROTOCOL VERIFICATION 118

6.1 The Multi-agent Theoretical Model 119

6.1.1 Penetrator . . . 119
6.1.2 Messages . . . 121
6.2 Syntax . . . 122
6.3 Semantics . . . 124
6.4 Theoretical Model for the 4-Way Handshake of 802.11i . . . 127
6.5 Formal Proofs . . . 128
6.5.1 Penetrator . . . 128
6.5.2 Authentication of the Client and the Access Point . . . 129
6.6 Summary . . . 132

CHAPTER 7 FUTURE WORK . . . 133
7.1 Additional Security Properties . . . 133
7.1.1 A Formal Threat model . . . 133
7.2 Verification of a Wireless Mesh Network . . . 135
7.2.1 Virtual Private Network . . . 136

CHAPTER 8 CONCLUSION . . . 137

LIST OF REFERENCES . . . 139

LIST OF FIGURES

Figure 1.1 Stages of Formal Verification 3

Figure 3.1 The Proposed Formal Verification Model 53

Figure 3.2 Authentication in WEP 55

Figure 3.3 Encryption in WEP 57

Figure 3.4 RC4 Algorithm 58

Figure 3.5 EAP Message Format 61

Figure 3.6 Authentication Using 802.1X 63

Figure 3.7 CCMP Encryption and Decryption 67

Figure 5.1 RSNA Stages of 802.11i 93

Figure 5.2 A Strand Space Representation of EAP-802.1X in 802.11i. 101

Figure 5.3 An Abstract Representation of the 4-way Handshake of 802.11i 102

Figure 5.4 Case Analysis of 802.11i. 112

LIST OF TABLES

Table 3.1 Security Objectives and Threats 52

Table 3.2 WEP Problems and Security Objectives 60

Table 3.3 Summary of Keys 64

Table 3.4 Confidentiality and Integrity Comparison 65

CHAPTER 1
INTRODUCTION

Computers are now performing complex tasks in critical environments such as financial institutions, medical facilities, and defense. These tasks are sensitive and can place both money and credibility at stake. A security (or cryptographic) protocol enables communicating parties to communicate securely over an insecure network. A security protocol can be characterized as a sequence of messages between two or more parties in which encryption is used to provide authentication or to distribute cryptographic keys for new conversations [NS78]. The aim of security protocols is to provide secure services on the network. Used with a communication protocol, a security protocol provides secure delivery of data between two or more parties using cryptographic operations such as encrypt, decrypt, and so forth.

1.1 Problem

History has proven security protocols to be vulnerable to attacks despite circumspect design and meticulous review by the experts. For example, the Needham-Schroeder (NS) protocol was proposed in [NS78] and was later found to be flawed in [Low96]. In order to foil such vulnerabilities, one would need to employ more sophisticated modes of analyses. Rigorous testing, programming, and simulation can only show the presence of errors but never their absence. Hence, some formal verification procedure is required to provide sound and complete proofs of the security properties of a system.

1.1.1 Formal Verification

Formal verification can be described as a process of formally proving that the model of a system satisfies the requirement specifications of that system. The application of formal methods to security protocols refers to mathematics or logic based techniques for the specification, development, and verification of these protocols. For security protocols, this implies modeling of both the communicating parties and the potential penetrator. Modeling of the penetrator is necessary as it may sabotage the security protocol by compromising one or more security properties such as authentication, secrecy, etc. Formal methods are useful for the analysis of security protocols. They allow one to do both a thorough analysis of the different paths which an intruder can take, and to specify precisely the environmental assumptions that have been made [Mea03].

Formal verification is a multi-stage process. The authors of [HR04] define verification techniques as comprising of three parts:

- a framework for modelling systems, typically a description language of some sort,
- a specification language for describing the properties to be verified,
- and a verification method to establish whether the description of a system satisfies the specification.

This is further shown in Figure 1.1.

1.2 Our Contributions

In this section, we describe our contributions.

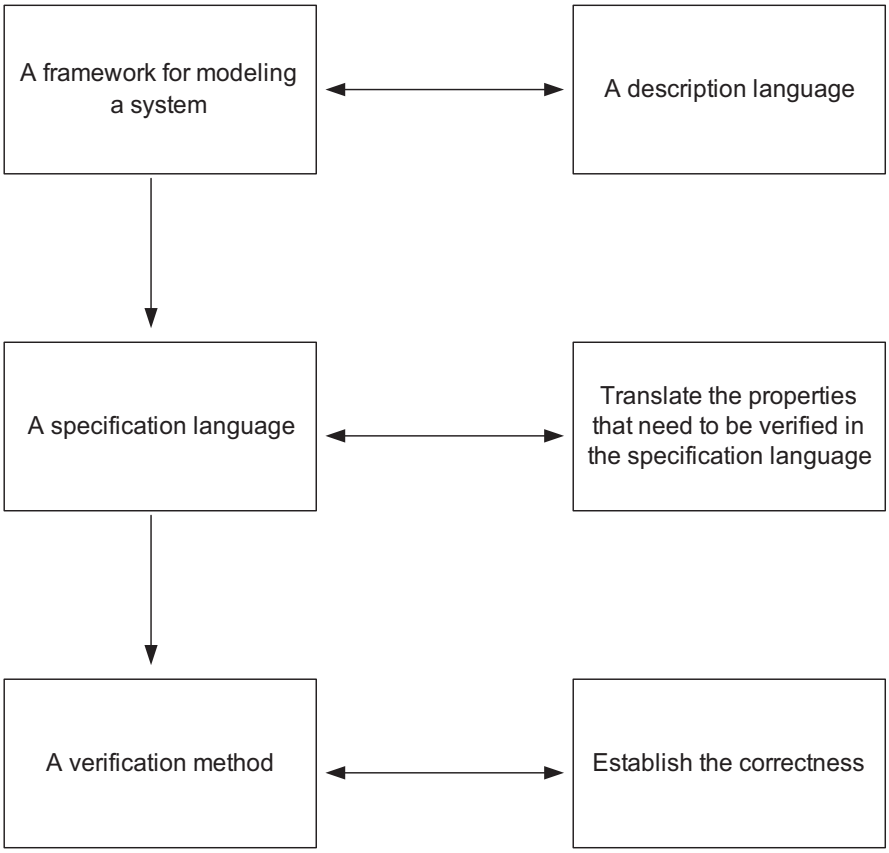

Figure 1.1: Stages of Formal Verification

1.2.1 Explicit Notion of Security Objectives

We introduce an explicit notion of security objectives that needs to be defined before starting the formal verification process. The definition of these objectives is critical because these are the methods that enforce security policies and mitigate security risks. In this work, we address the security of Wireless Local Area Networks (WLANs) in terms of its security objectives. In order to establish the correctness of security objectives, we present a formal model that provides basis for the formal verification of security protocols. The presented mathematical model is then used for the correctness proofs of a WLAN using the 802.11i protocol. We also present various lemmas that provide the foundations for our proof based framework.

1.2.1.1 Motivation and Advantages

The work done in the area of formal verification has not been focussed on the security objectives. A good amount of work has been devoted to the inclusion of explicit notions for different forms of knowledge. We posit that the formal approaches should explicitly model the security objectives. This is important because analysis can prove a certain protocol correct under a set of assumptions that might not actually hold for a specific environment. For example, BAN logic (explained in 2.1.2) proved the NS protocol correct under a set of assumptions that are not valid in some environments. An explicit notion of security objectives would have avoided this problem. In addition, our inclusion of security objectives helps us model the set of requirements that are specific to a certain environment.

We assert that if a designer wants to model and prove the correctness of a security protocol that is going to be used in a certain network environment, he needs to associate an explicit notion

of security objectives before initiating the formal verification procedure. If the security objectives definition is ignored for a security protocol, the security protocol will be proven correct despite the fact that it is not equipped to defend against all potential threats. Such a protocol will fail to achieve one or more of the ignored security objectives and will eventually be jeopardized by the saboteurs. This potential problem will not be easy to detect and a smart saboteur can exploit this potential weakness to cause damage.

1.2.2 Formal Verification of 802.11i

After defining the security objectives of the WLANs, we proceed to the formal verification of the IEEE (Institute of Electrical and Electronics Engineers) 802.11i protocol. We develop the modal logic, proof based, and multi-agent approaches using the Strand Space framework [FHG99]. In our modal logic approach, we present the logical constructs to model a protocol's behavior in such a way that the participants can verify different security parameters by looking at their own run of the protocol. In our proof based model, we present a generic set of proofs to establish the correctness of a security protocol. We model the 802.11i protocol into our proof based system and then perform the formal verification of the authentication property. We first categorize the 802.11i protocol in different stages. Then, we separate the stages using strong authentication techniques from those that do not employ strong cryptographic mechanisms. For example, open system authentication stage in the 802.11i protocol is based on a weak authentication mechanism and hence can not be verified or be used to provide strong authentication.

After identifying the stages that guarantee authentication in the 802.11i protocol, we formally represent these stages in our proof based system. We then present the high level abstraction of a

protocol run. We consider a penetrator, empower its capabilities, and evaluate the authentication of the proposed protocol. Given the constraints and suggestions of the proposed architecture, we state convincingly that any attempt to defy the authentication mechanism of the 802.11i protocol will not be successful. We further describe a situation where modifications to our model will lead to a successful man-in-the-middle attack. Our choice of SSM as a verification framework is based on its simplicity, elegance, precision of results, and ease of developing simple and powerful proofs even without any automated support.

In this work, we restrict ourselves to the verification of the authentication property of the security protocols. In security protocols, authentication can be described as a process of attempting to verify the identity of an agent. Authentication is a method to ensure that users are who they say they are. In [Low96], Lowe defined the role of the authentication protocol as a mechanism where a pair of agents can be assured of each others identity. The agents should become sure that they are really talking to each other rather than to an intruder impersonating the other agent. Our aim is to verify that the user who attempts to perform certain functions in a system is the user who is authorized to do so.

1.2.3 A Multi-agent Framework for Security protocol Verification

We present a new approach that includes the notion of multi-agent for the formal verification of security protocols. In particular, our focus is to verify the authentication property in a multi-agent environment. The intuition is to capture the behavior of a distributed system as a multi-agent system. An agent is an abstraction that is used in the distributed systems to represent an entity such as a process or a principal. It is important to give a clear notion of multi-agent systems

in the proposed frameworks for the security protocol verification in order to represent the formal model of knowledge. For example, the limitation of Strand Space framework is the assumption that all the information available to a principal is either supplied initially or is contained in messages received by that principal [HP03]. However, other important information may also be available to a principal in a security setting, such as a principal may combine information from different roles played by him in a protocol to launch a powerful attack. The presented model captures the combined information, the formal model of knowledge, and the belief of agents over time. After building this formal model, we present a formal proof of authentication of the 4-way handshake of the 802.11i protocol.

1.3 Organization of this Thesis

The rest of this document is organized as follows.

Chapter 2 describes a brief overview of efforts in the field of formal verification of cryptographic protocols. We categorize the efforts made in the formal verification of security protocols in three areas namely logic based approaches, theorem proving, and model checking. All formal techniques begin by expressing the cryptographic protocol in a formal notation or a model and then prove that the expressed model achieves its security goals. In addition, we describe how security objectives were targeted in these approaches. We also discuss how the lack of an explicit notion of security objectives will be a hurdle in the deployment of a protocol from one environment to another and will increase the level of required human interaction and expertise.

In Chapter 3, we describe the security objectives of the WLANs and the protocols developed to meet these objectives in the 802.11 WLANs. We specify authentication and access control,

confidentiality, integrity, and availability, as common security objectives of a WLAN. We establish the importance of these security objectives by describing attacks on the WLAN security. These attacks are induced in a WLAN when one or more security objectives are compromised.

In Chapter 4, we develop the modal logic and the proof based models for the security protocol verification using the Strand Space framework. Our model captures the notion of matching parameters and provides guarantees on the parameters associated with each principal. We present a generic set of proofs to establish the correctness of a security protocol. We also present a formal model for security objectives of 802.11i.

In Chapter 5, we perform the formal verification of the 802.11i protocol using our proof based system. In this Chapter, we prove the authentication of the proposed 802.11i protocol. We start with separating the stages of 802.11i that guarantee authentication from those that do not guarantee authentication because of the lack of any strong cryptographic mechanism. Then, we model the authentication stages of 802.11i with strong cryptographic support, using SSM. After modelling in SSM, we carry out the formal verification of these stages using the Strand Space verification techniques.

In Chapter 5, we present a framework to model the security protocols as multi-agent protocols. The presented model captures the combined information, the formal model of knowledge, and the belief of agents over time. After building this formal model, we present a formal proof of authentication of the 4-way handshake in the 802.11i protocol.

After describing our research, we describe our work plan for future in Chapter 7. The conclusion is followed in Chapter 8.

CHAPTER 2
FORMAL APPROACHES AND SECURITY OBJECTIVES

Employing informal methods for security protocol verification has yielded poor results. The application of formal methods to cryptographic protocols refers to mathematics or logic based techniques for the specification, development, and verification of these protocols. Formal methods are used to give a description of the system to be developed at the specific levels of detail desired. This formal description can be used for the system verification. We categorize the efforts made in the formal verification of security protocols in three areas namely logic based approaches, theorem proving, and model checking. We present a brief summary of them in this chapter.

When a protocol is developed to achieve a desired level of security, it is designed within the context of a set of requirements. When the same protocol is used to solve a similar problem under a different environment, different attacks are introduced. This is not because of any design weakness in the original protocol. Rather this is a consequence of a lack of an explicit security objectives notion associated with this protocol. In addition to describing related approaches in this chapter, we describe how security objectives were addressed in the earlier approaches, how assumptions were made about the operating environment, and how critical it was to address the security objectives under the operating environment.

2.1 Logic Based Approaches

These approaches utilize modal logic to reason about knowledge and belief about messages in a distributed system. Modal logic consists of various statements about belief in or knowledge about messages in a distributed system. It also provides the inference rules for deriving new beliefs

and/or new knowledge from available beliefs and knowledge. The goal of logic based analysis is to infer or prove the correctness condition of the protocol. The techniques based on the logic of belief are considered useful in evaluating the trust, which may be placed in a security protocol. On the other hand, techniques based on the logic of knowledge are suitable for proving the security protocol's security [CP97]. In the logic based approaches, protocol verification is a deductive reasoning process. The first step is to specify rules of inference and axioms. After specifications, the deductive process proceeds as follows [CP97]:

- formally express protocol steps in the language of the logic,
- formally express the desired protocol goals,
- starting with the initial protocol assumptions, build up logical statements after each protocol step using the logical axioms and inference rules,
- compare the logical statements with the desired protocol goals to see if the goals are achieved.

We present a brief overview of some efforts in the logic based protocol verification in the following lines.

2.1.1 Rangan's approach

Rangan proposed a logic about trust and belief in [Ran88]. [Ran88] provides an axiom schema for belief that is frequently referenced in the literature. In his notation, the term $B_i p$ means that the principal i believes p. The schema is as follows:

$\forall i; i = 1, \ldots, m$:

- **A1** All substitution instances of propositional tautologies.

- **A2** $B_i p \wedge B_i(p \Rightarrow q) \Rightarrow B_i q$.

- **A3** $B_i p \Rightarrow B_i B_i p$ (introspection of positive belief).

- **A4** $\neg B_i p \Rightarrow B_i \neg B_i p$ (introspection of negative belief).

- **A5** $\neg B_i$(false) (process i does not believe a contradiction).

The following are the inference rules.

- **R1** From p and p $\Rightarrow$ q infer q (modus ponens).

- **R2** From p infer $B_i p$ (generalization).

Rangan defines the transitivity, euclidian, and serial properties. From the above mentioned rules, A3 corresponds to transitivity, given R2, A4 corresponds to the euclidian property, and A5 corresponds to the serial property. The author views a distributed system as a collection of communicating agents in which an agent's state is the history of its messages. The approach builds a model to maintain the set of beliefs. Rangan formally develops a theory of trust where the expression of trust is the addition of a *well formed formula*(wff). The wff is valid for the axioms of a logic. The proposed model builds on simple trust statements that are used to define properties such as transitivity and Euclidean property.

Before this publication, the knowledge of trust and belief in security were used in intuitive sense. The authors provides a formal notation to these concepts. Formal security specification and verification methods can be integrated into Rangan's logic. The author's approach was targeted at defining the knowledge of belief and trust, which are fundamental to logic based approaches.

Rangan restricts himself to provide basis for logic based approaches and does not address security objectives in different environments particular to any protocol.

2.1.2 BAN Logic

Burrows, Abadi, and Needham devised a logic [BAN90a] based on belief. This logic, abbreviated as BAN logic, introduces wide range of security objects, formal notations to represent security concepts, formulas describing the relationship between principals and data, and inference rules that help derive new rules from existing rules. BAN logic is built upon messages sent and received throughout a protocol session. An informal example can be given as: "If A believes that A have received a message M encrypted with the key K, and A believes that only B knows the key K, then A believe that the message M was sent by B".

The analysis of a cryptographic protocol using BAN starts with a set of initial beliefs. New beliefs are added in the existing set of beliefs with each message sent and/or received. Inference rules are applied to the existing set of beliefs to add new beliefs. If the final set of beliefs meet the targeted goal, then the protocol is correct. A failure in meeting the targeted goal may suggest the presence of a weakness in the security protocol.

BAN defines certain constructs for its framework and then uses postulates in order to derive its conclusions. Some of the constructs are as follows:

$P \mid\equiv X$: P believes X. This construct is central to the logic.

$P \lhd X$: P sees X. It means that P received a message containing X from someone.

$P \mid\sim X$: P once said X. It means that P sent a message containing X.

$\sharp(X)$: The formula X is fresh.

$P \mid\Rightarrow X$: P has authority over X and should be trusted on this matter.

Examples of some of the postulates are given below:

$$\frac{P \mid\equiv \sharp(X), P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X}$$

The above is a nonce-verification rule and it says that if a participant P believes that a message X is fresh, and P believes that another participant Q said X, then P believes that Q believes X.

The jurisdiction rule says that if P believes that Q has jurisdiction over X, then P trusts Q on the truth of X:

$$\frac{P \mid\equiv Q \mid\Rightarrow (X), P \mid\equiv Q \mid\equiv X}{P \mid\equiv X}$$

The following rule states that P believes a set of statements if and only if P believes each individual statement separately:

$$\frac{P \mid\equiv X, P \mid\equiv Y}{P \mid\equiv (X, Y)}$$

Another rule is that if a principal sees a formula, then it also sees its components provided that it knows the necessary keys:

$$\frac{P \lhd (X, Y)}{P \lhd X}$$

Similarly, the authors describe several other rules that allow one to derive new beliefs from the previous beliefs when running a protocol. The part that went under strong criticism is when BAN needs a protocol to be transformed into an idealized form before the start of the analysis. For example, if the server sends a message containing the key K_{AB}, then the corresponding formula is $A < \overset{K_{AB}}{\longrightarrow} > B$. This means that the key K_{AB} is a good key for communication between A and B. A

message in the idealized form is a formula. The entire protocol can be transformed into a set of formulas that represent the intended behavior of the protocol. However, transforming a protocol in an idealized form requires understanding some hidden protocol behaviors. This process involves human intervention to an expert level.

BAN logic is a well known modal logic developed for the security protocol analysis. However, BAN proved the authentication of Needham-Schroeder protocol [NS78], which was later discovered to be flawed by Lowe [Low96]. BAN uses guarantees provided by the pubic key signatures. Lowe showed that the public key signatures are not enough to prove the authentication of the Needham-Schroeder protocol. The weakness is a consequence of the inability of the BAN logic to analyze security protocols to assure that private information remains private.

In addition to the lack of the model knowledge, one problem is pointed out by Nesset in [Nes90]. Nesset shows with a simple example that BAN logic is capable of deducing characteristics about security protocols that by inspection are obviously false. Nesset took the following protocol.

$1 : A \rightarrow B : \{N_A, K_{AB}\}_{K_A^{-1}}$

$2 : B \rightarrow A : \{N_B\}_{K_{AB}}$

This is represented in the idealized form as:

$1 : A \rightarrow B : \{N_A, A <^{\underline{K_{AB}}} > B\}_{K_A^{-1}}$

$2 : B \rightarrow A : \{A <^{\underline{K_{AB}}} > B\}_{K_{AB}}$

The analysis are then carried out as:

$1 : B\ believes \mid ^{\underline{K_A}} > A$

$2 : A\ believes\ A <^{\underline{K_{AB}}} > B$

3 : $A\ believes\ fresh(A <\overset{K_{AB}}{>} B)$

4 : $B\ believes\ fresh(N_A)$

5 : $B\ believes\ (A\ controls\ A <\overset{K}{>} B)$

6 : $B\ sees\ \{N_A, A <\overset{K_{AB}}{>} B\}_{K_A^{-1}}$

From 1, we can have 7 : $B\ believes\ A\ said\ \{N_A, A <\overset{K_{AB}}{>} B\}$

From 4, we can have 8 : $B\ believes\ A\ believes\ A <\overset{K_{AB}}{>} B$

Using 5, we get 9 : $B\ believes\ A\ controls\ A <\overset{K_{AB}}{>} B$

Combining 8 and 9, we get 10 : $B\ believes\ A <\overset{K_{AB}}{>} B$

For message 2, 11 : $A\ sees\ \{A <\overset{K_{AB}}{>} B\}_{K_{AB}}$

From 2, we get 12 : $A\ believes\ B\ said\ \{A <\overset{K_{AB}}{>} B\}$

Using 3, we get 13 : $A\ believes\ B\ believes\ A <\overset{K_{AB}}{>} B$

In the above statements, 2, 8, 10, and 13 are used by BAN that this protocol can be used to distribute cryptographic keys in a secure manner. However, this protocol is insecure as shown in [Nes90].

Burrows et al. [BAN90b] defends by stating that Nesset's example violates one of the assumptions of BAN. Mao and Boyd [MB93] claim that the method for determining assumptions in a protocol is weak in the BAN logic. A slight modification to the assumptions could turn a useless protocol into a valuable one or vice versa. Mao and Boyd also propose a procedure to formalize BAN logic.

BAN attempts to model only belief and does not attempt to model knowledge. Therefore, it can not be used to prove secrecy. BAN logic transforms the protocol into an idealized form but

leaves idealization to intuition. BAN logic lacks semantics to model freshness. Abadi and Tuttle define the semantics for the BAN logic in [AT91]. They also remove unnecessary assumptions by introducing new constructs.

The BAN logic has been successful in finding flaws in some well known protocols such as Andrew secure RPC handshake, and CCITT X.509. In addition, BAN has uncovered redundancy in the NeedhamSchroeder conventional key, OtwayRees, Kerberos, Yahalom, Andrew RPC handshake, and the CCITT X.509 protocols. As such, BAN logic can be called a success. With BAN logic, security objectives needs to be transformed into security properties and then verified using the proposed logical framework. As mentioned above, this transformation requires human expertise. Furthermore, BAN logic is not powerful enough to encompass all of the security objectives. For example, BAN is not able to cater secrecy.

2.1.3 Moser's Approach

Moser presented a knowledge about knowledge and belief in [Mos89]. The proposed logic models the dynamics of belief in cryptographic protocols. The presented logic is situation based where a protocol is viewed as a finite sequence of actions performed by various principals at different situations. Each action is viewed as a primitive term in the language. Moser's logic facilitates the ability to model the dynamic change of each principal's beliefs at each step of the protocol within a logic system.

The send and receive axioms defined in this approach read as:

1. $send_i(M, s, d) \Rightarrow B_i(send_i, (M, s, d)) \vee B_i(receive_j(M, s, d))$,

2. $(receive_i(M, s, d)) \Rightarrow B_i(receive_j(M, s, d))$.

The first axiom states that an agent believes that the other agents receive each message that it sends and it believes that it has indeed sent this message. The second axiom states that an agent who receives a message believes that the other agents also receive it.

The presented logic is basically a non-monotonic logic of belief based on a combined monotonic logic of knowledge and belief. The non-monotonicity is provided by a unless operator that enables one to express preference for beliefs and refutation of those beliefs. Although the theory presented is undecidable, the decision procedure given in for the un-quantified theory can still be used as the basis for a practical proof system. It is clear that great value would also result from combining the logic of knowledge and belief with a temporal logic. The author focuses on knowledge of belief and addressing or translating security objectives remain unaddressed.

2.1.4 CKT5 and KPL

Bieber's CKT5 [Bie90] and Syverson's KPL [Syv90] reason about knowledge. CKT5 extends the epistemic logic. CKT5 allows a user to describe the states of knowledge and ignorance associated with the communication. This is done using encrypted messages. Bieber also extends the logic of knowledge and time with operators that relate directly to the sending and receiving of messages. In CKT5, modal operators have been added to extend the basic epistemic logic. These modal operators express the transmission and receipt of messages.

In CKT5, the hostile environment is assumed. This means that if a message is intercepted by an intruder, it may not arrive at its destination. In other words, messages are not lost, rather the intruder can prevent them from reaching a target. In this approach, *univoque* messages are defined such that, X knows that a message is usable if it is *univoque*. Furthermore, the author reasons in

favor of knowledge rather than belief in order to guarantee security. he defends his argument by making a case that epistemic logic is better at describing the behavior of other agents, as is seen by a strong logic such as CKT5.

Snekkenes gives an example application of CKT5 in [Sne92]. He uses a protocol knows as KP, which is similar to the Needham and Schroeder protocol. He modifies CKT5 in order to distinguish between the role of a principal and its name. To achieve this, a predicate is introduced that maps principals to their roles in a protocol. Unfortunately, KP can be proven secure using CKT5 and hence points out the weakness of CKT5. This further demonstrates that strictly epistemic logic is not sufficient for analyzing the security of authentication protocols.

Syverson's use KPL [Syv90] to reason about the evolution of knowledge about words used in the cryptographic protocols. KPL is a quantified modal logic with corresponding word interpretations. In the logics presented so far, we have seen ways of representing the fact that P knows that K_{PQ} is the secret key between P and Q. However, there has been no way to represent simply the fact that the intruder Z knows $P's$ key. Syverson calls this a key simpliciter. The KPL logic is equipped to represent such a fact. KPL is a quantified modal logic with a corresponding possible-worlds interpretation. In KPL, Z knows $P's$ key if $P's$ key is present in all of the worlds accessible to Z from his current set of possible worlds via some transition. Syverson defines a semantics for the logic that he uses to prove the soundness and completeness of KPL. The soundness and completeness of KPL do not guarantee that there can be no error in the reasoning about a secure protocol, but they do prove that there can be no formal error [RH93]. Both CKT5 and KPL make a distinction between seeing a message and understanding its significance.

The focus in these languages has been to represent knowledge or the evolution of knowledge.

In this sense, these knowledge were more focussed on knowledge representation than devising notions for security objectives. These objectives can be implicitly represented in these languages if a formal mapping between these objectives and knowledge can be developed. However, this would require the presence of an expert. Also the objectives will vary from environment to environment and the expert will have to carefully translate the requirements in terms of knowledge between different environments.

2.1.5 GNY Logic

Gong, Needham, and Yahalom extended the BAN logic to exclude universal assumptions and include some new notions like recognizability [GNY90]. It has vastly improved the BAN logic at the expense of increased complexity. GNY has several rules for possessions and beliefs. It has a much finer level of detail than BAN and can cover more types of protocol. GNY has become an area of active research and several solutions have been proposed to automate the analysis process. GNY facilitates automation but at the same time considered as a complex approach. The complexity may cause significant weaknesses in the protocol analysis. Furthermore, the GNY approach addresses only authentication.

GNY logic recognizes that belief and possession are different. This notion of recognition is a very important contribution. In this extended logic, each principal maintains two sets. These sets correspond to a belief set and a possession set. In addition to the basic constructs of the BAN logic, GNY introduces the following additional constructs as explained in [RH93].

$P \lhd X$: P is told formula X. P receives X, possibly after performing some computation such as decryption. A formula being told can be the message itself, as well as any computable content of

that message.

$P \ni X$: P possesses, or is capable of possessing formula X. At a particular stage of a run, this includes all the formulas that P has been told, started the session with, or was able to compute for formulas it already possesses.

$\phi(x)$: The formula X is recognizable. If P *believes* $\phi(x)$, then P would recognize X if P had certain expectations about the nature of X.

$P \propto X$: P is eligible to send formula X. A principal is only eligible to send something that it possesses or can construct. P is eligible to send formula X. A principal is only eligible to send something that it possesses or can construct.

The GNY logic performs a step-by-step approach towards protocol analysis. The GNY approach makes explicit assumptions and draws conclusions about the final position it attains. The GNY logic is considered better than the BAN logic because it places a strong emphasis on the separation between the content and the meaning of messages, which increases consistency in the analysis. Moreover, GNY facilitates the ability to reason at more than one level. It equips the recipient with the ability to identify the expected messages and deal with the replay attacks.

In [Gon91], Gong describe certain enhancements to the GNY logic. One problem with both BAN and GNY logics is that they can lead to beliefs that do not preserve a causal relation. Another problem with both BAN and GNY logics is that a specification that could not possibly represent a real world situation can still be verified correct. For security objectives, this would imply proving of a non existent objective.

2.1.6 Coffey and Saidha's Approach

Coffey and Saidha combines the logic of belief and knowledge in [CP97]. The techniques that use logic of belief are generally limited to the analysis of authentication protocols. On the other hand, the techniques employing logic of knowledge lack flexibility in analyzing wide range of security properties. The authors combine the logics of belief and knowledge to reason about public-key cryptographic protocols. The axioms reflect the underlying assumptions of the logic and model the low level properties of a cryptographic communication system. The low level properties include the fundamental properties such as the ability of a principal to encrypt/decrypt. The deductive reasoning is carried out by applying inference rules to the axioms. The inference rules are the standard inferences required for natural deduction.

The proposed logic provides a means of verifying public-key cryptographic protocols. The logic can analyze the evolution of both knowledge and belief during a protocol execution and is therefore useful in addressing issues of both security and trust. The logic provides a belief operator and two knowledge operators. One knowledge operator is propositional and deals with the knowledge of statements or facts. The other knowledge operator is a predicate and deals with the knowledge of objects (e.g., cryptographic keys, ciphertext data, etc.). The inference rules provided in this scheme are the standard inferences required for the natural deduction. The axioms of the logic are sufficiently low level to express the fundamental properties of the public-key cryptographic protocols. For example, the ability of a principal to encrypt/decrypt is based on the knowledge of a cryptographic key. The axioms reflect the underlying assumptions of the logic and are summarized below.

The authors define the communication environment to be hostile but maintain that the data

communication system itself is reliable. They further assume that the cryptosystem is collision-free. This means that it is not possible to create the same ciphertext from two different pieces of plaintext. The logic incorporates the following rules of inference:

1. from $\vdash p$ and $\vdash (p \rightarrow q)$ infer $\vdash q$,
2. generalization rules which state that if p is a theorem, then knowledge and belief in p are also theorems,
3. from $p \wedge q$ infer p,
4. from p and q infer $p \wedge q$,
5. from p infer $p \vee q$,
6. from $\sim\sim p$ infer p,

Coffey and Saidha's logic allows the representation of both knowledge and belief, and is therefore capable of assessing the security and trustworthiness of a cryptographic protocol. Similar to other logic based approaches, the security objectives need to be translated in the language and then verified using certain constructs. If the same translated protocol is used in a different environment, the security objectives might need to be translation again before verification.

2.1.7 Other Logic Based Approaches

Syverson and Van Oorschot applied the belief logic to include protocols such as Diffie-Hellman in [SO94]. Mao and Boyd describe four weaknesses in the BAN logic and propose a new logic, which is based on BAN in [MB93]. Mao and Boyd's approach offers several improvements over

BAN. Gaarder and Snekkenes [GS91] extend the BAN logic to analyze a public key cryptosystem (PKCS). The authors of [CSP92] extend the BAN logic using probabilistic reasoning. Their approach calculates a measure of trust rather than the complete trust. The authors define the analysis problem in terms of an equivalent linear programming problem.

The authors of [KG91] present a logic for reasoning about the evolution of belief within the course of a protocol. Woo and Lam present a semantic model for the authentication protocols in [WL93]. They identify correspondence and secrecy as two correctness properties. Correspondence specifies that the different principals in a protocol must execute steps in a lockedstep fashion. This represents the idea that a protocol step can be in response to a previous protocol step and not just an independent event. The authors define an action schema to specify the steps in a protocol. In the protocol specification, each of these actions is preceded by a label [RH93].

The objective of logic based approaches is to provide a sound and complete framework that can be used to prove the correctness of security protocols. The emphasis has been to develop the syntax and the semantics to represent the knowledge of belief and trust and evolution of knowledge. The authors, who have worked in this area, have targeted the explicit notion of knowledge and have built frameworks that use existing facts to build guarantees about certain conditions. Since these approaches have not targeted security objectives primarily, these objectives need to be translated into the proposed logical frameworks. This translation requires the presence of experts. Also, an additional problem is to translate these objectives for every environment where the underlying protocol is supposed to be deployed. This problem can be minimized if general purpose explicit notions of security objectives are present in the logic based approaches.

2.2 Theorem Proving

All formalism based strategies begin by expressing the cryptographic protocol in a formal notation or model and then proving that the expressed model achieves its security goals. Theorem proving is based on logic theories where logic is used along with a formal proof. Mathematical proof is considered as the strongest argument to guarantee the correctness of a system. This method is build upon traditional mathematical reasoning. Existing logical notations are utilized with the addition of new notations to express the formal model into logic. Theorem proving techniques are applied on the resulting model to prove its correctness. These proofs are complex, challenging, and require the presence of mathematical experts.

2.2.1 Paulson's Inductive Approach

L. C. Paulson presents a method based on the proof by induction in [Pau98]. Paulson's inductive approach relies on the concept of trace as a list of events occurred on the network during the course of a protocol. Paulson's approach is inherently algebraic in which traces are defined inductively. An intruder is assumed and it has the access to all the traces. The intruder can decrypt messages if it has the appropriate decryption key. Similarly, the intruder can send messages if it has the appropriate encryption key. Proofs may be carried out by induction on generic trace of the model. This establishes the trace properties that represent goals of the underlying cryptographic protocol. The properties that the protocol must satisfy are proved by induction on all possible traces that the protocol can generate. The properties need to be satisfied after each protocol's step. The interactive theorem prover Isabelle supports the inductive modelling of the protocol in the *Higher Order Language* (HOL).

In Isabelle traces are described as the sequence of events that could occur as the protocol agents execute in a hostile environment. Traces are defined inductively from a set of rules that corresponds to the possible actions of the agents including spies. Paulson uses inductive definitions that list the possible actions, which can be performed by an agent or a system. The induction rule is used to reason about the consequences of an arbitrary finite sequence of actions. The attacker is modeled using the inductively defined operators *analz* and *synth*.The operator *partsH* represents the set of all components of H that can be obtained from it. The set *analzH* represents the most that could be obtained from H without breaking ciphers. The set *synthH* represents the set of messages a penetrator can build up from the elements of H. Only the known messages (or elements) can be used to build up new messages.

A protocol is described in terms of events of different forms. Two forms of events in a trace are defined: *Says A B X* and *Notes A X*, which means that A sends message X to B and A internally stores X respectively. Three additional rules are defined in order to capture the notion of an empty trace (an empty list []). These rules are:

- *Fake messages*. For example, $X \in synth(analzH)$ is a fraudulent message,
- $B \neq Spy$ then *Says Spy B X*, and
- *Accidents* (if S distributed the session key K in a run involving the nonces *Na* and *Nb*, then $Notes\ S\ py\ \{Na, Nb, K\}$).

Induction is applied on the set of traces. For the set of traces, the induction principle says that $P(evs)$ holds for each trace *evs* provided that the property P is preserved under all rules of creating traces. $P[]$ is proved to cover the empty trace. For each of the other rules, an assertion of the form

$P(evs) \Rightarrow P(ev\#evs)$ is proved where event ev containing the new message, is added to the trace evs.

The requirements to check a security property of a protocol is given in a syntax identical to that used to model the protocol. For instance, in the case of the Needham-Schroeder public-key protocol, the requirement may be that if an initiator A sends the nonce N_a to the responder B in its first message and receives the second message back that contains N_a, then B must have sent this message. Paulson's method places no limit on the number of instances and an arbitrary number of instances can be considered with this approach. This is all because of the inductive nature of this approach. However, being a theorem prover, it can not generate counterexamples in case of a failure and there is no guarantee of termination. Isabelle is a generic theorem prover typically used in an interactive fashion. It supports formal reasoning in a number of object logics including HOL.

The translation of security objectives in HOL requires human interaction. Furthermore, there are no explicit rules that adjust these objectives as they move from one environment to another. Objectives such as availability, can exhaust the theorem prover and it might end without creating a counterexample. A well defined representation for security objectives will solve this problem.

2.2.2 Other Theorem Proving Approaches

Kindred and Wing proposed a general theorem proving approach known as theory checking [KW99]. The proposed approach attempts to improve the automated support for using small logics. The authors exhaustively produce all truths in the protocol. They verify that properties hold and explore what effect changes in the protocol will have on the truths generated. They build a logic checker and verify that a logic satisfies certain restrictions.

PVS (Prototype Verification System), developed at the Computer Science Laboratory of SRI (Stanford Research International) provides support for formal specification and verification. It consists of a specification language, a number of predefined theories, a theorem prover, various utilities, documentation, and examples that illustrate different methods of using the system in several application areas [ORS92]. The language of PVS is based on the strongly typed higher-order logic. The PVS theorem prover provides a collection of inference rules that are applied interactively under user guidance within a sequent calculus framework. User-defined procedures can combine these inferences to yield higher-level proof strategies. PVS also provides some integration with model-checking.

2.3 Model Checking

Once a formal model of a system has been established; model checking can be utilized to establish the accuracy of the system. Model checking is an alternative resort to theorem proving. A model checker is a tool that explores the state space of the model to determine if there are any paths through the space that correspond to any successful attack.

There are several advantages associated with model checkers. Model checkers help detect bugs in the early stages of development. This advantage helps reducing the overall development life cycle time. An automated document producer can be associated with a model checker that can produce a documentation of the whole process including the attacks and the states that lead to an attack. Model checkers are clear, complete, and can be used to perform consistency checking and testing. In order to verify security objectives, model checkers need these objectives to be transformed into properties and then given as an input. A change in the environment will require

the model checkers to separately analyze the protocol under different environment variables. On the other hand, model checkers can loop forever while proving a property. This is because of the existence of the state space explosion problem associated with the model checkers.

2.3.1 Longley-Rigby Tool

Longley-Rigby tool [LR92] is a search tool to find a subtle and previously unknown flaw in a hierarchical key management scheme. The approach used in Longley-Rigby tool is similar to that of the Interrogator [MC87]. The main difference between Longley-Rigby tool and Interrogator is that the former have relied upon human intervention to assist in the search. In Longley-Rigby tool, if the system indicates that a word cannot be found by the penetrator, the user can intervene and determine whether or not that is likely to be the case. On the other hand, if the word is indicated to be accessible, this information can be inserted into the database and the search can proceed.

Longley and Rigby use a rule based system that transforms goals into subgoals and can constantly continue this process. They use this rule based scheme to build a tree. In this tree, each node represents a data item. The children of a node represent those data items that are required for the knowledge of the data represented by the father node. In this way, the authors are able to construct a tree in which the root node represents the data required by the intruder for an attack (e.g., a cryptographic key), and the leaves represent those data items that are required to know the root item. The Longley and Rigby tool allows the user to interact with the system. The user can determine whether a data can or cannot be found by the intruder. If the data is judged to be accessible, this information can be inserted into the system and the generation of the tree can proceed. Longley and Rigby managed to find a subtle and previously unknown flaw in a hierarchical key

management scheme.

Expert systems developed specifically for the analysis of cryptographic protocols are more successful than general purpose tools. However, they are often inefficient because they perform an exhaustive search. Sometimes they do not even halt and the results are inconclusive. To cope with these problems, they require human intervention. On the other hand, their advantage is that if they discover a flaw, then the attack scenario that exploits the flaw is directly available. Because of the presence of an exhaustive search, the verification of security objectives such as availability is difficult. This means that an additional provision is needed to model the denial-of-service attacks.

2.3.2 Interrogator

One of the earliest systems to utilize Dolev-Yao's model is the Interrogator [MC87]. The Interrogator is a software tool that takes the protocol specification and a target data item as its input [KMM94]. The tool then performs an exhaustive search to find flaws. The Interrogator models the protocol participants as communicating state machines. The Interrogator assumes the presence of a penetrator who can destroy, modify, and create messages. The tool then attempts to find a state in which the penetrator knows some secret. The output is a message history showing how the penetrator obtained the secret.

In the Interrogator, the protocol participants are modeled as communicating state machines whose messages to each other are intercepted by the intruder who can either destroy messages, modify them, or let them pass through unmodified. Given a final state, in which the intruder knows some word which should be secret, the Interrogator tries all possible ways to construct a path by which that state can be reached. If it finds such a path, then a security flaw is identified. The

Interrogator has not yet found a previously unknown attack on a cryptographic protocol. However, it has been able to reproduce a number of known attacks.

2.3.3 NRL Protocol Analyzer

The NRL Protocol Analyzer [Mea94] is based on the Dolev-Yao model. A specification in the NRL protocol analyzer is modeled on the communication of honest participants. Dishonest participants are assumed to be modeled by the intruder and so are not modeled separately. The NRL employs a strategy similar to that of the Interrogator and the Longley-Rigby tool. In the Interrogator, one uses the tool to find protocol security flaws by specifying an insecure state and attempting to construct a path to that state from an initial state. In NRL, an unlimited number of protocol rounds are allowed in a single path, which make the state space infinite. THE NRL focuses not only on finding paths that are insecure but on proving that these states are unreachable. The NRL allows human intervention, which is necessary when proofs lead into infinite states.

Moreover, in NRL protocol analyzer, each participant of the protocol contains its own local state. The global state of the system is simply the composition of these local states with some state information for the environment or the penetrator. Each participant maintains some learned facts *lfacts* in its local store. For example, if a user A attempts to initiate a conversation with a user B during a local round N at the time T, then the corresponding *lfact* is represented as follows.

$$lfact(user(A), N, init\_conv, T) \quad = \quad [user(B)]$$

Similarly, if a user B receives a message X during a local round S at the time P apparently from the

user A attempting to initiate a conversation, then the corresponding *lfact* is represented as follows.

$$lfact(user(B), S, rcvd\_init\_conv, P) \quad = \quad [user(A), X]$$

The NRL protocol analyzer uses unification in which an incomplete state description represents a set of states. The steps of the protocol are represented as conditional rewriting rules and the goals are formalized as un-reachability theorems.

The NRL protocol analyzer defines a set of requirements using some pre-defined actions that specifies a class of protocols. For instance, consider the following requirement that contains two conditions:

- $\neg(\diamond\, accept(B, A, M, N) \wedge \diamond\, learn(Z, M))$
- $accept(B, A, M, N)) \rightarrow \diamond(send(A, B, (Query, M)) \wedge \diamond\, request(B, A, Query, N))$

The first condition says that if the participant B accepted a message M from the participant A at some point in the past (the past time operator $\diamond$), then the intruder did not learn M at some point in the past. Second condition says that if B accepted message M from A in B's local round N then A sent M to B as a response to a query at B's local round N.

After the transition rules are defined for honest agents and the operations available to all agents are described, the atoms needs to be defined that serves as the basic building block of the words in the protocol. Finally, the rewrite rules are described. An example of a rewrite rule is given below, which says that encryption and decryption with the same key are self canceling.

$rr1 : pke(privkey(X), pke(pubkey(X), Y)) \Rightarrow Y$

$rr2 : pke(pubkey(X), pke(privkey(X), Y)) \Rightarrow Y$

The tool needs high level of expert user interaction. It performs backward search from some insecure state. If the initial state is found, then the path to the initial state represents the counterexample. The NRL has been used to find flaws in protocols by generating paths to insecure states. It has been used to find an authentication flaw in the Simmons' Selective Broadcast protocol and a flaw in the Burns and Mitchell's Resource Sharing Protocol. Since the state space of NRL is infinite, modeling availability attacks is difficult.

2.3.4 FDR

In FDR (Failures Divergence's Refinement) checker, the protocol and the property are described as CSP processes. FDR then verifies whether the property is a refinement of the protocol [Low97]. The simplest notion of refinement is a trace model. If traces of a program P are a subset of traces of a property T, then the program P refines the property T. The expression of the property and the program could be in different languages. For example, the program can be described as a set of states and a state transition function. The property can be expressed in terms of a logical formula. An exhaustive search can be used to check that the logical formula holds at every state.

FDR is a general purpose tool that was initially used to analyze many sorts of systems including distributed databases and communications protocols. It has been used to test whether the protocol correctly achieves authentication and discovers a specific kind of attack on the protocol. On a large scale system, the performance of FDR is constrained under user-specified limits. The failure to find an attack in the large scale systems only asserts that an attack can not be found within the user-specified limits.

In order to exemplify the method, we take the example of the Needham-Schroeder (NS) public-

key protocol. Assume the set *Initiator* represents initiators, and the set *Responder* represents responders, *Key* represents public keys, and *Nonce* represents nonces. Also assume $a, a' \in Initiator, b \in Responder, k \in Key, n_a, n_b \in Nonce$. Then the three messages in NS protocol can be represented using three sets of communication events as follows.

$$
\begin{aligned}
MSG1 &\triangleq \{Msg1.a.b.Encrypt.k.n_a.a'\} \\
MSG2 &\triangleq \{Msg2.b.a.Encrypt.k.n_a.n_b\} \\
MSG3 &\triangleq \{Msg3.a.b.Encrypt.k.n_b\} \\
MSG &\triangleq MSG1 \cup MSG2 \cup MSG3
\end{aligned}
$$

The channels are defined as follows. "channel $comm, fake, intercept : MSG$" represent the standard communication channel, intruder's faking messages, and the intruder's interception of messages. These channels assumes MSG as their type. Moreover, additional channels "channel $user, session, I\_running, R\_running, I\_commit, R\_commit : Initiator.Responder$" are the channels of the type *Initiator.Responder* that represent a user's request to connect the initiator and the responder, a session channel, initiator's taking part in a run of the protocol, responder's taking part in a run of the protocol, initiator committing to a session, and the responder committing to a session respectively.

A CSP process $INITIATOR(a, n_a)$ represents an initiator with an identity a and a nonce n_a.

Without intruder action, this process is defined as follows.

$$
\begin{aligned}
INITIATOR(a, n_a) \triangleq\ & user.a?b \rightarrow I\_running.a.b \rightarrow \\
& comm!Msg1.a.b.Encrypt.key(b).n_a.a \rightarrow \\
& comm.Msg2.b.a.Encrypt.key(a)?n'_a.n_b \rightarrow \\
& if\, n_a = n'_a\ then\ comm!Msg3.a.b.Encrypt.key(b).n_b \rightarrow \\
& I\_commit.a.b \rightarrow session.a.b \rightarrow Skip \\
& else\ Stop
\end{aligned}
$$

Now renaming is applied in order to cater the intruder who can intercept messages 1s and 3s and can fake messages 2s. The resulting initiator is given as follows.

$$
\begin{aligned}
INITIATOR1(a, n_a) \triangleq\ & INITIATOR(A, N_a) \\
& [[comm.Msg1 \leftarrow comm.Msg1, comm.Msg1 \leftarrow intercept.Msg1, \\
& comm.Msg2 \leftarrow comm.Msg2, comm.Msg2 \leftarrow fake.Msg2, \\
& comm.Msg3 \leftarrow comm.Msg3, comm.Msg3 \leftarrow intercept.Msg3]]
\end{aligned}
$$

The responder can also be defined similarly. The intruder is defined such that it can fake all the messages using its knowledge base. It can also intercept all the message and learn new nonces if it possesses the right decryption key.

In order to test whether any protocol meets its authentication goal, FDR takes two inputs, a specification and an implementation. FDR then tests whether the implementation refines the

specification. The system is defined as the parallel composition of the agents and the intruder synchronizing on the set of channels. The system is represented as follows:

$$AGENTS \triangleq INITIATOR1 |[\{|comm, session.A.B|\}]| RESPONDER1,$$

$$SYSTEM \triangleq AGENTS |[\{|fake, comm, intercept|\}]| INTRUDER.$$

The above mentioned system represents parallel composition of all the processes in the system. Running the system results into several traces. Now, it needs to be verified that each trace of the implementation is also a trace of the specification. The specification for authentication of a responder AR is given as:

$$AR_0 \triangleq R\_running.A.B \rightarrow I\_commit.A.B \rightarrow AR_0$$

$$A_1 \triangleq \{|R\_running.A.B, I\_commit.A.B|\}$$

$$AR \triangleq AR_0 ||| RUN(\Sigma \backslash A_1)$$

AR_0 means that an $I\_commit.A.B$ event should only occur after an $R\_running.A.B$ event. If Σ is the set of all events, then $AR_0 ||| RUN(\Sigma \backslash A_1)$ represents occurring of all events in an arbitrary order. The above specification says that the initiator A commits to a session with the responder B only if the responder has really taken part in the protocol run. FDR can now be used to verify that *SYSTEM* refines AR, indicating that the protocol correctly authenticates the responder.

*2.3.5 Mur*Φ

The authors of [MMS97] devised a general-purpose state enumeration tool, named MurΦ (pronounced as "Mur-phi"), to analyze security protocols. The approach used by MurΦ is similar to the approach used in CSP model checking of cryptographic protocols. It involves modeling the protocol and the desired properties in the Φ language. MurΦ uses breadth-first or depth-first full state enumeration to verify that all reachable states of the system satisfy the specification.

The analyzing methodology of MurΦ involves the following successive steps: formulate the protocol, add an adversary to the system, state the desired correctness condition, run the protocol for some specific choice of system size parameters, experiment with alternate formulations, and repeat. MurΦ has been used to demonstrate flaws already known as TMN and Kerberos version 5. In MurΦ, it is feasible to modify a system description to reflect a situation where one or more pieces of secret information have been compromised.

The Murϕ language is a high-level language for describing nondeterministic finite-state machines. First the protocol is modeled in this language, and then the desired properties to be verified is specified by the invariants, which are boolean conditions that have to be true in every reachable state. The state showing the violation of the invariant contributes the flaw in the protocol. Lets take the example of the NS public-key protocol and see its model in murϕ. The data structure for the initiator is given below.

```
const
    NumInitiators: 1;
type
```

```
    InitiatorId: scalarset (NumInitiators);
    InitiatorStates: enum{I_SLEEP,I_WAIT,I_COMMIT};
    Initiator: record
        state: InitiatorStates;
        responder: AgentId;
    end;
var
    ini: array [InitiatorId] of Initiator;
```

The state of each initiator is stored in the array `ini`. `I_SLEEP`, `I_WAIT`, and `I_COMMIT` represent that the initiator has not started the protocol, the initiator has started the protocol, and the initiator is committing the protocol respectively. The behavior of an initiator is modeled by two murϕ rules. In the first rule, the initiator starts the protocol by sending the first message of the NS protocol and changes its local state from `I_SLEEP` to `I_WAIT`. The second rule models the reception and checking of the second message of the NS protocol, and then commit (`I_WAIT` to `I_COMMIT`) and send the final message.

Finally the invariants represents the correctness specification of the protocol as follows.

```
invariant "responder correctly authenticated"
    forall i: InitiatorId do
        ini[i].state = I_COMMIT &
        ismember(ini[i].responder, ResponderId)
        ->
        res[ini[i].responder].initiator = i &
```

```
        ( res[ini[i].responder].state = R_WAIT |
          res[ini[i].responder].state = R_COMMIT )
end;
```

It says that for each initiator `i`, if it is committed to a session with a responder, this responder (with the identifier stored in `ini[i].responder`) must have started the protocol with the initiator `i`, i.e, have stored `i` in its field `initiator` and be in state `R_WAIT` or `R_COMMIT`. The intruder maintains a set of overheard messages and an array of all known nonces. Three rules represent an intruder: one for overhearing and intercepting messages, one for replaying messages from the set of overheard messages, and one for generating messages using the known nonces and injecting them into the network.

2.3.6 Brutus

The authors of [CJM00b] devised a special-purpose tool Brutus for analyzing security protocols. The approach of Brutus separate the intruder from the model. The intruder is encoded as a set of rewrite rules that can be applied to messages sent during the execution of the protocol. Brutus contains a state exploration tool and message derivation engine. The state exploration tool performs the search. The message derivation engine uses the rewrite rules to model the intruder capabilities. Thee state exploration tool and message derivation engine interact with each other. In addition to the components, a specification language is provided, which is powerful enough to describe a variety of security properties. The language can be used to specify what information should be kept secret and what should be known to the specific participants. The authors applied partial order reductions in [CJM00a] to address the state space explosion problem.

Brutus uses some rules to define message *derivability* relation in order to model the capabilities of the adversary. A protocol is modeled as an asynchronous composition of a set of named communicating processes, which model the honest agents and the penetrator. Brutus makes the model finite by placing a bound on the number of *sessions* (number of times a principal may attempt to execute the protocol). Each session is modeled as a principal instantiating some role in the protocol, called an *instance*. Each instance is described as a separate copy or instantiation of a principal and consists of a single execution of the sequence of actions, which makes up that agent's role in the protocol along with all the variable bindings and knowledge acquired during the execution. A principal can have multiple instances but each instance is executed once. The entire model for the protocol is obtained by combining these instances with a single instance of the adversary. A *trace* is defined as each possible execution of the model and can be obtained from a finite alternating sequence of global states and actions. Two kinds of *actions* are defined: *send*, and *receive*. In addition, user-defined actions are defined as well.

In order to specify properties of a protocol, Brutus uses first-order logic in the model. Modal logic is also combined with the predicate logic in order to capture the notion of the past-time operator. In this way, one can use the past-time operator to talk about the things that happened in the history of a particular protocol's run. The atomic propositions of the logic allow to refer to the bindings of variables in the model to actions that occur during execution of the protocol, and to the knowledge of the different agents participating in the protocol. The logic used, can be seen as a variant of the linear-time temporal logic with the past-time operator where one can express actions and knowledge. After the protocol is modeled, the model checker runs and checks the desired specifications in each of its states. Like any model checker, it gives the trace of the run (a

counterexample) whenever any state does not meet the specification.

2.3.7 Athena

Athena [SBP01] is a model checker that is based on the Strand Space Model (SSM) of [FHG98]. Athena formally reduces the infinite-state-space problem into a finite-state-space problem, which can be verified using model checking. Hence, for a well-formed formula, wff, if the evaluation procedure terminates, then it will generate a counterexample if the formula is false, or provide a proof if the formula is true. Although the evaluation procedure is not guaranteed to terminate, experience shows that it does terminate for many useful protocols. In those cases when the procedure does not terminate for arbitrary configurations of the protocol execution, termination can always be forced by bounding the number of concurrent protocol runs and the length of messages. This is similar to the bounds in current model checkers such as FDR, MurΦ, and Brutus.

Athena also exploits several state space reduction techniques.

1. The state transition is not asynchronously composed of independent process transitions, hence, avoid the state space explosion caused by asynchronous composition.

2. The state structures and state transitions capture exact causal relations, hence, achieve compact and efficient state representations.

3. Athena takes advantage of the symbolic state transitions, instead of an explicit state search, by allowing a state to contain free variables. A state $s(x)$ with free term variables, x, represents a class of variable free states, $\{[\sigma / x]s(x)\}$.

A state transition between two states, which contain free variables represents a set of state tran-

sitions between two variable-free states. Thus, Athena can represent states and state transitions much more efficiently. As a special case of this, it naturally avoids the symmetry redundancy problem. Finally, Athena uses backward search instead of forward search. With forward search, all the participating principals have to be pre-stated. The approach of Athena starts with a simple initial strand and then add new strands only when necessary according to the exact causal relations. These techniques greatly reduce the state space that needs to be explored comparing with other current approaches. Athena also has the advantage that it can easily incorporate results from theorem proving through un-reachability theorems. By using the un-reachability theorems, authors prune the state space at an early stage, hence, reduce the state space explored even further and increase the likelihood of termination.

To evaluate a wff formula of the form, $\forall.C.f$, the authors define a lemma. The lemma states that if H is an algorithm that decides the validity of any wff of the form $\forall.C.f_1 \Rightarrow f_2$, in a model M in finite steps, where f_1 is a conjunction and f_2 is a disjunction of the atomic propositions, then there exists an algorithm to evaluate any formula of the form $\forall.C.F$ in the model M in finite steps, for any propositional formula F. The authors then formulate their verification procedure in terms of a proof search in a very specialized proof system.

Athena first transforms the security property to be verified into an initial sequent, which contains an initial state. It then applies a small set of inference rules with certain decision procedures to the states, building a proof tree until it either completes the proof or refutes a sequent. In the latter case, Athena reports the protocol to be incorrect and the state of the refuted sequent represents a counterexample or a successful attack on the protocol. One main difference between Athena and previous approaches is that Athena uses fundamentally different representation of protocol execu-

tions. Instead of the finite state verifications techniques described before, Athena uses an extension to SSM, a much more compact state structure based on *semi-bundles* and *goal-bindings*. The goal-binding is the causal relation $\twoheadrightarrow$ that captures the exact information about the origins of messages in a protocol execution. A set of protocol runs that differ only in the order of interleaving executions of individual parties is in fact represented by one state in Athena. Athena can reason about all such executions simultaneously. This form of the state structure allows to develop efficient state search procedures avoiding the exponential growth of the state space due to an asynchronous composition.

2.4 Other Approaches

2.4.1 Dolev and Yao

The authors of [DEk82, DY83] contributed the early work in the application of formal methods to verify cryptographic protocols. Dolev and Yao's work [DY83] is significant as it proposed the first algebraic model for the security of protocols and encompassed the capabilities of the penetrator. Dolev and Yao's model assumes and intruder who can create, destroy, read, and alter messages. The intruder can also perform encryption and decryption operations that are available to legitimate users of the network. The authors assume perfect encryption.

The authors suggest the following two conditions to be necessary in order to assure the security of the cascade protocols:

1. The messages transmitted between two participants must contain some layers of encryption functions.

2. In generating a reply message, each participant A never applies decryption function D_A without also applying encryption function E_A.

Similarly, for a two party name stamp protocol T, the authors define that T is insecure if a string $\gamma \in V^*\overline{N_i(X,Y)}$ exists such that $\overline{\gamma} = \lambda$. Here, V is the string of operators that an intruder can apply on any message, and $\overline{N_i(X,Y)}$ is the sequence of texts transmitted between X and Y when X wishes to send plaintext M to Y. Otherwise, T is defined to be secure. Consider the following two party name stamp protocol:

1. $X \rightarrow Y$: $(X, E_y(E_y(M)X), Y)$

2. $Y \rightarrow X$: $(Y, E_x(E_x(M)Y), X)$

The authors have proven this protocol to be flawed. For the above mentioned protocol, the sequence of operators applied by the participant X in the first message is equal to $\overline{N_1(X,Y)} = E_Y i_X E_Y$. That is, the participant X first applies encryption E_Y (using Y's public key) on message M, then appends his own id X (append operation i_X), and finalizes the message by applying the encryption operation E_Y again. Similarly, in the second message the participant Y applies the operators $\overline{N_2(X,Y)} = E_X i_Y E_X$. The authors also defined additional operators such as d for deleting an id from a string, d_X for deleting a known id X from a string, and D_X for the decryption using X's private key. The attack on this protocol exists as the authors found the string $\gamma = D_Z d D_Z E_Z i_X E_Z D_X d_Z D_X E_X i_Z d D_Z d D_Z E_Z i_X E_Z D_X d_Z D_X E_X i_Z \overline{N_2(X,Y)} \in V^*\overline{N_i(X,Y)}$ such that $\overline{\gamma} = \lambda$. γ simply represents the sequence of operators a saboteur applies on a protocol message resulting into a null string. In this way, a saboteur can obtain the secret plaintext from an encrypted message.

The proposed model is extremely limited. It can only be used to detect failures of secrecy.

The suggested model does not allow the participants to remember their state information as they move from one state to another. However, the proposed model is significant because it was the first formal model to represent multiple executions of a protocol. Most of the work done in formal analysis of security protocols is inspired from Dolev and Yao's model. Dolev and Yao define some classes of protocols. They reason about these classes of protocols rather than the individual protocols themselves. In addition, they prove certain interesting properties of these classes.

2.4.2 Kemmerer's Work

Kemmerer models cryptographic protocols as communicating state machines [Kem89]. The protocols are modeled in a specification language. The specification language has an attached theorem prover. Because of this attached theorem prover, it is possible to use the prover to prove theorems about the security of the specified protocols. The security is proved by defining security properties as state invariants and proving that these invariants are preserved by each transition.

Kemmerer uses an extension of the first-order predicate calculus, a formal specification language called Ina Jo. Ina Jo was designed as a general purpose tool to support software development and correctness proofs. Kemmerer describes an example security system and then gives an Ina Jo specification of it. He also specifies critical requirements that the system is to satisfy in all states. Once the specification is complete, Ina Jo generates theorems that can be used to verify whether the critical requirements are satisfied or not. Kemmerer un-covers a weakness in his sample system. The value of this method is limited because the specification of the critical requirements needs that the designer knows the potential attacks in advance. This in turn makes the translation of security objectives in Ina Jo a difficult process.

2.4.3 Type Checking

Abadi proposed type checking [Aba99], which is an approach towards protocol analysis. This approach has a potential disadvantage of defining security violations in terms of type inconsistencies. Hence, the security requirements must be considered when the specifications are being written. Another approach [AG99] is to develop a computational model. A formal system is also developed and a proof of soundness is established for the system using the computational model. This work develops a method for the static checking of secrecy properties of programs written in a minimal but expressive programming language, the spi-calculus. These programs can be concurrent and can use cryptography. The method is embodied in a set of typing rules. The principles and rules developed are neither necessary nor sufficient for security. They are not necessary because like most practical static type-checking disciplines, this is incomplete. They are not sufficient because they ignore all security issues other than secrecy, and also because they do not account for how to implement the spi-calculus while preserving secrecy properties. However, these principles and rules provide some useful guidelines.

In this approach, encryption keys are pieces of data, and as such they are labeled. The result of encrypting secret data under a secret key can be made public but only with some precautions. For example, given a secret bit b and a secret key K, we cannot simply publish b under K. If we may also publish 0 or 1 under K, an attacker could deduce the value of b by comparing ciphertexts. The rules of this paper capture a sufficient set of simple precautions that permit the publication of ciphertexts that contain secrets. The approach relies on a binary view of secrecy according to which the world is divided into a system and an attacker. A finer model could distinguish the individual principals and groups within the system. It could also permit a declassification operation whereby

a principal can reduce the secrecy level of the data that it owns. The typing system does not allow comparisons between terms of level. The typing system may be relaxed to allow such operations in cases where their outcomes are not revealed on the public channels.

2.4.4 Strand Space Model

In [FHG99], a *strand* is defined as a sequence of events that a participant may engage in a protocol. A strand represents either a protocol execution by a legitimate party (*regular strand*) or by a penetrator (*penetrator strand*). A *strand space* is a set of strands, consisting of strands of various legitimate protocol parties, together with the penetrator strands. In SSM, each participant of the security protocol is represented by a strand and the individual run of each participant is captured by the traces of these strands. The correctness claims for the protocol are then expressed in terms of connections between different kinds of strands. In the following lines, we describe the basic Strand Space Model (SSM) terminologies that are necessary to understand the motivation and intuition behind strategies based on SSM.

The set of actions, that a participant may take during the execution of a protocol includes actions such as send, (denoted by +) and receive (denoted by -). Let M be the set of possible messages that can be exchanged by all the participants in a protocol. A *signed term* is a pair $\langle \sigma, u \rangle$ with $\sigma \in \{+, -\}$ and $u \in M$. A signed term $\langle +, u \rangle$ represents the sending of a message u and is typically written as $+u$. Similarly, $-u$, represents the reception of a message u. The set of finite sequence of signed terms is represented as $(\pm M)^*$. A strand space over M consists of a set Σ, whose elements are called strands, together with a trace mapping $tr : \Sigma \rightarrow (\pm M)^*$. The trace mapping tr associates each strand in Σ with a sequence of signed terms.

A *node* is a pair $\langle s, i\rangle$ with $s \in \Sigma$, and i is an integer with $1 \leq i \leq |tr(s)|$. The set of all the nodes is donated by N. For a node $n = \langle s, i\rangle$, where $tr(s) = \langle \sigma_1, u_1\rangle ... \langle \sigma_k, u_k\rangle$, $term(n)$ is defined as $\langle \sigma_i, u_i\rangle$. There is an *edge* $n_1 \rightarrow n_2$ if and only if $term(n_1) = +a$ and $term(n_2) = -a$ for some $a \in M$. The edge $\rightarrow$ represents a potential causal link between two strands. When $n_1 = \langle s, i\rangle$ and $n_2 = \langle s, i + 1\rangle$ are nodes, then there is an edge $n_1 \Rightarrow n_2$. The edge $\Rightarrow$ indicates that n_1 is an immediate causal predecessor of n_2. In a similar fashion, $n' \Rightarrow^+ n$ implies that n' precedes n (not necessarily immediately) on the same strand. A term t occurs in $n \in N$ if and only if $t \sqsubset term(n)$. The node $n \in N$ is an entry point for the term t if $term(n) = +t$ and whenever $n' \Rightarrow^+ n$, $term(n') \notin term(n)$. A term t *originates* on $n \in N$ if and only if n is an entry point for t. A term t *uniquely originates* if and only if t originates on a unique $n \in N$. It can be seen that N, together with both sets of edges $n_1 \rightarrow n_2$ and $n_1 \Rightarrow n_2$, is a directed graph $\langle N, (\rightarrow \bigcup \Rightarrow\rangle$.

A *bundle* represents a full protocol exchange. It consists of a number of strands hooked together where one strand sends a message and another strand receives the same message. Intuitively, a bundle is a portion of a strand space large enough to represent at least a full protocol exchange. Bundle has a natural causal precedence relation relative to which inductive arguments are carried out. A bundle is a finite acyclic subgraph that captures the natural causal precedence relation among nodes as defined by the edges $\rightarrow$ and $\Rightarrow$. For a given strand space Σ, let $B = \langle N_B, (\rightarrow_B \bigcup \Rightarrow_B\rangle$ be a subgraph of $\langle N, (\rightarrow \bigcup \Rightarrow\rangle$. The graph B is a bundle if:

1. B is finite,

2. if $n_2 \in N_B$ and $term(n_2)$ is negative, then there is a unique n_1 such that $n_1 \rightarrow_B n_2$,

3. if $n_2 \in N_B$ and $n_1 \Rightarrow n_2$, then $n_1 \Rightarrow_B n_2$,

4. B is acyclic.

The bundle-height of a strand is the largest i such that $\langle s, i\rangle \in$ bundle.

The set $T \subseteq M$ is the set of texts (representing the atomic messages). The set $K \subseteq M$ contains cryptographic keys disjoint from T. The term $\{g\}_k \in M$ represents the encryption of the term $g \in T$ using $k \in K$. The *subterm relation* $\sqsubset$ is inductively defines as the smallest relation such that:

1. any term $m \in M$ is a subterm of itself,

2. a term $m \in M$ is a subterm of $\{g\}_k$, if m is a subterm of g,

3. a term $m \in M$ is a subterm of gh, if m is a subterm of g or h.

The detailed explanation of the concepts introduced in this Section can be found in [FHG99].

2.5 Summary

All formalism based strategies begin by expressing the cryptographic protocol in a formal notation or model and then proving that the expressed model achieves its security goals. We have categorized the verification process in three major categories; logic based approaches, theorem proving, and model checking. Logic based approaches utilize modal logic to reason about knowledge and belief about messages in a distributed system. The goal of logic based analysis is to infer or prove the correctness condition of the protocol. Theorem proving approaches employ mathematical proofs, which are considered as the strongest argument to guarantee the correctness of a system. This method is build upon the traditional mathematical reasoning. In model checking, a formal model of a system is established and then finite state verification is performed to establish the accuracy of the system.

The logic based approaches has emphasized syntax and semantics to represent knowledge of belief and trust and evolution of knowledge. These approaches were not targeting security objectives primarily. Hence, these objectives need to be translated into the proposed logical frameworks. This translation requires expert human interaction. An additional problem is to translate these objectives for every environment where the underlying protocol is supposed to be deployed. This problem can be minimized if general purpose explicit notions of security objectives are present in the logic based approaches. The translation of security objectives in theorem provers requires human interaction. Furthermore, there are no explicit rules that adjust these objectives as they move from one environment to another. Objectives such as availability, can exhaust a theorem prover and it might end without creating a counterexample. A well defined representation for security objectives will solve this problem. Model checkers help detect bugs in the early stages of development. Model checkers are clear and helpful in testing and consistency checking. At the same time, they suffer from the state space explosion. This problem may prevent a model checker to provide a correctness proof about a security protocol.

In this chapter, we have discussed many formal approached that have been utilized for the analysis of security protocols. An interesting observation is that most of these approaches are not targeting the security objectives. Instead, they are focussed on a specific sub problem that will be used to solve a bigger problem. We state that ignoring security objectives in designing formal tools may make it difficult to verify protocols with evolving requirements and operating in different environments. An explicit notion of security objectives will not only increase protocol portability across different platforms, it will also make a language/tool more marketable, reliable, and complete in terms of addressing different security requirements.

CHAPTER 3
SECURITY OBJECTIVES IN WLANS

Many standards with varying levels of security have been developed for WLANs. IEEE ratified the 802.11 standard, also known as Wireless Fidelity (Wi-Fi), for WLANs in 1997 [IEE99]. In order to provide security, a scheme known as Wired Equivalent Privacy (WEP) was used in the 802.11 standard. Many vulnerabilities were discovered in WEP [CHW03][FMS01][MRH04][PD03] that motivated the researchers to design a new security mechanism for WLANs. IEEE addresses the problems of WEP in a new amendment known as IEEE 802.11i [IEE04a].

802.11i introduces a range of new security features such as Robust Security Network (RSN), which are designed to overcome the shortcomings of WEP and provide additional security measures. RSN is defined as a wireless security network that allows the creations of Robust Security Network Associations (RSNAs) only. RSNAs are wireless connections that provide moderate to high levels of assurance against WLAN security threats through the use of a variety of cryptographic techniques [Nat06]. The three types of RSN components are Station (STA), Access Point (AP), and Authentication Server (AS). The STAs are wireless endpoint devices such as laptops. The APs are network devices that allow STAs to communicate with each other or with a network such as an Internet, without a physical wired connection. STAs and APs were also present in the earlier WLANs. The AS is a new component introduced by the RSN framework to provide authentication services to the STAs.

This chapter is organized as follows. We present the importance of using security objectives and the threats that result when these objectives are violated in Section 3.1. Section 3.2 describes the initial measures taken by the IEEE 802.11 standard to meet its security objectives and their

effectiveness. In Section 3.3, we discuss how 802.11i addresses its security objectives.

3.1 Security Policy and Security Objectives

A security policy is a set of rules that govern activities for the resources belonging to an organization. In a computing environment, these rules govern activities such as device security, administrative controls, network security, etc. The security policy much define what is to be protected and what are the expectations of the system users. This policy is very important as it provides the basis for security planning when new applications are designed or the current network is expanded. In order to define a security policy, definition of security objectives is necessary. Security objectives address areas such as resource protection, authentication and authorization, integrity, etc.

3.1.1 Importance of using Security Objectives

The work done in the area of formal verification lacks an explicit notion of security objectives. This is because the focus has been on the inclusion of explicit notions for knowledge of trust, knowledge of belief, and evolution of knowledge as described in detail in Chapter 2. We introduce an explicit notion of security objectives for the formal verification approaches. This notion is important because analysis can prove a certain protocol correct under a set of assumptions that might not actually hold for a specific environment. For example, BAN logic (as is explained in 2.1.2) proved the NS protocol correct under a set of assumptions that are not valid in some environments. An explicit notion of security objectives would have avoided this problem.

In addition, our inclusion of security objectives helps us model the set of requirements for a specific network environment. For example, assume that an organization is to deploy the NS

Table 3.1: Security Objectives and Threats

Threat	**Security Objective(s)**
Eavesdropping and Traffic Analysis	Confidentiality
Message Insertion, Modification, and Replay	Confidentiality and Integrity
Masquerading	Authentication and Access Control
Man-in-the-Middle	Authentication and Access Control
Session Hijacking	Authentication, Confidentiality, and Integrity
Denial-of-Service (DoS)	Availability, Authentication, Confidentiality, and Integrity

protocol from a wired environment to a wireless environment. Under our model, the proof of correctness of NS protocol in a wired network is not sufficient to guarantee the legitimate working of the protocol in the wireless network. The organization has to define its security objectives for the wireless environment and then verify the NS protocol with the newly added constraints. Under our scheme, the formal verification process would appear as shown in Figure 3.1.

In future years, we expect large deployments of WLANs in different spheres of life. Many critical domains such as defence and banks, are concerned with the security threats in WLANs. These threats are potential violations of one or more security objectives. In Table 3.1, we map these threats to their corresponding security objectives.

3.2 Security objectives in 802.11

Wireless networks introduce additional risks than those associated with the wired networks. In order to mitigate these risks, organizations need to adopt security policies that reduce threats to a manageable level. This calls for analyzing possible threats and then forming security objectives that counter these threats. The IEEE 802.11 specification utilizes several security services in order to carry out secure operations in WLANs. These security services are provided largely by the

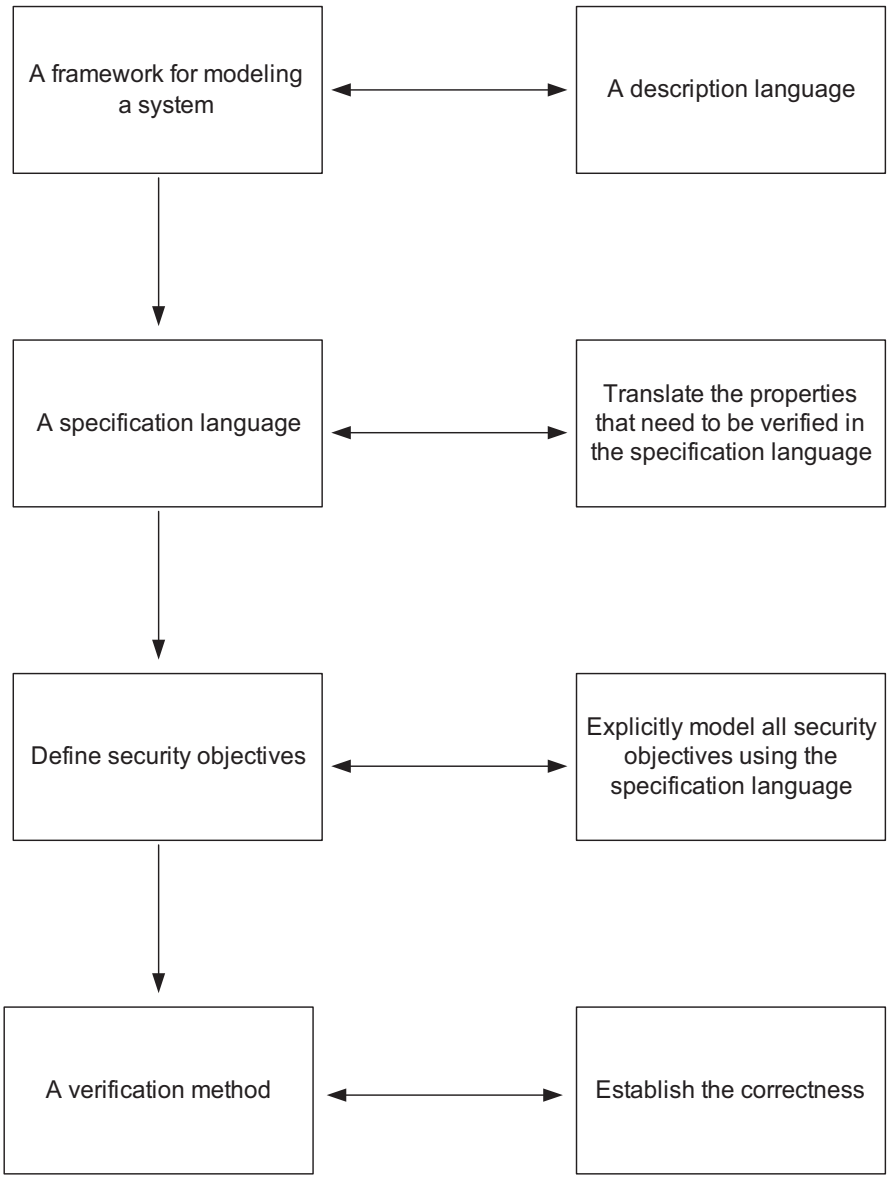

Figure 3.1: The Proposed Formal Verification Model

Wired Equivalent Privacy (WEP) protocol. This section explains how the security objectives were addressed in the 802.11 standard. It also discusses the shortcomings of WEP in order to understand the motivation behind the IEEE 802.11i amendment and its RSN framework.

3.2.1 Authentication and Access Control

The IEEE 802.11 specification defines two authentication methods to validate the identities of the participants attempting to gain access to a WLAN. These authentication methods are open system authentication and shared key authentication. In open system authentication, a wireless station is authenticated to an AP by providing its MAC address and Service Set Identifier (SSID) for the AP. 802.11 allows MAC address filtering, which is prone to traffic analysis attack. The adversary can easily spoof an authorized MAC address and then set its own MAC address to the spoofed MAC address. Similar to MAC addresses, SSIDs are prone to eavesdropping because they are broadcasted in the plaintext. Furthermore, open system authentication only allows one way authentication where an AP authenticates a client. Hence, the wireless client is not sure whether it is communicating with a legitimate AP or an adversary impersonating as a legitimate AP.

In contrast to open system authentication, shared key authentication is a cryptographic technique based on a simple challenge-response criteria. The shared key authentication scheme is based on a secret cryptographic key known as the WEP key, which is shared by the legitimate participants. In order to authenticate a STA, the AP generates a 128-bit random challenge and sends this challenge to the STA as shown in Figure 3.2. The STA encrypts the AP's challenge using the WEP key and returns the result to the AP. The AP decrypts the STA's result using the same WEP key. The AP grants access to the STA, if the decrypted value is the same as the initial challenge

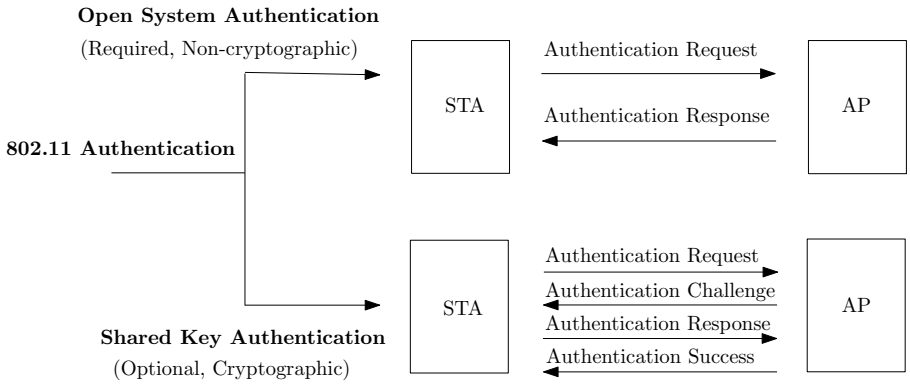

Figure 3.2: Authentication in WEP

sent by the AP. Otherwise, the AP denies access to the STA. Both open system authentication and shared key authentication are summarized in Figure 3.2. Similar to open system authentication, the AP is still not authenticated to the STA in the shared key authentication.

3.2.1.1 Key Management

The IEEE 802.11 does not specify a key management technique. In shared key authentication, all devices are configured using the same or small sets of WEP keys. This means that if an 802.11 device is lost or stolen, the cryptographic keys could become compromised and would put security of the rest of the network devices in jeopardy. In WEP, same key is extended to provide confidentiality and integrity. Thus, a compromised WEP key jeopardizes other security objectives beyond authentication and needs to be replaced on all wireless stations and APs. Doing so, however, is not an easy process in the 802.11 WLANs because the new WEP key needs to be deployed manually. In addition, the administrators need to perform the record keeping and destruction of keys on every 802.11 device in the network, which limit the scalability of the 802.11 WLANs.

3.2.2 Confidentiality

Confidentiality in 802.11 standard is implemented through the use of a cryptographic technique offered by WEP. The WEP technique uses RC4 stream key cipher algorithm to generate a pseudo-random data sequence, which is then applied with an exclusive-or (XOR) operation to the data to be transmitted. The WEP specification supports a 40-bit key-size for the shared key although many vendors use a 104-bit key as a non standard extension. A 24-bit value known as IV is used as a seed value for initializing the cryptographic key stream. The adversary can use eavesdropping and traffic analysis to calculate certain packet fields, if same encrypted data is used over and over again. Figure 3.3 summarizes the encryption process of WEP.

Most attacks for confidentiality target vulnerabilities in the IV. The 24-bit IV is sent in clear text and its relatively small size is open to eavesdropping and traffic analysis attacks. The small size of IV combined with the static shared key is prone to traffic analysis attacks as there is a high possibility of key stream re-usage [BGW01][Wal00]. Also, the concatenation of IV and the shared key has weaknesses in generating the per-packet RC4 key [FMS01]. An adversary can attack this weakness by eavesdropping on several million packets [SIR02]. The 128-bit WEP key was publicly broken by FBI agents in about three minutes [Che05].

3.2.2.1 RC4

The working of RC4 is carried through two different algorithms: a Key Scheduling Algorithm (KSA) and a Pseudo Random Generation Algorithm (PRGA), as shown in Figure 3.4. KSA initializes the permutation of the secret internal state with a variable (40-256 bits) length key. The KSA loops N times and initializes the array S to be the identity permutation. The array S is then

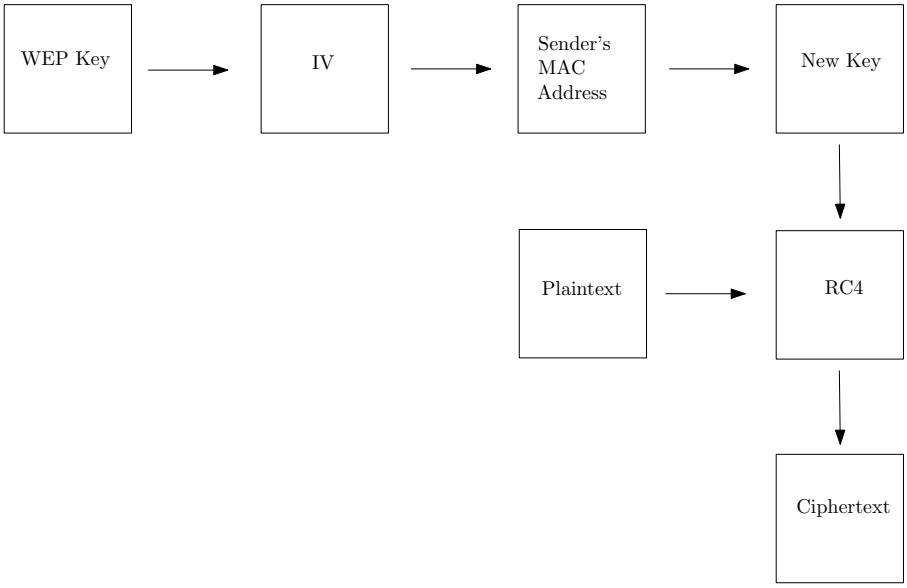

Figure 3.3: Encryption in WEP

processed for 256 iterations. After this initialization, PRGA is used to modify the state and generate a stream of bits. The algorithm is based on the use of a random permutation. The PRGA initializes two indices *i* and *j* to 0 and then in each iteration, increments *i*, adds the value of the array *S* pointed to by *i* to *j*, exchanges the values of *S[i]* and *S[j]*, and outputs the value of *S* at the location *S[i] + S[j]* (modulo 256). Each value of the array *S* is swapped at least once every 256 iterations.

Many problems with RC4 have been reported. The attack given in [FM00] targeted the keystream generated by RC4, which is slightly biased in favor of certain sequences of bytes. The authors of [MS01] used a better distinguisher that can only be used for a partial attack in broadcast applications using 2^{28} data. RC4 does not take a separate nonce alongside the key, which can ensure that encrypting the same message twice produces a different ciphertext each time. One way to address this problem is to generate each RC4 key by hashing a long term key with a unique nonce using a construction such as Keyed-Hash Message Authentication Code (HMAC) [KBC97]. The authors

PRGA(K)

```
i = 0
j = 0

While GeneratingOutput
   i = i + 1
   j = j + S[ i ]
   Swap(S[ i ], S[ j ])
   Output z = S[ S[ i ] + S[ j ] ]
endwhile
```

KSA(K)

```
for i = 0 to N - 1
   S[ i ] = i
endfor
j = 0
for i = 0 to N - 1
   j = j + S[ i ] + K[ i mod l ]
   Swap(S[ i ], S[ j ])
endfor
```

Figure 3.4: RC4 Algorithm

of [FMS01] state that over all possible RC4 keys, the statistics for the first few bytes of output keystream are strongly non-random. This non-random nature can leak information about the key. If the long term key and nonce are simply concatenated to generate the RC4 key, this long term key can be discovered by analyzing a large number of messages encrypted with this key. This vulnerability can be used to break WEP.

3.2.3 Integrity

WEP provides data integrity for wireless communications between the participants of a WLAN. The aim is to reject any message that has been modified in transit. WEP provides this integrity service through a simple encrypted Cyclic Redundancy Check (CRC) scheme. A 32-bit CRC checksum is computed on each payload prior to transmission. The resultant packet is then encrypted using the RC4 key stream and is transmitted over the wireless link. Upon the receipt of this message, the receiver performs decryption on the received payload. The receiver then recomputes the CRC and compares it with the one computed with the original message. If the checksums do not match, the received message is discarded.

Similar to confidentiality, the integrity scheme in 802.11 is vulnerable to certain attacks. The

CRC is not a secure cryptographic mechanism. An adversary can find out which bits in a 32-bit CRC will change when message bits are modified in transit. One way to protect this is to form an Integrity Check Value (ICV) by encrypting the 32-bit CRC. But it will not serve any good purpose as the encryption can not prevent the same bits from flipping. Furthermore, an adversary can arbitrarily modify or forge a packet because ICV is a linear and non-cryptographic function of the message [BGW01]. These weaknesses in integrity allows an adversary to recover the plaintext as in IP redirection and reaction attacks [BGW01] and inductive chosen plaintext attacks [Arb01].

3.2.4 Availability

The unprotected management and control frames of 802.11 WLANs provoke many DoS vulnerabilities. In order to obtain an unfair allocation of the bandwidth, an adversary can use a smaller and ultimately no backoff time and transmit legitimate messages without following the standard [KV03]. The adversary can forge the Deauthentication, Disassociation, Traffic Indication Map (TIM), or Poll messages to launch a DoS on the entire Basic Service Set (BSS) [BS03]. The Ready To Send (RTS) frame in the virtual carrier-sense scheme can be forged with an extremely large value of Network Allocation Vector (NAV), which will cause the participants to consider the channel busy [BS03][CDV03]. The Address Resolution Protocol (ARP) cache poisoning can also be used to launch a DoS [FD01]. In addition, the speed limitation of a WLAN makes it easier for an adversary to launch a DoS attack from the IP or upper layers by ICMP ping flooding [Neo03].

In Table 3.2, we map the problems with WEP to a set of corresponding security objectives.

Table 3.2: WEP Problems and Security Objectives

WEP Problem	**Compromised Security Objective**
Open system authentication is vulnerable	Authentication
SSID is vulnerable	Authentication
No mutual authentication	Authentication
Short cryptographic keys	Authentication
Non-automated key update	Key Management
Static/short IV	Confidentiality
Incorrect RC4 key scheduling	Confidentiality
Inadequate CRC-32	Integrity
Unprotected frames	Availability
ARP cache poisoning	Availability

3.3 Security Objectives in 802.11i

The IEEE 802.11i amendment addresses the security vulnerabilities in WEP through two general classes of security capabilities for 802.11 WLANs. The first class, pre-RSN security, includes the legacy security capabilities developed in the original IEEE 802.11 specification. The pre-RSN security addresses authentication through open system or shared key authentication, and confidentiality and integrity through WEP. The second class defines RSN that includes security enhancements to address all known weaknesses of WEP. In this section, we describe how 802.11i addresses security objectives in WLANs.

3.3.1 Authentication

The 802.1i amendment uses the IEEE 802.1X standard [IEE04b] to provide mutual authentication between a STA and an AS. The 802.1X uses the Extensible Authentication Protocol (EAP) [BV98][ABV04]. EAP provides a framework that allows the use of multiple methods for achieving authentication including static passwords, dynamic passwords, and public key cryptography certificates. We briefly explain EAP and 802.1X in the following sections.

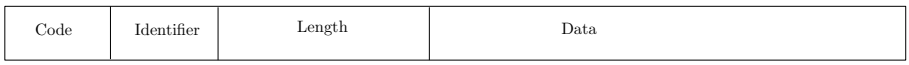

Figure 3.5: EAP Message Format

3.3.1.1 Extensible Authentication Protocol (EAP)

EAP [ABV04] is an authentication framework that is frequently used in the wireless networks and in the Point-to-Point connections. EAP provides a wide variety of authentication mechanisms known as EAP methods. The EAP messages have a similar basic format consisting of four fields: Code, Identifier, Length, and Data, as shown in Figure 3.5. The code field is one byte and can have values equal to Request, Response, Success, or Failure. The Identifier field is one byte and is provided to match Responses with their corresponding Requests. The Length field is two bytes and indicates the total number of bytes in the EAP message including the Code, Identifier, Length, and Data fields. The Data field is zero or more bytes and contains the actual data.

In addition to special types, EAP defines three basic authentication types as Message-Digest-5 Challenge (MD5-Challenge), One-Time Password (OTP), and Generic Token Card (GTC). The EAP specification requires that all EAP implementations support three special types and MD5-Challenge method. The support for the other two basic authentication types (i.e, OTP and GTC) is optional. The extensible nature of EAP has allowed the development of many EAP methods. Some well known methods are EAP-MD5, EAP-OTP, EAP-GTC, EAP-TLS, EAP-IKEv2, EAP-SIM, and EAP-AKA. In addition, a number of vendor specific methods and new proposals exist. Commonly used modern methods capable of operating in wireless networks include: EAP-TLS, EAP-SIM, EAP-AKA, PEAP, and EAP-TTLS. Requirements for EAP methods used in the wireless LAN authentication are described in [SWA05].

3.3.1.2 802.1X

802.1X [IEE04b] is an IEEE standard for port-based network access control. The 802.1X standard addresses the security of WLANs by providing an authentication framework that allows a user to be authenticated by a central authority. The authentication of 802.1X has three main participants known as supplicant (also known as client or peer or STA), authenticator (or AP), and AS as shown in Figure 3.6. The protocol used between the supplicant and the AP is typically EAP over LAN (EAPOL) and between the AP and the AS is RADIUS. This is further elaborated in Figure 3.6.

3.3.2 Key Management

In this section, we discuss key management or Key Generation and Distribution (KGD) in 802.11i. The pre-RSN 802.11 WLANs relied on WEP and did not provide any KGD mechanism. On the other hand, the IEEE 802.11i EAPOL-key exchange uses a number of keys and has two key hierarchies to divide initial key material into useful keys. The two key hierarchies are Pairwise Key Hierarchy and Group Key Hierarchy. The Pairwise Key Hierarchy addresses unicast, whereas the Group key Hierarchy targets multicast or broadcast traffic protection.

In the Pairwise Key Hierarchy, two root keys are used to generate additional keys required for various confidentiality and integrity protections. These root keys are Pre-Shared Key (PSK) and Authentication Authorization and Accounting Key (AAAK). A PSK is a static secret, which is shared between the STA and the AS using some secure channel before it needs to be used. A PMK is formed using one of the root keys. The PMK is used for the derivation of the Pairwise Transient Key (PTK) along with the MAC addresses and nonces of the STA and the AP. A Pseudo-Random Function (PRF) is used to generate the PTK from the PMK.

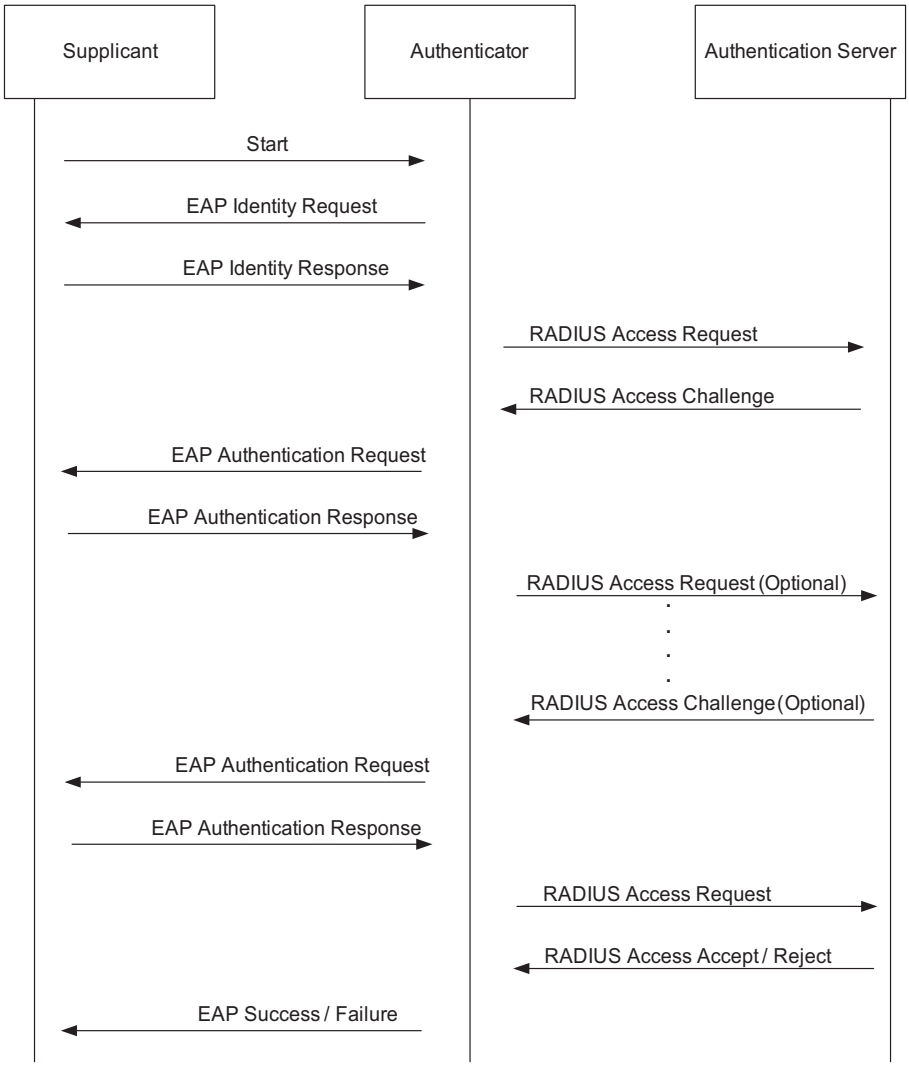

Figure 3.6: Authentication Using 802.1X

Table 3.3: Summary of Keys

Name	**Size (bits)**	**Purpose**
WEP Key	40-104	Used with WEP.
AAA Key	≥ 256	Used to derive PMK. Used with 802.1X authentication and key management.
MIC Key	64	TKIP's Michael MIC uses it to provide integrity protection.
EAPOL-KCK	128	Used for integrity protection of the key material.
EAPOL-KEK	128	Used for confidentiality of the key material.
TK	256 (TKIP), 128 (CCMP)	Used with TKIP or CCMP to provide confidentiality and integrity protection for unicast traffic.
PSK	256	Forms PMK.
PMK	256	Used to derive the PTK.
GMK	128	Used to derive the GTK.
PTK	512 (TKIP), 384 (CCMP)	Consists of EAPOL-KCK, EAPOL-KEK, and TK and provides confidentiality and integrity protection.
GTK	40-104 (WEP), 256 (TKIP), 128 (CCMP)	Used to provide confidentiality and integrity protection for multicast or broadcast traffic.

The Group Key Hierarchy consists of a Group Master Key (GMK) and a Group Temporal Key (GTK). The GMK is a random number. The GTK is derived by running a pseudorandom function over the GMK and some other parameters. The GTK is encrypted using the EAPOL-KEK assigned to the STA and protects the data from being tampered using a Message Integrity Code (MIC). The GTK may need to be updated when a session expires or when a device leaves a network. We provide a summary of keys and their usage in Table 1.

3.3.3 Confidentiality and Integrity

The IEEE 802.11i amendment addresses confidentiality and integrity through two protocols: Temporal Key Integrity Protocol (TKIP), and Counter Mode with Cipher Block Chaining MAC Proto-

Table 3.4: Confidentiality and Integrity Comparison

Security Feature	**WEP**	**TKIP**	**CCMP**
Cryptographic algorithm	RC4	RC4	AES
Authentication	Open system or Shared key	PSK or EAP with 802.1X	PSK or EAP with 802.1X
Key management	Manual	Manual or 802.1X	Manual or 802.1X
Integrity provision	CRC-32	Michael MIC	CCM
Replay protection	None	IV	IV

col (CCMP). In order to claim RSNA compliance, support for CCMP is mandatory. Support for TKIP, on the other hand, is optional. In Table 2, we present a comparison between the confidentiality and integrity approaches of pre-RSN and RSN frameworks. TKIP and CCMP are described further in the following sections.

3.3.3.1 Temporal Key Integrity Protocol (TKIP)

TKIP is a cipher suite that is employed to address the security concerns in WEP. The motivation behind the usage of TKIP is to enhance the security of WEP protocol. TKIP does not require computationally advanced encryption because it was designed to work on pre-RSN hardware. In a WLAN, TKIP addresses confidentiality through RC4 and integrity protection using the Michael message digest algorithm. In addition, TKIP employs a frame sequencing technique to prevent replay attacks. The attack of [FMS01] is countered by using a new encryption key for each frame. WEP simply concatenated its key with the IV to form a traffic key that can be attacked by an adversary. TKIP addresses this weakness by hashing the initialization vector (IV) values.

3.3.3.2 Counter Mode with Cipher Block Chaining MAC Protocol (CCMP)

802.11i uses CCMP to address confidentiality and integrity in RSNs [WHF03]. CCMP uses the Advanced Encryption Standard (AES) algorithm [Nat01] for data protection. The encapsulation of a plaintext MAC Protocol Data Unit (MPDU) is carried out in several steps. The PN (maintained for the session) is incremented to obtain a fresh PN for each MPDU. The PN and other portions of the address field are then combined to form a nonce. A CCMP header is formed by combining identifier for the TK or KeyID, and the PN. The fields in the MAC header are used to construct the Additional Authentication Data (AAD). The nonce, AAD, and plaintext are passed to CCM along with the TK in order to encrypt the data. Concatenations is performed on the CCM and packet headers and the ciphertext data to form the ciphertext.

On the receiving side, decapsulation is performed. The receiver parse the encrypted frame to reconstruct the AAD and the nonce. The reconstructed AAD and nonce along with the TK, MIC, and encrypted payload are used by the CCM to retrieve the plaintext. CCM also verifies the MIC integrity checking to ensure that the message is not modified in transit. Concatenation is then performed on the received plaintext and the frame header to form the plaintext frame. In order to provide replay protection, the PN in the frame is compared with the PN maintained for the session. The frame is discarded if the received PN is not greater than the session PN. The CCMP encryption and decryption are summarized in Figure 3.7.

3.3.4 Availability

Many DoS vulnerabilities in 802.11i exist despite the use of strong authentication and access control, confidentiality, and integrity protocols. During KGD, an adversary is able to launch a memory

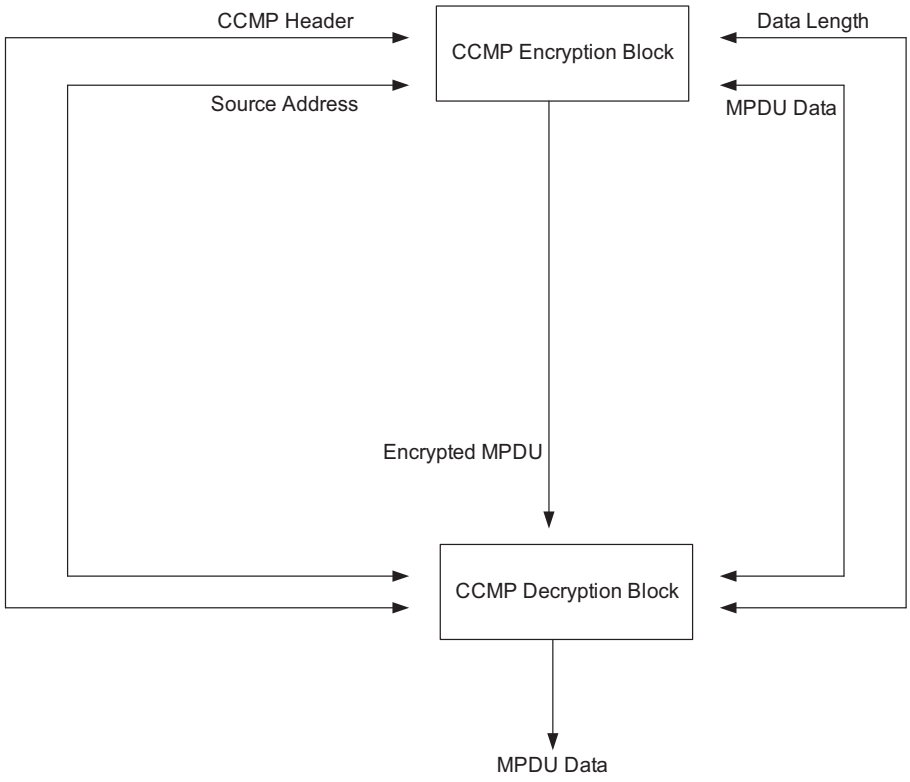

Figure 3.7: CCMP Encryption and Decryption

DoS attack by sending out numerous forged messages. The authors of [HM04] counters this attack by simply reusing the nonces in the supplicant. This approach inherently eliminates the vulnerability but may consume more computation power in the supplicant. The management frames and control frames can be forged to launch a DoS attack. The unprotected EAP messages in 802.1X authentication can be exploited by an adversary to prevent the 802.1X authentication from succeeding. In addition, the AP can be flooded by forged Association Request frames sent by an adversary. In order to address these vulnerabilities, the authors of [HM05] propose some modifications in 802.11i. They also use the idea of swapping authentication and association of [FC02].

3.4 Summary

The security objectives enforce the security policy that defines what is to be protected in a WLAN. This section explains WLAN security with a focus on its security objectives. We define authentication and access control, confidentiality, integrity, and availability as common security objectives of a WLAN. We also describe the security attacks that are induced in a WLAN when one or more of these security objectives are compromised. The IEEE 802.11 standard uses WEP to achieve security objectives in WLANs. Many vulnerabilities were discovered in WEP which led the designers to replace it with new security mechanisms defined in the IEEE 802.11i amendment. We explain how security objectives were addressed in the 802.11 standard and discuss the shortcomings of WEP in order to understand the motivation behind 802.11i and its RSN framework.

CHAPTER 4
DEVELOPING FORMAL MODELS FOR SECURITY PROTOCOLS

In this chapter, we develop the modal logic and the proof based models for the security protocol verification using the Strand Space framework. In our modal logic approach, we present the logical constructs to model a protocol's behavior in such a way that the participants can verify different security parameters by looking at their own run of the protocol. Our model captures the notion of matching parameters and provides formal proofs for the security protocol verification. We also present a proof based model that contains a generic set of proofs to establish the correctness of a security protocol.

This chapter is organized as follows. In Section 4.1, we present a Modal logic approach that simplifies the protocol analysis by avoiding the different ways a penetrator can attack a protocol. We explain a formal model for the security objectives of 802.11i in Section 4.2. In Section 4.3, we present a mathematical model that provides basis for the correctness proofs of 802.11i in the form of various lemmas.

4.1 A Modal Logic Approach Based on SSM

In this section, we lay out a model for analyzing the security protocols. Our model has a modal logic approach that is based on SSM. The presented model aims to establish a state where one principal furnishes assurance that its set of parameters matches with that of its counterpart [MFG06d, MFG06b]. We provide guarantees about certain aspects of a protocol that include, but are not limited to, data freshness, identity of the principals, message origination, etc. SSM incorporates the guaranteeing mechanism in [FHG99]. Our work is unique because unlike SSM, we do not need

to explicitly model the penetrator [MFG06c, MFG06a]. However, we do utilize some notions presented in SSM to facilitate the understanding of our proof mechanism [MFGa, MGF04a, MFGb, MGF04b].

Our approach uses simple logical formulas to represent the security primitives. The crux of the matter rests in the trust that a participant can establish by looking at its own run of the protocol. Security protocols can be thought of as consisting of a set of basic operations, which can be applied to achieve desired goals. We first define these operations in terms of predicates. We group the predicates such that they represent the basic building blocks of a security protocol. For instance, we group together predicates that represent communication (sending or receiving a message). Similarly, notion of subterm (ingredient of a message), freshness (used in nonce and the idea based on freshness like origination and unique origination), and cryptographic algebra (encrypting or decrypting any message) are categorized into their respective groups. Then, we formulate the inference rules that help us achieve the desired goals of security protocols. The inference rules are also categorized in order to define notions like subterm and to describe security protocol primitives like cryptography, message structure, etc.

We begin by laying out the framework for analyzing the authentication property of the security protocols. The underlying logical structure of our framework is based on the predicate logic. We define the syntax and semantics of our language for the security protocol verification in the following sections.

4.1.1 Syntax of the Language

We begin by laying out the framework for analyzing the authentication property of the security protocols. The meta-symbols P and Q are used to denote participants, the letter K is used to represent the encryption keys (K^{-1} for the corresponding decryption key), and the letters M, M', N and N' are used to represent messages of the language. We use $\{M\}_K$ to represent an encrypted message and $\{M\}_K^{-1}$ to represent a signed message. Moreover, $Hash(M)$ represents a one-way computationally feasible function of M. We define the following operations on messages:

- *subterm*: Subterm defines what can be analyzed from a message or a set of messages. Assuming that all the participants are capable of separating a concatenated message and decrypting a message with the known keys, we define subterm as follows. Every term is a subterm of itself, that is, $M \in subterm(M)$. Concatenates are the subterms of the concatenated term, that is, $M, N \in subterm(MN)$. Contents of an encrypted term are the subterm of the message, that is, $M \in subterm(\{M\}_K)$. Contents of a signed term are the subterm of the message, that is, $M \in subterm(\{M\}_K^{-1})$. Subterm of a set of terms is simply the union of the subterms of each of its terms.

- *superterm*: Since a participant can concatenate two messages and encrypt a message with the known key, superterm gives the idea of what can be synthesized from a message or a set of messages. Every term is a superterm of itself, $M \in superterm(M)$. Every concatenated term is the superterms of its concatenates, that is, $M, MN, NM \in superterm(M)$. An encrypted term is the superterm of its contents, that is, $\{M\}_K \in superterm(M)$. A signed term is the superterm of its contents, that is, $\{M\}_K^{-1} \in superterm(M)$. Superterm of a set of terms is

simply the union of the superterms of each of its terms. The notion of superterm can be applied recursively to find more superterms.

- *Key(P)*: It returns the secret key of a participant P.

- *Ss(N,M)*: This relation holds between two terms N and M such that it yields *true* if $N \in subterm(M)$, and $false$ otherwise.

- *Se(N,M)*: This relation holds between two messages N and M such that it yields *true* if $N = M$, and $false$ otherwise. If M, M', N, N' represent messages and K and K' are keys, then the equivalence of two messages implies:

  1. $\{M\}_K = \{M'\}_{K'} \Rightarrow M = M' \wedge K = K'$

  2. $MN = M'N' \Rightarrow M = M' \wedge N = N'$

- *Encrypt*(N, K): This relation holds between two terms N, and K, and it returns M such that $M = \{N\}_K$.

- *Sign*(N, K^{-1}): This relation holds between two terms N, and K^{-1}, such that it returns $\{N\}_K^{-1}$.

4.1.2 Propositional Constructs

The language of formula used in the cryptographic protocol world are based on the following constructs:

Cr**:** The set of messages received by a participant.

Cs**:** The set of messages sent by a participant.

Fr: The set of messages that are considered fresh.

Fo: It represents the message that originates in a participant's trace. A message may be received at a destination after being forwarded by several participants, but our analysis focuses on the originator of the message.

Fu: It represents a message that uniquely originates in a protocol run. That is, the term has been originated by only one participant in the entire run of the protocol.

Hf: This is the message that occurs first in the trace of the current authentication of a participant.

Hs: The relation that holds between two terms such that the second term is sent into the first term.

Hp: It represents the occurrence of all the messages that precede a message in the state of a participant.

Hx: The relation that holds within three terms such that the first term does not exist in any message that precedes the second message.

Er: The relation that holds for terms P, Q, N, and M, such that only participant P can reply a challenge generated by Q by extracting the secret N from M.

Hm: The relation that holds within four terms P,M, N, and N', such that a message M is received in which N is a subterm and N' is not a subterm.

Eo: The relation that holds within three terms P, M and Q such that the participant P successfully answers the challenge M generated by the participant Q.

Sec: It holds if a secret generated by a participant remains secret in the protocol.

4.1.3 Model of Computation

In order to provide rigorous semantics of the proposed logic, we lay out the model of computation for the system. A protocol is executed by a finite set of participants, $P_1, \ldots, P_n$. The participants communicate by sending messages to each other. Each participant P_i has a local state given at time t as $s_i(t)$. A run r is an execution of the protocol and can be represented as an infinite sequence of global states at integral times. For a fixed run r of the protocol, the global state at time t is defined as an n-tuple of the local states of all the participants at t. That is, $g_r(t) = (s_1(t), \ldots, s_n(t))$. Individual participants perform actions such that each action produces a transition from one state to the next. These actions include external actions only, such as $send(message, recipientList)$, $receive(message)$, and $generate(atomicTerms)$. Internal actions (such as internal computations, encryption, etcetera), on the other hand, do not contribute towards a transition in a state. A trace tr_i is the local view of an execution of the protocol by participant P_i. It can be represented as an infinite sequence of local states s_i at integral times. Alternatively, a run can be given as an n-tuple of the traces of all the participants of a protocol, that is, $r = (tr_1, \ldots, tr_n)$. The first state of a run r is assigned some time $t_r \leq 0$ and the initial state of the current authentication is at $t = 0$.

Each participant maintains T_{ot}, a set of all atomic terms, and X_t, a set of received messages up to time t, in its local state $s_i(t)$. $T_{ot} = T_t \cup K_t$, where T_t is the set of atomic terms such as participant names (ids) and generated secrets (nonces) up to time t, and K_t is the set of keys that P_i possesses up to t. If $\{M\}_k \in X_t$ and $k^{-1} \in K_t$, then $M \in X_t$. Similarly, if $MN \in X_t$ then $M, N \in X_t$. The local state maintains a counter, $messageCount$, for the protocol messages in the current authentication. This counter increments only when a message is sent or is received by a participant in a protocol run. The local state also contains a local history of all the actions the participant has performed, along

with a set of computations (encryption and decryption with available keys, hashes, signatures) available to P_i.

4.1.4 Semantics

Here we describe the semantics of the constructs of the proposed logic. Let Φ be the set of atomic propositions. Fix a system R and an *interpretation* π that maps each proposition $p \in \Phi$ to the set of global states $\pi(p)$ in R at which p is true. Truth of a formula φ in global state $g_r(t)$, written $g_r(t) \models \varphi$, is inductively defined below.

$g_r(t) \models p$ iff $g_r(t) \in \pi(p)$ for $p \in \Phi$,

$g_r(t) \models \varphi \wedge \psi$ iff $g_r(t) \models \varphi$ and $g_r(t) \models \psi$, and

$g_r(t) \models \neg\varphi$ iff it is not the case that $g_r(t) \models \varphi$.

Now semantics for the remaining constructs is as follows.

Received Message: $g_r(t) \models Cr(P, M)$ iff for message M at time t in r, $receive(M)$ appears in P's local history. In other words, $receive(M)$ appears in the trace of the participant P, tr_P at t. It also results in an increment of $messageCount$ in $s_P(t)$.

Sent Message: $g_r(t) \models Cs(P, M)$ iff for message M at time $0 \leq t' \leq t$ in r, P performs $send(M, recipientList)$ and $M \in \{superterm(X_{t'} \cup T_{ot'})\}$. Moreover, $s_P(t)$ increments its local counter $messagCount$ by one.

Fresh Message: It is necessary as well as sufficient for freshness that the message has never been sent before the initial state of the current authentication of the protocol. Therefore, if $M'(r, t)$ represents the set of all messages sent by any participant in time t in r, $g_r(t) \models Fr(M)$ iff, for all participants at $t' < 0$, $M \notin subterm(M'(r, t'))$.

Term Origination: Term origination means that a message is sent by a participant such that it contains a term, which appears first in the trace of its current authentication. $g_r(t) \models Fo(M, P)$ iff for message M at time t in r, P performs $send(superterm(M), recipientList)$ and for tr_p at $0 \leq t' < t$, $M \notin subterm(M'(tr_p, t') \cup X_{t'})$, where $M'(tr_p, t')$ represents the set of all messages sent in the trace tr_p in t'.

Unique Origination: A message M uniquely originates by a participant P in $g_r(t)$ if no one else originates M in r. $g_r(t) \models Fu(M, P)$ iff $g_r(t) \models Fo(M, P)$ and it is not the case that $g_r(t') \models Fo(M, Q)$, where $t' \geq 0$ and $Q \neq P$.

First Messages: A message is termed first message in the trace of a participant if it lies first in its trace of the current authentication. If $M_0 \in \{M'(tr_p, t) \cup X_t\}$ represents the message sent or received by the participant P in the trace tr_p at time $t = 0$, then $g_r(t) \models Hf(M, P)$ iff $M = M_0$.

Sending a Subterm: $g_r(t) \models Hs(M, N)$ iff $g_r(t) \models Cs(P, M)$ and $N = subterm(M)$.

Preceding Messages: A set of messages precede a message M if they occur (sent or received) before M occurs in the current authentication of a particular trace. $g_r(t) \models Hp(N, M)$ iff $N = \{M'(tr_p, t') \cup X_{t'}\}$, $0 \leq t' < t$, and M is the message sent or received in tr_p at t.

Non-existence in the Preceding Messages: A message is termed non-existent in the preceding messages if it does not occur in any of those messages in the trace of a participant. $g_r(t) \models Hx(N, M, P)$ iff $N \notin subterm(M'(tr_p, t') \cup X_{t'})$, $N \in subterm(M)$, and $0 \leq t' < t$, where $M'(tr_p, t')$ represents the set of messages sent in the trace tr_p in time t' and M is the message sent or received in tr_p at t.

Challenge Reply: Only the participant who can extract the secret challenge message can reply the challenge. $g_r(t) \models Er(P, Q, N, M)$ iff $g_r(t) \models Fu(N, Q)$, $g_r(t) \models Fr(N)$, and M is an encrypted

message of the form $\{M'\}_K$ such that $N \in subterm(M')$, $K^{-1} \in K_t$ of P (and Q in symmetric cryptography) and $K^{-1} \notin K_t$ of any other participant.

Receiving Specific Sub-messages: A participant receives a messages in which it expects certain sub-messages. $g_r(t) \models Hm(P, M, N, N')$ iff $receive(M)$ appears in P's local history such that $N \in subterm(M)$ and $N' \notin subterm(M)$.

Assurance in a Received Message: A participant gains assurance that its challenge has been successfully answered by the desired participant. $g_r(t) \models Eo(P, M, Q)$ iff $g_r(t') \models Er(P, Q, N, M)$, $g_r(t) \models Hm(Q, N', N, M)$, $P \neq Q$, and $t' < t$. That is, the participant Q has uniquely originated a fresh secret N in a message M such that only P can extract the secret from the message, and Q has received a reply in which N is present then Q gets convinced that the participant P must have received its message M and decrypted it. Notice here that the message Q receives (N') must not contain M as a subterm, otherwise any participant could simply forward Q's challenge back to him without even letting P know about the communication at all.

Secrecy: A uniquely originated secret within an encrypted message of a protocol maintains its secrecy if the secret is never sent out in any form other than that of the encrypted message of the protocol. Notice that this definition is sufficient but not necessary for secrecy. A superterm of the protocol message containing encrypted secret also maintains its secrecy. Hence, we posit that $g_r(t) \models Sec(P, M)$ iff M uniquely originates in an encrypted message M' with the decryption keys available to only intended recipients and at time $t' \geq t$ in r, no participant Q performs $send(M'', recipientList)$ such that $M \in \{subterm(M'') \backslash M'\}$. Here, $A \backslash B$ means the set difference of A and B.

4.1.5 The Proposed Logic

The proposed theory captures the entire notion about how we prove a cryptographic protocol correct. The security protocol is defined in terms of generic building blocks. The proposed postulates then establish the properties that each building block should possess in order to provide some guarantees. These properties are essential in establishing the confidence in the working of security protocols. We use modus ponens as the inference rule in our logic. That is,

$$\varphi \wedge (\varphi \rightarrow \psi) \vdash \psi$$

The following rules possess descriptive nature about information contained in a message. These postulates tell us about sending any message that contains some nonce as a subterm [P_1], nonexistence of a subterm inside all the messages preceding a message [P_2], and a message having specific terms as its subterms [P_3]. Notice that the postulate [P_2] contains the idea of term origination of a participant's current authentication. If the term lies on the first node in the trace of current authentication, then that node may serve as the originating node for that term. Whereas, if it is not the first node, one needs to check all the preceding nodes in order to find out if they contain that term.

[P_1] $S\,s(N, M) \wedge Cs(P, M) \rightarrow Hs(M, N)$

[P_2] $\neg Hf(M, P) \wedge \nexists M' \in N'((Hp(N', M) \wedge (S\,s(N, M'))) \rightarrow Hx(N, M, P)$

[P_3] $Cr(P, M) \wedge S\,s(N, M) \wedge \neg S\,s(N', M) \rightarrow Hm(P, M, N, N')$

The following axioms describe the notion of origination and unique origination of a message. Our notion of origination captures simple information about the first sender of a message. [P_4] states that a term sent in the first message of a trace originates in that message. [P_5] implies that a node

serves as the originating node for a term if the term is sent in the message such that it has not been sent in any message preceding that node.

$[P_4]$ $Hf(M,P) \wedge Hs(M,N) \rightarrow Fo(N,P)$

$[P_5]$ $\neg Hf(M,P) \wedge Hs(M,N) \wedge Hx(N,M,P) \rightarrow Fo(N,P)$

A received signed message originates from where it is signed.

$[P_6]$ $Cr(P,M) \wedge Se(M, Sign(M', K^{-1})) \wedge Se(Key(Q), K^{-1}) \rightarrow Fo(M,Q)$

If a message uniquely originates in a protocol run, then it uniquely originates at its originating trace $[P_7]$. If *Part* represents the set of all participants playing in a protocol, then $\forall Q \in Part, Q \neq P$:

$[P_7]$ $Fo(N,P) \wedge \neg Fo(N,Q) \rightarrow Fu(N,P)$

A message is fresh if its ingredients contain a fresh term.

$[P_8(a)]$ $Fr(M) \rightarrow Fr(superterm(M))$

$[P_8(b)]$ $Fr(M) \rightarrow Fr(Hash(M))$

The following are the concluding rules that contain guarantees within them about messages reaching at a destination.

$[P_9]$,

$Fu(N,Q) \wedge Fr(N) \wedge Ss(N,N') \wedge Se(M, Encrypt(N', K)) \wedge Se(Key(P), K^{-1}) \rightarrow Er(P,Q,N,M)$

Finally,

$[P_{10}]$ $Er(P,Q,N,M) \wedge Hm(Q,N',N,M) \rightarrow Eo(P,M,Q)$

That is, if a fresh message is originated such that only the participant P is able to decipher it and obtain the secret N, then only P can successfully answer the challenge, $[P_9]$. $[P_{10}]$ implies if only the participant P is able to answer some challenge and the desired answer of the challenge is received in a protocol then P must have received and opened the challenge. Notice here that the guarantees

we provide are somewhat strict in the sense that even after getting the desired response from a participant, the receiver does not provide guarantee about the sender of the response. However, receiver does guarantee that the intended participant has received and decrypted the challenge.

A protocol typically contains some secret parameter in its one or more messages that is used either to help achieve proper authentication, or simply to distribute the secret to other participants. The secret must be the subterm of an encrypted message such that only the legitimate participants hold the proper decryption key in order to open the message and extract that secret from it. The protocol must guarantee that its secret never appears in any message other than those purposely encrypted messages that contain the secret. Provided the keys with which the messages containing the secret are encrypted do not get compromised, one needs to check if the secret appears in any form other than that of the encrypted secret messages. That is, if M' is the set of messages of the protocol in which secret N occurs, K' is the set of secret keys known to only legitimate participants, and M'' is the set of messages sent by any participant such that $M' \cap M'' = \emptyset$. Then in order to preserve secrecy of N generated by participant P, $\forall m \in M'', M \in M', K \in K'$

$[P_{11}]$ $Fu(N, P) \wedge Ss(N, N') \wedge Se(M, Encrypt(N', K)) \wedge \neg Cs(Q, m) \rightarrow Sec(P, N)$

such that $N \in \{subterm(m) \backslash M\}$. Notice that the messages of the form $superterm(M)$ (for example, $\{MX\}_k, MX$) may occur in M'' but as long as $N \notin subterm(X)$, the secrecy of N never gets compromised.

4.1.6 Soundness

We prove that our logical schema is sound with respect to the given semantics and the model of computation. We show that our set of formulas are valid and the rules to derive new axioms from

the valid axioms preserves the truth. If Γ represents a set of formulae and φ is a formula, then soundness implies that if $\Gamma \vdash \varphi$ then $\Gamma \models \varphi$. That is, if φ can be derived from Γ then φ satisfies Γ (whenever all of Γ is true, φ is true as well). The only rule that we use to derive φ is modus ponens. There are two cases:

1) $\Gamma \vdash \varphi$ such that φ is a theorem or $\varphi \in \Gamma$. In this case $\Gamma \models \varphi$ is trivially true.

2) If modus ponens is used to derive φ then by induction on the structure of the derivation and definition of truth conditions, $\Gamma \models \varphi$.

Now we show that our set of formulas are valid. We take an example postulate P_5 and sketch the proof of its validity in the following.

$$\neg Hf(M, P) \wedge Hs(M, N) \wedge Hx(N, M, P) \rightarrow Fo(N, P)$$

This conditional is true in a state $g_r(t)$ if either the premises are false, that is, $g_r(t) \not\models (\neg Hf(M, P) \wedge Hs(M, N) \wedge Hx(N, M, P))$, or the conclusion is true, $g_r(t) \models Fo(N, P)$. Therefore, we only need to check $Fo(N, P)$ at $g_r(t)$ if $g_r(t) \models \neg Hf(M, P) \wedge Hs(M, N) \wedge Hx(N, M, P)$. In this case, since the antecedent is true at $g_r(t)$, all of its conjuncts must be true in that state. So $g_r(t) \models \neg Hf(M, P)$ implies that there exists a message $M_{t'} \in \{M'(tr_p, t') \cup X_{t'}\}$ such that $0 \leq t' < t$. Here $M'(tr_p, t') \cup X_{t'}$ represents the set of messages sent or received by the participant P in the trace tr_p in time t'. Now $g_r(t) \models Hs(M, N) \wedge Hx(N, M, P)$ implies $N \in subterm(M)$ and a participant P performs $send(M, recipientList)$ operation such that $N \notin subterm(M'(tr_p, t') \cup X_{t'})$, which leads us to conclude that $g_r(t) \models Fo(N, P)$. Hence, axiom P_5 is true at all states $g_r(t)$.

4.2 Formal Model of Security Objectives for 802.11i

We develop a formal model of security objectives for 802.11i in the following lines. Our formal model is built using the SSM [FHG99].

We consider a set A, which contains all possible exchanged messages in 802.11i. We also use the word *subterm* with members of A. We represent the transmission of a term a_0 with a positive sign as $+a_0$, whereas the reception of a term a_0 is represented as $-a_0$. In this way, a signed term is defined as a pair $\langle \sigma, a \rangle$. Here, $\sigma \in \{+, -\}$ and $a \in M$, where M is a set of all possible messages of the 802.11i protocol.

In the context of 802.11i, the set of finite sequence of signed terms is represented as $(\pm M)^*$. A strand space over M consists of a set Σ, whose elements are called strands, together with a trace mapping $tr : \Sigma \rightarrow (\pm M)^*$. The trace mapping tr associates each strand in Σ with a sequence of signed terms.

For 802.11i, the nodes consists of client (or station or peer) nodes, access point nodes, RADIUS server nodes, and penetrator nodes. Formally, a *node* is a pair $\langle s, i \rangle$ with $s \in \Sigma$, and i is an integer with $1 \leq i \leq |tr(s)|$. The set of all the nodes is donated by N. For a node $n = \langle s, i \rangle$, where $tr(s) = \langle \sigma_1, u_1 \rangle ... \langle \sigma_k, u_k \rangle$, $term(n)$ is defined as $\langle \sigma_i, u_i \rangle$.

There is an *edge* $n_1 \rightarrow n_2$ if and only if $term(n_1) = +a$ and $term(n_2) = -a$ for some $a \in M$. The edge $\rightarrow$ represents a potential causal link between two strands. When $n_1 = \langle s, i \rangle$ and $n_2 = \langle s, i + 1 \rangle$ are nodes, then there is an edge $n_1 \Rightarrow n_2$. The edge $\Rightarrow$ indicates that n_1 is an immediate causal predecessor of n_2. In a similar fashion, $n' \Rightarrow^+ n$ implies that n' precedes n (not necessarily immediately) on the same strand. A term t occurs in $n \in N$ if and only if $t \sqsubset term(n)$.

The node $n \in N$ is an entry point for the term t if $term(n) = +t$ and whenever $n' \Rightarrow^+ n$, $term(n') \notin term(n)$. A term t *originates* on $n \in N$ if and only if n is an entry point for t. A term t *uniquely originates* if and only if t originates on a unique $n \in N$. It can be seen that N, together with both sets of edges $n_1 \rightarrow n_2$ and $n_1 \Rightarrow n_2$, is a directed graph $\langle N, (\rightarrow \bigcup \Rightarrow \rangle$.

A *bundle* represents a full protocol exchange. It consists of a number of strands hooked together where one strand sends a message and another strand receives the same message. Intuitively, a bundle is a portion of a strand space large enough to represent at least a full protocol exchange. Bundle has a natural causal precedence relation relative to which inductive arguments are carried out. A bundle is a finite acyclic subgraph that captures the natural causal precedence relation among nodes as defined by the edges $\rightarrow$ and $\Rightarrow$. For a given strand space Σ, let $B = \langle N_B, (\rightarrow_B \bigcup \Rightarrow_B \rangle$ be a subgraph of $\langle N, (\rightarrow \bigcup \Rightarrow \rangle$. The graph B is a bundle if:

1. B is finite,
2. if $n_2 \in N_B$ and $term(n_2)$ is negative, then there is a unique n_1 such that $n_1 \rightarrow_B n_2$,
3. if $n_2 \in N_B$ and $n_1 \Rightarrow n_2$, then $n_1 \Rightarrow_B n_2$,
4. B is acyclic.

We use the bundle-height to establish guarantees about the client and the access point. The bundle-height of a strand is the largest i such that $\langle s, i \rangle \in$ bundle.

The set $T \subseteq M$ is the set of texts (representing the atomic messages). The set $K \subseteq M$ contains cryptographic keys disjoint from T. The term $\{g\}_k \in M$ represents the encryption of the term $g \in T$ using $k \in K$. The *subterm relation* $\sqsubset$ is inductively defines as the smallest relation such that:

1. any term $m \in M$ is a subterm of itself,

2. a term $m \in M$ is a subterm of $\{g\}_k$, if m is a subterm of g,

3. a term $m \in M$ is a subterm of gh, if m is a subterm of g or h.

The authentication is represented in this framework as:

$\forall C \forall s \exists s'.C - height(s) = i \wedge \phi(s) \Longrightarrow C - height(s') = j \wedge \psi(s, s')$

In the above formula, $\phi(s)$ says the type of strand of s. For example, s can be an initiator strand or a responder strand. The statement $\psi(s, s')$ tells about the type of strand s' and the data values that must be shared between s and s'.

Similarly, secrecy can be written as:

$\forall C \forall s \forall n.\phi(s) \wedge n \in C \Longrightarrow term(n) \notin I_K[S]$

Secrecy establishes that none of the values in S can be disclosed.

4.3 Building Lemmas

In this section, we present lemmas of SSM that will help us in our proof process.

Lemma 1: If C is a bundle then every non-empty subset of the nodes in C has $\preceq_C$ minimal members.

Proof: $\preceq_C$ is regarded as expressing causal precedence. In terms of SSM, this can be represented as $n \preceq_S n'$ holds only when n's occurrence contributes to the occurrence of n'. The existence of minimal members in non-empty sets serves as an induction principle that is related to the one laid out by [Pau98, Sch97]. Intuitively, the arguments about $\preceq_C$ minimal elements are focussed not only on the participant's knowledge but the participant's temporal knowledge also.

Lemma 2: If C is a bundle and $S \subseteq C$ is a set of nodes such that: $\forall m, m'$ if the unsigned term of m is equal to the unsigned term of m', then this implies that $m \in S$ iff $m' \in S$.

Proof: This Lemma says that if n is a $\preceq_C$ minimal member of S, then the sign of n is positive. If the term of n was negative, then the bundle property will cause $n' \rightarrow n$ for some $n' \in C$ and the unsigned term of n will be equal to the unsigned term of n'. This makes $n' \in S$ violating the minimal property of n.

Lemma 3: If C is a bundle, $a \in A$, and $n \in C$ is a $\preceq_C$ minimal element of $\{m \in C : a \sqsubset term(m)\}$, then the node n is an originating occurrence for a.

Proof: Since n is a member, $a \sqsubset term(n)$. Lemma 2 says that the sign of n is positive. If $n' \Rightarrow^+ n$, then by applying the bundle properties we will have $n' \in C$. Hence from the minimality property of n, $a \not\sqsubset term(n')$. This implies that n is originating for a.

Lemma 4: A nonce t_0 originates at a node n_0.

Proof: This is a generic lemma that will be frequently used to build guarantees about a specific term. In this proof, we will provide conditions that will guarantee this lemma to hold true. Let us call the term of n_0 to be v_0. First, we notice the sign of n_0. If this is positive, we can proceed. Otherwise, the term t_0 can not be originating because a negative v_0 represents the reception of a term.

Next, we check that $t_0 \not\sqsubset$ of n', where n' is a node n_x that preceded n_0 on the same strand. if term(n') is equal to v_1, then we need to check that $t_0 \neq t_1$. Here, t_1 is a nonce and $t_1 \in v_1$. The guarantee that $t_0 \neq t_2$, where $t_2 \in v_1$ and $t_2 = v_1 - t_1$, is established on the basis that set of nonces is unique and the algebra is freely generated. The contents of a received matches are added into the parameter list and this matching parameters can guarantee the truth about $t_0 \neq t_1$.

Lemma 5: The set $S = \{n \in C : t_0 \sqsubset term(n) \wedge v_0 \not\sqsubset term(n)\}$ *has a* $\preceq$ *minimal node* n_0*. The node* n_0 *has a positive sign and does not originate on a penetrator strand.*

Proof: We use this lemma in our proofs to guarantee that a term is not originated on a penetrator strand. In this proof, we provide all possible cases where the term t_0 can be originated on the penetrator strand and establish the truth about this term.

In order to prove the sign of the term t_0, we consider a node that lies on the receiver side say n_1. If this node n_1 contains t_0 but not v_0, then S is not empty. This implies that S has at least one $\preceq$ minimal element n_0 by Lemma 1. Lemma 2 says that sign of n_0 is positive.

Now, the question remains whether n_0 lie on the penetrator strand say p. In the following lines, we present all possible penetrator strands to analyze this case.

M. The trace of such a penetrator strand has the form $\langle +t \rangle$ where $t \in$ set of terms. In our case, it corresponds to checking whether the subject term originates on this type of strand or not. This can be proven by the help of Lemma 4, which states that this term uniquely originates on the regular node n_0.

F. This penetrator strand has the form $\langle -g \rangle$. A node with a positive sign does not originate on this strand and can be safely assumed not to be a victim of this strand.

T. This strand has the form $\langle -g, +g, +g \rangle$. Similar to the previous strand, this strand is receiving a term initially. Any node with a positive sign does not originate on this type of strand.

C. The C strand is represented as $\langle -g, -h, +gh \rangle$. Again, the first node is negative. Any positively occurrence node can not lie on this strand.

K. The trace of this strand is represented as $\langle +K_0 \rangle$. Here, $K_0 \in K_P$ where K_P is a set of keys possessed by the penetrator. A term t_0 can not be equal to a term K_0 unless the term $t_0 \in K$ where K is a set of all possible keys. The algebra freeness assumption guarantees and prevents any nonce to be accidentally or deliberately equal to a cryptographic key.

E. This strand has a trace of the form $\langle -K_0, -h, +\{h\}_{K_0} \rangle$. If the subject term $t_0 \sqsubset \{h\}_{K_0} \wedge v_0 \not\sqsubset \{h\}_{K_0}$. The term $t_0 \neq \{h\}_{K_0}$ implies $t_0 \sqsubset h$. Also, $v_0 \not\sqsubset h$ that guarantees that the positive node is not minimal in S.

D. This trace has the form $\langle -K_0^{-1}, -\{h\}_{K_0}, +h \rangle$. If the positive node is minimal in S, then $v_0 \not\sqsubset h$. But $v_0 \sqsubset \{h\}_{K_0}$. We can use the free encryption assumption here that says that if $h = t_0 t_1 t_2$ and $K_0 = K_{participant(A)}$, then there exists a node n, which is the first on this strand and have a term $K^{-1}_{participant(A)}$. The assumptions about the key state guarantee that $K^{-1}_{participant(A)} \notin K_P$. $K^{-1}_{participant(A)}$ originates on a regular node.

S. The S penetrator strand is represented is $\langle -gh, +g, +h \rangle$. There are two cases to be considered here. Both cases are symmetrical. In the first case, we have to consider that $term(n_0) = g$, whereas in the second case $term(n_0) = h$. Lets proceed with the first case.

Since $term(n_0) \in S$, $t_0 \sqsubset g$ and $v_0 \not\sqsubset g$. This makes $v_0 \sqsubset h$. By the minimality of n_0, $v_0 \sqsubset gh$.

However, $v_0 \neq gh$, this implies that $v_0 \sqsubset h$.

If we consider $T = \{m \in C : m \prec n_0 \wedge gh \sqsubset term(m)\}$. Every member of T is a penetrator node, because no non-penetrator node contains a subterm gh where h contains any encrypted subterm. T is non-empty and has a minimal member m by Lemma 1. This minimal member m has a positive sign by Lemma 2. Now, we have to consider all possible penetrator strands where m can be present.

M. m is a minimal member and thus can not lie on this strand.

F. By the definition of this type of strand, m can not lie on this strand.

T. Since m is a minimal member, it can not lie on this strand.

K. A minimal member of T can not lie on this strand.

S. $gh \sqsubset term(m)$, where m is a positive node on a strand p' of kind S. This can be rejected again by the minimality property.

E. $gh \sqsubset term(m)$, where m is a positive node on a strand p' of kind E. This leads to the contradiction of minimality of m in T.

D. $gh \sqsubset term(m)$, where m is a positive node on a strand p' of kind D. By the minimality property, this strand can be rejected.

C. If $gh \sqsubset term(m)$, where m is a positive node on a strand p' of kind D and m is minimal in T. Then gh = term(m). This will make the trace of p' to be equal to $\langle -g, -h, +gh \rangle$. This is contradicting the minimality of n_0 in S.

From the application of all such strands, it can be inferred that the node n_0 lies on a regular strand.

The following lemma is a generic one that will help us prove when will a node precede another node.

Lemma 6: A node n_1 precedes a node n_2 on t and term(n_1) = t_0.

Proof: In order to prove this, we have to focus on the term say $t_0 \in n_1$. We also take t_1 to be equal to a nonce value such that $t_1 = n_1 - t_0$. Next, we have to identify a node say n_0 at which the term t_1 originates. As t_1 is chosen to be a nonce, by the basic protocol assumptions it is fair to assume that this value will be uniquely originated in Σ.

Next we have to identify v_0 such that $v_0 \sqsubset term(n_0)$ and $v_0 \not\sqsubset term(n_2)$. This will establish that $n_0 \neq n_2$. We can use this fact to infer that t_1 does not originate at n_2. This implies that there is a node n_1 preceding n_2 on the same strand such that $t_1 \sqsubset term(n_1)$. Now, we can apply the minimality property of n_2 to say that $v_0 = t_0 \sqsubset term(n_1)$. Since no regular node contains an encrypted term as a proper subterm, $t_0 = term(n_1)$.

Lemma 7: A regular strand t containing the node n_1 and n_2 is an initiator strand and is contained in C.

Proof: In order to prove this, we have to establish the fact that n_2 is a positive non-penetrator node that comes after a node n_1. Assume that n_1 is of the form $\{xyz\}_K$. If T is a responder strand, it will contain a negative node of the above mentioned form. This will establish the statement of the lemma. Furthermore, if the last node of t is contained in C, it mush have a C-height of the number of nodes included in C.

4.4 Summary

This chapter contains the modal logic and the proof based models for the security protocol verification that we developed using the Strand Space framework. In our modal logic approach, we present the logical constructs to model a protocol's behavior in such a way that the participants can

verify different security parameters by looking at their own run of the protocol. In our proof based model, we present a generic set of proofs to establish the correctness of a security protocol. In addition, we explain a formal model for security objectives of 802.11i. We present a mathematical model that provides basis for the correctness proofs of 802.11i. We also present various lemmas that are going to provide foundation for our proofs in Chapter 5 and Chapter 6.

CHAPTER 5
FORMAL VERIFICATION OF 802.11i

In this chapter, we perform the formal verification of the 802.11i protocol and prove the authentication of the proposed 802.11i protocol. We start with explaining different stages of 802.11i in Section 5.1. We separate the stages using strong authentication techniques from those that do not employ strong cryptographic mechanisms. Next, we model these authentication stages of 802.11i in our proof based approach in Section 5.2. In Section 5.3, we carry out the formal verification of these stages using our proof based approach. We then present the high level abstraction of a protocol run. We consider a penetrator, empower its capabilities, and evaluate the authentication of the proposed protocol. Given the constraints and suggestions of the proposed architecture, we state convincingly that any attempt to defy the authentication mechanism of the 802.11i protocol will not be successful. We further describe a situation where modifications to our model will lead to a successful intrusion. Our choice of SSM as a verification framework, is based on its simplicity, elegance, precision of results, and ease of developing simple and powerful proofs even without automated support.

5.1 Authentication in 802.11i

IEEE 802.11i defines RSNA establishment procedure to address weaknesses in the open system authentication and the shared key authentication of the 802.11 standard. This section describes the stages involved in an RSNA establishment, which guarantees strong mutual authentication and generate fresh TKs for the data confidentiality protocols. The entities involved in an RSNA establishment are supplicant (STA), authenticator (AP), and authentication server (AS). For the

purpose of authentication analysis, we divide the RSN operations of 802.11i into five different stages as shown in Figure 5.1.

5.1.1 Discovery

The first phase in establishing RSNAs is discovery. The AP either broadcasts its 802.11i security policies through Beacons or responds to Probe requests. The Beacon and Probe Response of an AP contains its SSID and RSN Information Elements (RSN IEs). The RSN IE carries information such as enabled authentication suites, unicast suites, and multicast suites. The station uses Probe requests or advertised Beacon frames of an AP to discover the existence of a network. The discovery phase permits a station to learn the security capabilities of the AP. These security capabilities include authentication mechanism for the mutual authentication of the AP and the AS, key management mechanism, and confidentiality and integrity protocols for protecting the unicast traffic. The multicast traffic is dictated by the AP.

5.1.2 Open System and EAP-802.1X Authentication

The second phase in the establishment of an RSNA is authentication where the STA and AP prove their identities to each other. The authentication stage is further divided into 802.11 open system authentication and EAP-802.1X authentication. Open system authentication is weak and is included only to provide backward compatibility. The 802.1X standard is used to provide mutual authentication between a STA and an AS. 802.1X uses the EAP framework that allows the use of multiple methods for achieving authentication such as static passwords, dynamic passwords, and public key cryptography certificates.

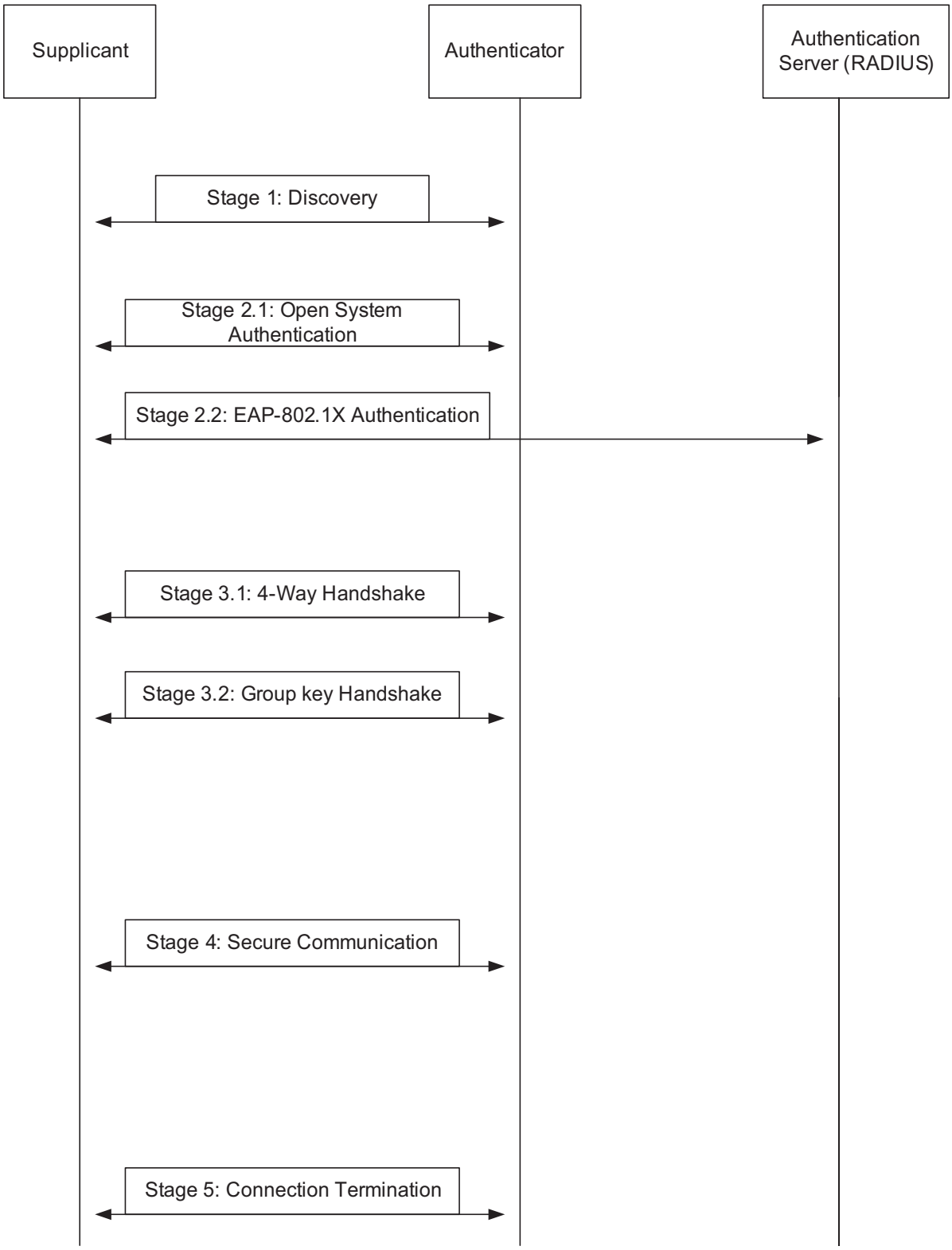

Figure 5.1: RSNA Stages of 802.11i

In EAP-802.1X, the AP acts as a pass through server between the STA and the AS and blocks non-authentication traffic between the STA and the AS until the authentication is complete. The authentication phase provides mutual authentication of a STA and an AS and delivers the Master Session Key (MSK) to the AP and, sometimes, to the STA. If an RSNA negotiated the PSK AKM during the discovery, the authentication phase is skipped entirely. The shared key has already been distributed and installed in an out-of-band manner to provide an implicit authentication. The message exchange of the EAP-802.1X authentication is explained in Section 3.3.1.2

Intuitively, a successful authentication means that the peer and the authenticator verify the identity of each other and generate some shared secret for future data transmissions. In Figure 5.1, we illustrate an abstract model of the 802.11i's authentication. After the authentication stage, the peer and the RADIUS server have authenticated each other and generate a common secret called the Master Session Key (MSK). The peer uses this MSK to derive a Pairwise Master Key (PMK). The Authentication, Authorization and Accounting (AAA) key material on the RADIUS server is transferred to the authenticator to derive the same PMK in the authenticator.

A mechanism called pre-authentication enables roaming STAs to improve their performances. Pre-authentication enables a client to establish a PMK security association to an access point with which the client has yet not been associated. Intuitively, the first time a client associates to the network, it must do a full authentication. The client can then pre-authenticate to a AP, if it knows where it will roam. This procedure creates a Pairwise Master Key Security Association (PMKSA) between the STA and the remote AP. The PMKSA is cached on both the STA and the remote AP. When the STA enters within range of the previously remote AP, authentication procedure is skipped and a 4-Way Handshake takes place. The advantage is that the client reduces the time that it is

disconnected from a network. However, pre-authentication adds an extra load to the authentication server. The caching of PMKSA can also be enabled to improve performance regardless of the pre-authentication and allows the AP and the STA to skip the IEEE 802.1X and the EAP authentications and start the 4-Way Handshake.

5.1.3 Key Generation and Distribution

After authentication, the participants perform several operations that cause cryptographic keys to be generated and installed on the AP and the STA. The purpose is to derive and install keys that will be used in future secure data transfer. Intuitively, in IEEE 802.11i, the Key Generation and Distribution (KGD) confirms the existence of the Pairwise Master Key (PMK) between the STA and the AP, ensures the freshness of the security association keys, derives the keys to be used in future data traffic, install these traffic keys in the AP and STA, distributes a group key for multicast and broadcast traffic, and confirms the selection of the cipher suite. In order to achieve all these objectives, a 4-Way Handshake and a Group Key Handshake take place.

5.1.3.1 4-Way Handshake

The 4-Way Handshake confirms the existence of the PMK, the liveness of the peers and the selection of the cipher suite. After a successful handshake, a fresh Pairwise Transient Key (PTK) for each subsequent session is generated. In [FMG07, FMG06], we analyze the handshake after a shared PMK is achieved and before the data communication begins.

As shown in Figure 5.1, the 4-Way Handshake involves a frame exchange between the STA and the AP. The earlier EAP exchange has provided the shared secret key PMK. The 4-way handshake

must be executed for a successful RSNA despite the fact that PMK is derived from authentication, reused from a cached PMK, or configured using a PSK. A Pairwise Transient Key (PTK) is generated by concatenating PMK, AP nonce (ANonce), STA nonce (SNonce), AP MAC address and STA MAC address. The product is then put through a cryptographic hash function. The PTK (64-bit) is divided into five separate keys: 16 bytes of EAPOL-Key Encryption Key (KEK), 16 bytes of EAPOL-Key Confirmation Key (KCK), 16 bytes of Temporal Key (TK), 8 bytes of Michael MIC Authenticator Tx Key, and 8 bytes of Michael MIC Authenticator Rx Key. The KEK is used by the AP to encrypt additional data sent (in the 'Key Data' field) to the client (for example, the RSN IE or the GTK). The KCK is used to compute MIC on WPA EAPOL Key message. The TK is used to encrypt/decrypt unicast data packets. The Michael MIC Authenticator Tx Key is used to compute MIC on unicast data packets transmitted by the AP. The 8 bytes of Michael MIC Authenticator Rx Key is used to compute MIC on unicast data packets transmitted by the station. The Michael MIC Authenticator Tx/Rx Keys provided in the handshake are only used, if the network is using TKIP to encrypt the data. The handshake also yields the GTK (Group Temporal Key), used to decrypt multicast and broadcast traffic.

In the first message, the authenticator sends a nonce to the supplicant. This is referred to as ANonce. The supplicant generates its own nonce, SNonce and calculates the PTK. In the second message, the supplicant sends its SNonce to the authenticator. The supplicant also sends the security parameters that it used during association. The entire message is protected from modification using a MIC. MIC is performed by computing a message integrity check over the entire frame and sending the MIC along with the frame. The AP sends the GTK and a sequence number together to the STA. This sequence number provides replay protection and is used in the next multicast or

broadcast frame. Again, the entire message is protected using a MIC. In the fourth message, the STA sends a confirmation to the AP indicating that the temporal keys are now in place to be used by the data-confidentiality protocols. At that point, the IEEE 802.1X controlled ports are opened to allow the flow of frames for data traffic. The MIC protection is mandatory for all frames except the first frame.

5.1.3.2 Group Key Handshake

The Group Key Handshake is necessary only to support the multicast or the broadcast traffic. In a two message Group Key handshake, the AP send a new GTK to the STA. The GTK is encrypted using the KEK assigned to that STA and protects the data from being tampered using a MIC. In the second message, the STA acknowledges the new GTK. The GTK is 32 bytes and is made up of 16 bytes of Group Temporal Encryption Key, 8 bytes of Michael MIC Authenticator Tx Key, and 8 bytes of Michael MIC Authenticator Rx Key. The Group Temporal Encryption Key is used to encrypt Multicast data packets. The Michael MIC Authenticator Tx Key is used to compute MIC on Multicast packet transmitted by AP. The Michael MIC Authenticator Rx Key is unused as stations do not send multicast traffic. The Michael MIC Authenticator Tx/Rx Keys provided in the handshake are only used, if the network is using TKIP to encrypt the data. The GTK used in the network may need to be updated due to the expiry of a preset timer or when a device leaves the network.

5.1.4 Secure Data Communication

In this stage, the supplicant and the AS exchange frames through the AP using supported data confidentiality and integrity protocols.

5.1.5 Connection Termination

During this stage, the STA and the AP exchange frames to terminate the existing secure connection.

5.2 Modelling 802.11i using Strand Spaces

In this section, we model 802.11i using the proof based approach. We abstract away the lower-level communication details involved in the 802.11i protocol while modeling it using the proof based approach. In order to prove the authentication property of a protocol, we do not need to focus on the intricate implementation details of the protocol. Instead, we concentrate on the higher level abstraction and emphasize on necessary actions to avoid possible attacks by the illegitimate parties.

5.2.1 Modelling EAP-802.1X Authentication

A sample run of the 802.11i's EAP-802.1X authentication in terms of SSM is depicted in Figure 5.2.

The authentication starts with an EAPOL-Start message sent to the AP by the supplicant. The AP sends an EAP-Request/Identity frame to the supplicant. The supplicant replies to the AP with an EAP-Response/Identity frame. The AP acts as a pass through server and passes this frame to the RADIUS server as a RADIUS-Access-Request packet. The RADIUS server replies with a

RADIUS-Access-Challenge packet to the AP. The AP, acting as a pass through server, passes this frame to the supplicant as an EAP-Request. The EAP-Request is of an appropriate authentication type and contains relevant challenge information. The supplicant answers this challenge by sending an EAP-Response packet to the AP. The AP translates this EAP-Response packet into a Radius-Access-Request packet and forwards it to the RADIUS server. Based on the this Radius-Access-Request, the RADIUS server grants or rejects access with a Radius-Access-Accept or Radius-Access-Reject packet respectively. In some implementations, the RADIUS server can continue the authentication process with further challenges. The AP translates the Radius-Access-Accept or Radius-Access-Reject packets into the EAP-Success or EAP-Failure packets respectively and forwards them to the supplicant. If the response from the RADIUS server was accept then the controlled port is authorized. When the supplicant wants to leave the network, an EAPOL-Logoff message is sent by the supplicant that sets the controlled port to the initial unauthorized state.

In Figure 5.2, we represent the message exchange between the peer, the authenticator, and the RADIUS server. P, A, R, and M represent the peer, the authenticator, the RADIUS server and the message respectively. Note that P, A, R, are the abstractions of the ids of the peer, the authenticator and the RADIUS server respectively. M is the abstraction of the rest of the communication message. The term, $\{MPARc_1\}_{K_{PA}}$, represents a message that is encrypted with a shared secret between P and A, i.e., (K_{PA}). The term c_1 represents a challenge. The challenges (represented as $c_1, c_2, ...$), answers (represented as $x_1, x_2, ...$) and the challenge response (represented as cr) are the abstractions of combinations of different nonces, shared secrets and keys. This abstraction facilitates the formal analysis as it hides the lower level communication and cryptographic details. These lower level communication and cryptographic details are not essential for the authentication analysis,

although they could improve the security in some sense.

5.2.2 Modelling the 4-way Handshake

In Figure 5.3, we illustrate an abstract model of the 4-way handshake. After the pre-association and authentication, the client and the RADIUS server have associated with each other and generate a common secret called the Master Session Key (MSK). The client uses this MSK to derive a Pairwise Master Key (PMK). The Authentication, Authorization, and Accounting (AAA) key material on the RADIUS server is transferred to the access point to derive the same PMK in the access point. The handshake is executed in order to establish a Robust Security Network Association (RSNA). The handshake confirms the existence of the PMK, the liveness of the peers, and the selection of the cipher suite. After a successful handshake, a fresh Pairwise Transient Key (PTK) for each subsequent session is generated.

The A and C in Figure 5.3 corresponds to ids of the access point and the client respectively whereas N_a and N_c are nonces of the access point and the client respectively. We use s to denote sequence number, MIC for Message Integrity Code, GTK for Group Temporal Key, and M is the abstraction of the rest of the communication message. K is the set of shared keys and is represented as K_{ij} where {i,j} $\in C, A$. $K_{pen} \in K$ is the set of keys held by the penetrator. $M_K \in M$ is the message M encrypted with the key K.

5.3 Authentication Proofs

In this section, we carry out algebraic operations to prove the authentication of 802.11i. A protocol needs to ascertain that the messages are intact and that no secret is divulged during its execution.

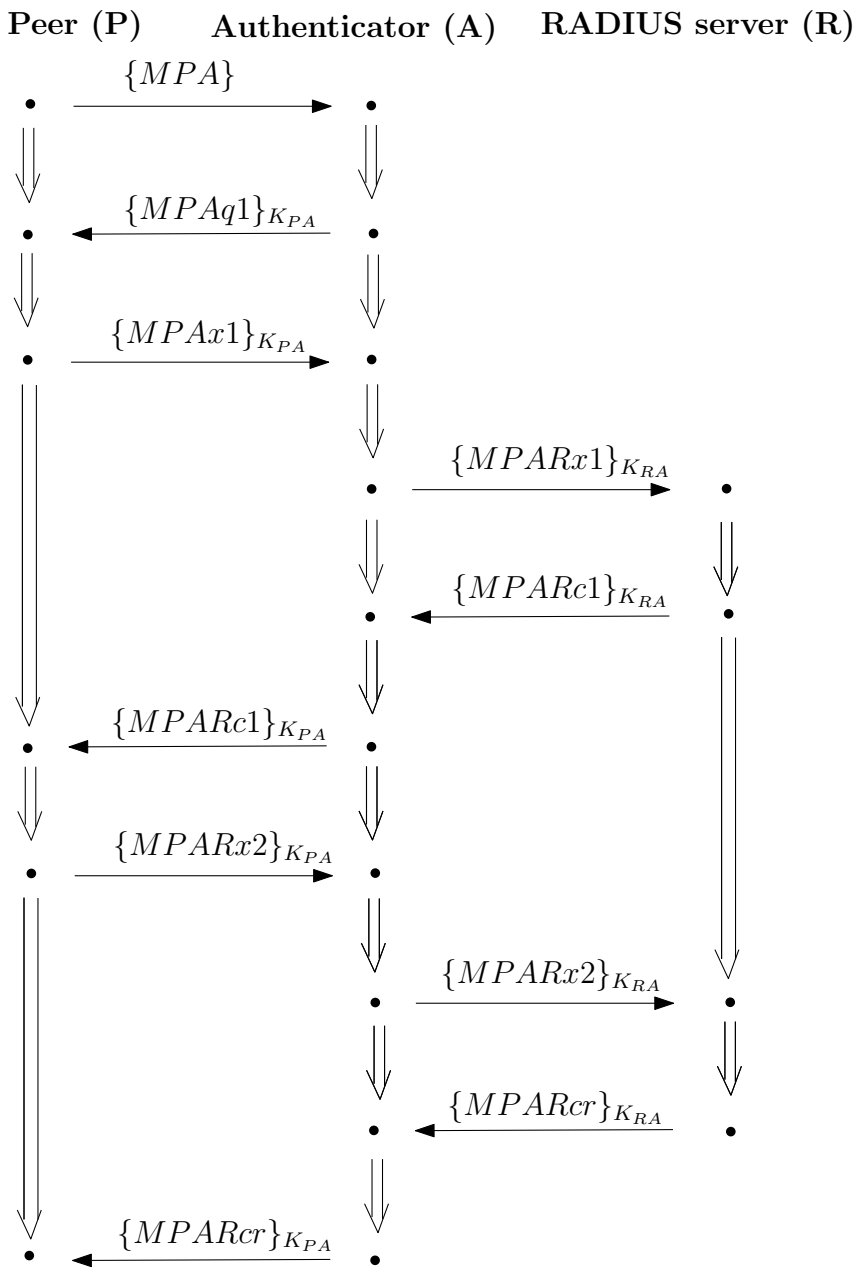

Figure 5.2: A Strand Space Representation of EAP-802.1X in 802.11i.

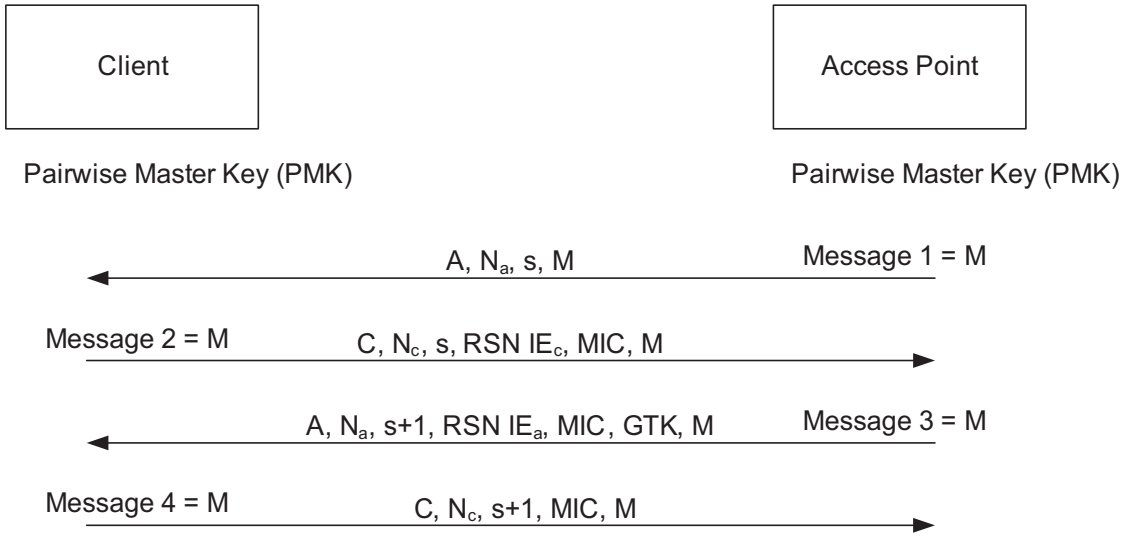

Figure 5.3: An Abstract Representation of the 4-way Handshake of 802.11i

Our focus is to prove the authentication property in which each participant involved in the communication should be certain that the messages are coming from the legitimate participants. In terms of strand space formalism, encrypted messages should originate from the regular strands. We make use of the ideal cryptography and algebra freeness assumptions provided in Section 5.3.1.

5.3.1 Assumptions

We lay out our assumptions in this Section. Our set of assumptions is in accordance with the assumptions set forth by other researchers in the related area.

1. In a protocol environment, each participant is associated with a unique ID. Moreover, for each ID there is a key (or a pair of public-private keys in case of asymmetric cryptosystem) associated with it.

2. In the present work, we do not consider encrypting a message more than once. Earlier work has shown that double or multiple encryptions do not serve a better purpose than a

single encryption. Instead, sometimes it is detrimental to encrypt a message more than once [DY83].

3. In literature of security protocols, a penetrator is generally assumed to be a legitimate participant of the network. The legitimate participant becomes a penetrator when he behaves in an undesired manner.

4. We assume ideal cryptography where all participants share limited computational and cryptanalytic abilities.

5. For analysis purposes, we confine ourselves to a single penetrator. It is shown that increasing the number of penetrators does not increase the probability of a successful attack. A single penetrator can be modeled such that it emulates any number of penetrators [Low96].

6. We assume our algebra to be freely generated as is assumed in [FHG99, GF02].

5.3.2 Defining the Authentication Security Objective

The authentication can be defined using the agreement property presented in [Low97, Low99]. A protocol guarantees a participant B (say, as the responder) agreement for certain data items x if: Each time a participant B completes a run of the protocol as responder using x, apparently with A, then there is a unique run of the protocol with the principal A as initiator using x, apparently with B.

A weaker non-injective agreement does not ensure uniqueness, but requires only: Each time a participant B completes a run of the protocol as responder using x, apparently with A, then there exists a run of the protocol with the principal A as initiator using x, apparently with B. We used

this property for defining a heuristic state space search model for the authentication of security protocols in [FMG04].

5.3.3 Proving EAP-802.1X

We present our theoretical model for EAP-802.1X in this section. Figure 5.2 represents a bundle B_1 in a strand space Σ where the participants have disjoint ids. The nodes in Figure 5.2 are represented by little circles.

5.3.3.1 Theoretical Model

We represent the peer, the authenticator, and the RADIUS server by P, A and R respectively. M represents the set of messages exchanged among the participants. We define $T_{name} \in$ {P, A, R } and $N \in$ {P, R}. K is the set of shared keys and is represented as K_{ij} where {i,j} $\in T_{name}$. $K_{pen} \in K$ is the set of keys held by the penetrator. $M_K \in M$ is the message M encrypted with the key K. The set of questions asked by the authenticator is denoted by the set q = {q1, q2, q3,...}. The set $c = \{c1, c2, c3, ...\}$ represents the challenges raised by the RADIUS server. The challenge response c_r indicates a success or a failure. The set $x = \{x1, x2, x3, ...\}$ is the set of answers given by the peer.

The set of queries is represented by $Q = c \bigcup q$. The single arrow, $\rightarrow$, indicates that a message is sent to another participant. The double arrow, $\Rightarrow$, connects two successive nodes on the same strand. The peer strand is represented as S_{peer} and is equal to $Peer[P, M, R, A, c_i, q_j, x_k, K_{PA}]$. This strand is a symbolic representation of infinite instantiations of the peer strand. From 5.2, the trace of the peer strand can be represented as follows:

$\langle +\{PAM\}, -\{MPAq_1\}_{K_{PA}}, +\{MPAx_1\}_{K_{PA}}, -\{MPARc_1\}_{K_{PA}}, -\{MPARc_1\}_{K_{PA}}, -\{MPARc_1\}_{K_{PA}},$

$-\{MPARc_1\}_{K_{PA}}, +\{MPARx_2\}_{K_{PA}}, -\{MPARc_r\}_{K_{PA}}\rangle$.

Similarly, the authenticator strand S_{auth} is represented as $Auth[P, M, R, A, c_i, q_j, x_k, K_{NA}]$. We represent the trace of the authenticator strand as:

$\langle -\{MPA\}, +\{MPAq_1\}_{K_{PA}}, -\{MPAx_1\}_{K_{PA}}, +\{MPARx_1\}_{K_{RA}}, -\{MPARc_1\}_{K_{RA}}, +\{MPARx_1\}_{K_{RA}},$

$-\{MPARc_1\}_{K_{RA}}, +\{MPARc_1\}_{K_{PA}}, -\{MPARx_2\}_{K_{PA}}, +\{MPARx_2\}_{K_{RA}}, -\{MPARc_r\}_{K_{RA}}\rangle$.

The Radius strand S_{RADIUS} is equal to $Radius[P, M, R, A, x_k, c_j, K_{NA}]$. We represent the trace of the RADIUS strand as:

$\langle -\{MPARx_1\}_{K_{RA}}, +\{MPARc_1\}_{K_{RA}}, -\{MPARx_2\}_{K_{RA}}, +\{MPARc_r\}_{K_{RA}}\rangle$.

After formally representing EAP-802.1X, we begin by proving the authentication property of the protocol. We make use of the agreement property to claim that if the peer completes its protocol run with a unique set of parameters, it can be inferred that the authenticator and the RADIUS server must also have completed their part of protocol run using the same set of parameters. Authentication is guaranteed in both directions, that is, the peer authenticates the authenticator and the authenticator authenticates the peer. The RADIUS server is assumed to be authentic in 802.11i.

5.3.3.2 Peer's Authentication

A bundle represents a *unique* run of a protocol. Unique refers to a specific set of parameters used in only one protocol run in the entire strand space. Consider a strand space Σ in which a bundle B_1 represents a protocol run. SSM defines authentication in terms of bundle-height of strands of legitimate parties. If a strand $S_{peer} \in \Sigma$ represents a peer strand in a bundle B_1, then

the RADIUS and the authenticator strands should also lie in the same bundle B_1 to ensure that the peer, the authenticator and the RADIUS server are involved in the same session of the protocol. The presence of the peer (S_{peer}), the authenticator (S_{auth}) and the RADIUS server (S_{RADIUS}) in the same bundle B_1 guarantees that the peer is talking to the legitimate authenticator and the RADIUS server and not with any masquerading agent.

We represent the peer strand $S_{peer} = Peer[P, M, R, A, c_i, q_j, x_k, K_{NA}]$, the authenticator strand $S_{auth} = Auth[P, M, R, A, c_i, q_j, x_k, K_{NA}]$. Similarly, the strand of the Radius server is given as $S_{RADIUS} = Radius[P, M, R, A, x_k, c_j, K_{NA}]$.

Theorem 1: If S_{peer} has the bundle-height of 6 then for a unique set of questions and answers, there will be authenticator and RADIUS strands of bundle-height 10 and 4 respectively.

Proof 1: Proof of theorem 1 will imply that both the authenticator and the RADIUS server have completed their protocol run with matching variables in the same session with the peer.

The trace of the strand $S_{peer} \in Peer[P, M, R, A, c_i, q_j, x_k, K_{NA}]$, as can be seen in Figure 5.2, is given below:

$\langle +\{MPA\}, -\{MPAq_1\}_{K_{PA}}, +\{MPAx_1\}_{K_{PA}}, -\{MPARc_1\}_{K_{PA}}, +\{MPARx_2\}_{K_{PA}}, -\{MPARc_r\}_{K_{PA}} \rangle$

We need to prove that the term $\{MPARc_r\}_{K_{PA}}$ originates on a regular node in the bundle B_1. This proof will help in determining the bundle-height of the authenticator.

Theorem 1.1: $\{MPARc_r\}_{K_{PA}}$ originates on a regular node in the bundle B_1.

Proof 1.1: Consider the bundle B_1 in Σ. We assume that $K_{NA} \notin K_{pen}$. Intuitively, this assumption implies that the penetrator does not possess the secret keys of any other participant. We need to prove that any term encrypted with K_{NA} must originate only on the regular nodes in a bundle. Since K_{NA} $(= \{K_{PA}, K_{RA}\}) \notin K_{pen}$, we just need to show that any regular node does not generate the key

K_{NA}. The traces of the peer (S_{peer}), the authenticator (S_{auth}), and the RADIUS (S_{RADIUS}) server show that no key is a subterm of any term of these traces.

We need to prove that K_{PA} is generated on a regular node. We will prove this by showing that K_{PA} is not generated on a penetrator node. We consider the case in which the penetrator generates this secret. As we assumed that $K_{PA} \notin$ set of penetrator keys, we discuss the possible set of actions that a penetrator can take in the following lines.

Message: The penetrator strand of this type is $\langle +term \rangle$. The *Message* strand has only one positive node. This means that the penetrator emits a term without previously obtaining it from anywhere. In our case, the term is equal to $\{MPARc_r\}_{K_{PA}}$. Since any penetrator can not originate a term encrypted with the key K_{PA}, this implies that Σ lacks the message strand.

Flushing: The flushing represents a strand $\langle -term \rangle$ which indicates that the penetrator received a term from somewhere and then flushed it. We claim that $\{MPARc_r\}_{K_{PA}}$ is not originated by the penetrator because an originating node is always a positive node. We do not need to worry about any strand with only negative nodes because a negative node does not imply origination.

Tee: Its trace has a strand of the form $\langle -term, +term, +term \rangle$. The tee strand shows that the penetrator received a term and then forwarded that term twice. As the penetrator received a term in the first node, the term could not have been originated by the penetrator. So $\{MPARc_r\}_{K_{PA}}$ was not originated on a Tee strand.

Concatenation: Its strand is of the form $\langle -term1, -term2, +term1term2 \rangle$. The penetrator received two terms, concatenated them to form a new term and then forwarded the concatenated term. Considering our algebra to be free, no new term can be obtained by simply concatenating other terms. Hence, no new term is generated at the positive node of the concatenation strand.

Instead, the penetrator is sending a concatenated term. So $\{MPARc_r\}_{K_{PA}}$ is not originated on a concatenation strand.

Separation: The trace of separation is $\langle -term1term2, +term1, +term2\rangle$. Since penetrator is getting both the terms, term1 and term2, from the previous node, the separation strand lacks any positive originating node.

Key: This strand emits the key $\langle +K\rangle$. The algebra freeness assumption guarantees that a key cannot be equal to any encrypted message. So $\{MPARc_r\}_{K_{PA}}$ cannot be generated by the key penetrator strand.

Encryption: Its strand can be written as $\langle -K, -term, +\{term\}_K\rangle$. The trace of encryption states that the $\{term\}_K$ is equal to $\{MPARc_r\}_{K_{PA}}$. Using algebra freeness, if two encrypted terms are equivalent, the only possibility is that the term is equal to $\{MPARc_r\}$ and K is equal to K_{PA}. It means that the penetrator receives the key K_{PA} in its first node. We assume that any legitimate participant never sends any secret without encryption. Therefore, encryption strand does not produce $\{MPARc_r\}_{K_{PA}}$.

Decryption: The decryption strand has the trace $\langle -K^{-1}, -\{term\}_K, +term\rangle$. As the positive term is obtained by decrypting the previous node, no positive node serves as an originator.

After ruling out all the possible penetrator strands, it is safe to conclude that $\{MPARc_r\}_{K_{PA}}$ originates on a regular node in B. Equivalently, this proof can also be extended for $\{MPARc_r\}_{K_{RA}}$. The proof os this is similar to the proof 1.1. However, we will prove that $\{MPARc_r\}_{K_{RA}}$ originates on the RADIUS strand. We present this proof in the following lines.

Theorem 1.2: $\{MPARc_r\}_{K_{RA}}$ originates on the RADIUS strand.

Proof 1.2: We have proved that $\{MPARc_r\}_{K_{PA}}$ originates on a regular node in proof 1.1. Since

c_r is a subterm of the $\{term\}_{K_{PA}}$, it implies that c_r originates on a regular node. If c_r is a subterm of $\{term\}_{K_{RA}}$, then it is clear from the traces of regular strands that this term originates only on the RADIUS strand. We know that c_r can not originate on the penetrator strand. Hence, $\{MPARc_r\}_{K_{RA}}$ belongs to the RADIUS strand where M is any message, $\{PAR\} \in T_{name}$, and c_r is the challenge response.

The term $\{MPARc_r\}_{K_{RA}}$ is present on the last node on the RADIUS strand. According to the bundle property, a bundle must contain all the previous nodes that collectively makes the bundle-height to be equal to 4.

Since all the messages are encrypted with symmetric keys and we assume that K_{PA} and K_{RA} do not belong to K_{pen}, we say that no penetrator can sit in the middle and behave as a regular participant by simply forwarding the messages to the legitimate parties. The keys K_{PA} and K_{RA} are shared between the peer and the authenticator, and the authenticator and the RADIUS server respectively. Hence, all the messages arriving at any regular node must be originated on the legitimate strands. The peer receives the term $\{MPARc_r\}_{K_{PA}}$ in its last message of the peer strand that makes the peer height equal to 6. Since this message occurs in the last node of the authenticator strand, according to the bundle property it must contain all the previous nodes that makes the bundle-height of the authenticator equals to 10. Similarly, the RADIUS server strand possesses the term $\{MPARc_r\}_{K_{RA}}$ on its 4th node which makes the RADIUS server's height equals to 4. Thus authenticator's height is equal to the peer's height plus the RADIUS server's height, i.e, 4 + 6 = 10. This also implicitly guarantees that the authenticator is not rogue because it has to have the secret keys to forward messages between the peer and the RADIUS Server.

5.3.3.3 Authenticator's Authentication

Consider a strand space Σ with the bundle B_1 representing a protocol run. We proceed by stating that if a bundle contains a strand $S_{auth} \in \Sigma$ then the peer and the RADIUS strands will agree with the authenticator.

Theorem 2: Assume a typical run of the protocol in which the RADIUS server challenges the peer. If $S_{auth} \in Authenticator[P, A, M, R, c_i, q_j, x_k, K_{NA}]$ *of B-height of at least 9, then there are regular strands such that:*

1. $S_{peer} \in Peer[P, A, M, R, c_i, q_j, x_k, K_{PA}]$ *of B-height at least 5.*

2.$S_{radius} \in Radius[P, A, M, R, c_i, x_j, K_{RA}]$ *of B-height = 4.*

Proof 2: The trace S_{auth} can be written as: $\langle -\{MPA\}, +\{MPAq_1\}_{K_{PA}}, -\{MPAx_1\}_{K_{PA}},$
$-\{MPAx_1\}_{K_{PA}}, +\{MPARx_1\}_{K_{RA}}, -\{MPARc_1\}_{K_{RA}}, +\{MPARc_1\}_{K_{PA}}, -\{MPARx_2\}_{K_{PA}}$
$-\{MPARx_2\}_{K_{PA}}, +\{MPARx_2\}_{K_{RA}}, -\{MPARc_r\}_{K_{RA}}\rangle$.

We do not take the trivial case in which the RADIUS server does not challenge the peer. We assumed that $K_{RA} \notin K_{pen}$. By using a proof similar to proof 1.1, we can infer that $\{MPARc_r\}_{K_{RA}}$ originates on a regular node in the bundle. Using proof 1.2, we can say that the term $\{MPARc_r\}_{K_{RA}}$ originates on the RADIUS server strand. Using theorems 1.1 and 1.2, we can conclude that the bundle-height of the RADIUS server strand is equal to 4. Similarly, using theorem 1.1 we can conclude that the term $\{MPAx_2\}_{K_{PA}}$ lies on a regular peer strand. Since $\{MPAx_2\}_{K_{PA}}$ is the last node on the peer strand, bundle property makes the bundle-height of the peer equal to 5.

The term $+\{MPA\}$ is not necessarily an encrypted message. Even if some intruder tries to impersonate as a legitimate user, the reply from the authenticator is encrypted by a shared key between the authenticator and the legitimate user. The intruder will not be able to reply to the

authenticator with the term $+\{MPAx_1\}_{K_{PA}}$. We need to emphasize the fact that the authenticator can not guarantee the bundle-height of its strand to exceed from 9 because any penetrator can sit in the middle and throw away (flush) messages among regular parties. The basic notion is the guarantee that if a message encrypted with a secret key is received by a participant, then that message must have originated on a regular strand. However, we can not guarantee that a message sent by a legitimate party will always be received.

5.3.3.4 Case Analysis

We present a scenario where peer-authenticator communication is not supported by a shared secret. In a case in which the authenticator forwards messages to the peer after communicating with the RADIUS server, we expect a man-in-the-middle attack by a penetrator lying in between the peer and the authenticator. The penetrator possesses a strand of the form Tee with a trace $\langle -term, +term, +term \rangle$. In this strand, the penetrator gets a message from a legitimate party and forwards it without changing it. The penetrator may apply multiple strands to form a complex attack. In the scenario presented above, any penetrator lying in the middle can pose itself as a legitimate participant to the peer, the authenticator, or to both.

The authenticator acts as a gateway between the peer and the RADIUS server. It first decrypts the incoming messages using its shared secret with the source and then forwards the outgoing messages after encrypting it using a shared secret with the destination. In Figure 5.4, a penetrator P_{pen} behaves as an authenticator and engages the peer into a protocol run. If the penetrator starts communicating with the authenticator, a hostile situation may exist. Consider a public key system in which the penetrator starts running the protocol by simply forwarding the messages between

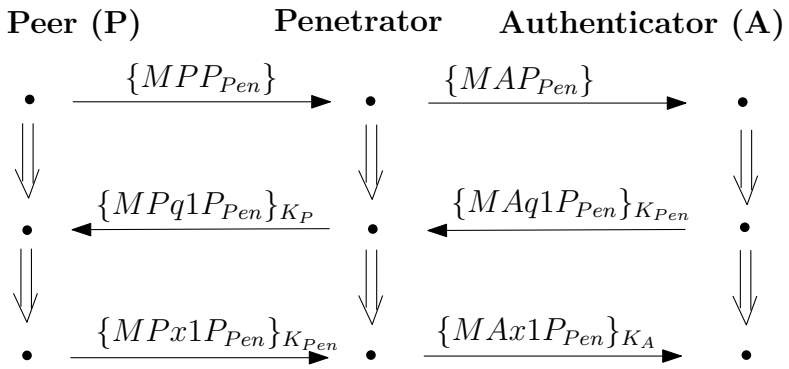

Figure 5.4: Case Analysis of 802.11i.

the peer and the authenticator. In this way, a penetrator may be able to get all the secret questions answered and gets authenticated. The rest of the protocol proceeds as normal. The penetrator behaves as an authenticator for the peer and as a peer for the authenticator as shown in Figure 5.4.

5.3.4 Proving 4-Way Handshake

We present our theoretical model in this Section. Figure 5.3 represents a bundle B_2 in the strand space Σ where the participants have disjoint ids. The nodes in Figure 5.3 are represented by little circles.

5.3.4.1 Theoretical Model

A bundle represents a *unique* run of a protocol. Unique refers to a specific set of parameters used in only one protocol run in the entire strand space. Consider a strand space Σ in which bundle B_2 represents a protocol run. SSM defines authentication in terms of bundle-height of strands of legitimate parties. If a strand $S_{client} \in \Sigma$ represents a client strand in a bundle B_2, then the access point's strand should also lie in the same bundle B_2 to ensure that the client and the access point are involved in the same session of the protocol. The presence of the client (S_{client}) and the access

point (S_{ap}) in the same bundle B_2 guarantees that both access point and client are legitimate.

We represent the client's strand S_{client} to be equal to

$Client[C, A, M, N_a, N_c, s, s+1, MIC, RSNIE_c, RSNIE_a, GTK]$, and the access point's strand S_{ap} to be equal to $AP[C, A, M, N_a, N_c, s, s+1, MIC, RSNIE_c, RSNIE_a, GTK]$.

5.3.4.2 Client's Authentication

We perform the Client's authentication in the following lines.

Theorem 3: If S_{client} has the bundle-height of 3 then for a unique set of nonces, there will be an access point strand of bundle-height 3.

Proof 3: Proof of theorem 3 will imply that the access point has completed its protocol run with matching variables in the same session with the client.

The trace of the strand $S_{client} \in Client[C, A, M, N_a, N_c, s, s+1, MIC, RSNIE_c, RSNIE_a, GTK]$, as can be seen in Figure 5.3, is given below:

$\langle -\{A, N_a, M, s\}_{K_{CA}}, +\{C, N_c, M, s, MIC, RSNIE_c\}_{K_{CA}}, -\{A, N_a, M, s{+}1, MIC, RSNIE_a, GTK\}_{K_{CA}}, +\{C, N_c, M, s+1, MIC\}_{K_{CA}}$.

We need to prove that the term $\{A, N_a, M, s{+}1, MIC, RSNIE_a, GTK\}_{K_{CA}}$ originates on a regular node in the bundle B. This proof will help in determining the bundle-height of the access point.

Theorem 3.1: $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$ originates on a regular node in the bundle B.

Proof 3.1: Consider a bundle B_2 in Σ. We assume that $K_{CA} \notin K_{pen}$. Intuitively, this assumption implies that the penetrator does not possess the secret keys of any other participant. We need to prove that any term encrypted with K_{CA} must originate only on the regular nodes in a bundle. Since

$K_{CA} \notin K_{pen}$, we just need to show that any regular node does not generate the key K_{CA}. The traces of the peer (S_{client}) and the access point (S_{ap}) show that no key is a subterm of any term of these traces.

We need to prove that K_{CA} is generated on a regular node. We will prove this by showing that K_{CA} is not generated on a penetrator node. We consider the case in which the penetrator generates this secret. As we assumed that $K_{CA} \notin$ set of penetrator keys, we discuss the possible set of actions that a penetrator can take in the following lines.

Message: The penetrator strand of this type is $\langle +term \rangle$. The *Message* strand has only one positive node. This means that the penetrator emits a term without previously obtaining it from anywhere. In our case, the term is equal to $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$. Since any penetrator can not originate a term encrypted with the key K_{CA}, this implies that Σ lacks the message strand.

Flushing: The flushing represents a strand $\langle -term \rangle$ which indicates that the penetrator received a term from somewhere and then flushed it. We claim that $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$ is not originated by the penetrator because an originating node is always a positive node. We do not need to worry about any strand with only the negative nodes because a negative node does not imply origination.

Tee: Its trace has a strand of the form $\langle -term, +term, +term \rangle$. The tee strand shows that the penetrator received a term and then forwarded that term twice. As the penetrator received a term in the first node, the term could not have been originated by the penetrator. So $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$ was not originated on a Tee strand.

Concatenation: Its strand is of the form $\langle -term1, -term2, +term1term2 \rangle$. The penetrator re-

ceived two terms, concatenated them to form a new term and then forwarded the concatenated term. Considering our algebra to be free, no new term can be obtained by simply concatenating other terms. Hence, no new term is generated at the positive node of the concatenation strand. Instead, the penetrator is sending a concatenated term. So $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$ is not originated on a concatenation strand.

Separation: The trace of separation is $\langle -term1 term2, +term1, +term2 \rangle$. Since penetrator is getting both the terms, term1 and term2, from the previous node, the separation strand lacks any positive originating node.

Key: This strand emits the key $\langle +K \rangle$. The algebra freeness assumption guarantees that a key cannot be equal to any encrypted message. So $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$ cannot be generated by the key penetrator strand.

Encryption: Its strand can be written as $\langle -K, -term, +\{term\}_K \rangle$. The trace of encryption states that the $\{term\}_K$ is equal to $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$. Using algebra freeness, if two encrypted terms are equivalent, the only possibility is that the term is equal to $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}$ and K is equal to K_{CA}. It means that the penetrator receives the key K_{CA} in its first node. We assume that any legitimate participant never sends any secret without encryption. Therefore, encryption strand does not produce $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$.

Decryption: The decryption strand has the trace $\langle -K^{-1}, -\{term\}_K, +term \rangle$. As the positive term is obtained by decrypting the previous node, no positive node serves as an originator.

After ruling out all the possible penetrator strands, it is safe to conclude that $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$ originates on a regular node in B_2. Since all the messages are encrypted with symmetric keys and we assume that K_{CA} does not belong to K_{pen}, we say that no penetrator

can sit in the middle and behave as a regular participant by simply forwarding the messages to the legitimate parties. The key K_{CA} is shared between the client and the access point. Hence, all messages arriving at any regular node must be originated on the legitimate strands. The client receives the term $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$ in its last message of the client strand that makes the peer height equals to 3. Since this message occurs in the last node of the access point, according to the bundle property it must contain all the previous nodes that makes the bundle-height of the access point equals to 4. Thus access point's height is equal to the client's height, i.e, 3.

5.3.4.3 Authenticator's Authentication

Consider a strand space Σ with the bundle B representing a protocol run. We proceed by stating that if a bundle contains a strand $S_{ap} \in \Sigma$ then the client's strand will agree with the access point's strand.

Theorem 4: If $S_{ap} \in AP[C, A, M, N_a, N_c, s, s+1, MIC, RSNIE_c, RSNIE_a, GTK]$ *of B-height of at least 4, then there is a regular strand of the client such that:*

$S_{client} = Client[C, A, M, N_a, N_c, s, s+1, MIC, RSNIE_c, RSNIE_a, GTK]$ *of B-height equals to 4.*

Proof 4: The trace S_{ap} can be written as: $\langle +\{A, N_a, M, s\}_{K_{CA}}, -\{C, N_c, M, s, MIC, RSNIE_c\}_{K_{CA}}, +\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}, -\{C, N_c, M, s+1, MIC\}_{K_{CA}}$.

We assumed that $K_{CA} \notin K_{pen}$. By using a proof similar to proof 3.1, we can infer that the term equal to $\{C, N_c, M, s+1, MIC\}_{K_{CA}}$, originates on a regular node in the bundle. And hence similar to proof 3, we can conclude that the bundle-height of the access point's strand is equal to 4. Since $\{C, N_c, M, s+1, MIC\}_{K_{CA}}$ is the last node on the client's strand, bundle property makes the

bundle-height of the client to be equal to 4.

5.4 Summary

In this chapter, we have performed the authentication analysis of 802.11i protocol using a proof based approach. We divide 802.11i in five different stages: discovery, open system and EAP-802.1X authentication, key generation and distribution, secure data communication, and connection termination. The only two stages that guarantee authentication in 802.11i are EAP-802.1X authentication and 4-way handshake. We model these stages in the proof based approach. After modelling these stages, we develop a theoretical model and build proofs of authentication for the 802.11i protocol. We also present a special case and analyze it to show how modifications in our model can compromise authentication by causing a man-in-the-middle attack.

CHAPTER 6
A MULTI-AGENT APPROACH FOR SECURITY PROTOCOL VERIFICATION

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors [RN95]. A Multi-agent System (MAS) can be characterized as a system composed of several agents that aim to solve problems collectively. These problems are difficult to achieve by an individual agent. The distinguishing characteristics of MASs are:

1. each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint,
2. there is no system global control,
3. data are decentralized, and
4. computation is asynchronous.

A MAS can be viewed as a distributed parallel computer system where each agent (participant) communicates with other agents. A primary aim of MAS systems is to achieve flexibility and fault tolerance. A MAS system should be modifiable without much rewriting. In addition, these systems are designed to facilitate rapid self-recovery.

We present a new approach that includes the notion of multi-agent in the formal verification of security protocols [GFM07a, GFM07b]. In particular, our focus is to verify the authentication property in a multi-agent environment. The intuition is to capture the behavior of a distributed system as a multi-agent system. An agent is an abstraction that is used in the distributed systems to

represent an entity such as a process or a principal. It is important to give a clear notion of multi-agent systems in the proposed frameworks for security protocol verification in order to represent the formal model of knowledge. For example, the limitation of SSM is that it assumes that all the information available to a principal is either supplied initially or is contained in messages received by that principal [HP03]. However, other important information may also be available to a principal in a security setting, such as a principal may combine information from different roles played by him in the protocol. The presented model captures the formal model of knowledge and belief of agents over time. We also present a formal proof of authentication of the 4-way handshake in the 802.11i protocol.

6.1 The Multi-agent Theoretical Model

In this section, we describe the participants involved in a security protocol and their behavior. The participants or agents in our protocol consists of honest agents and the penetrator. Our model is an asynchronous composition of a set of these honest agents and the penetrator. The strength of our system is that we assume the presence of an insecure communication channel. We model this insecure communication channel in a way where an honest agent has no guarantees about the origin of a message and every communication goes through the penetrator. We define the penetrator and its abilities in the following section.

6.1.1 Penetrator

In our model, the penetrator (or intruder or adversary) is a user of the system that tries to impersonate himself as some other honest agent of the system. The penetrator is assumed to have the

ability to eavesdrop on every communication and generate fake messages. Intuitively, every sent message is intercepted by the penetrator and all messages received by honest agents were actually sent by the penetrator. We also allow the penetrator to create and destroy messages. The complete set of abilities of a penetrator is defined as follows.

Generation A penetrator is generally assumed to generate messages from the set of messages he possesses. He may generate new messages using some operations performed on this set.

Forwarding Forwarding is defined as sending a message after receiving it from some participant. A penetrator is equipped with the ability to forward any message in the network to any participant he likes.

Copying Message copying can be done by simply duplicating the data item even though the ingredients of the data items are unknown to a participant. The notion of a penetrator possess this property that he can copy any message to create it duplicates as many times as he wants in order to attack a protocol.

Deleting A penetrator can obtain any message from the network and then can flush it out from the network. This is similar to deleting of messages from a public network channel.

Concatenation A penetrator is assumed to be able to perform concatenation on two messages he possesses.

Encryption A penetrator is assumed to able to encrypt any data item it possess with any key he possesses. This can result in infinite number of messages that a penetrator can generate by simply encrypting a message over and over using a key. However, in a realistic scenario,

only a bounded number of encryption suffices the analysis.

Decryption Similar to encryption, a penetrator is capable of decrypting a message using a key he possesses. The key may have been obtained by any successful attack on a protocol, or he may have performed cryptanalysis on any old session key to obtain the key.

However, we make use of the general set of assumptions set forth by the researchers in our area and put a bound on the capability of every participant involved in a protocol. This set of assumption is described in Section 5.3.1.

6.1.2 Messages

Before building a model for security protocol analysis, it is important to describe a complete description of the behavior of agents involved in a typical security protocol. In the following lines, we provide the description of messages in our model.

The messages involved in a protocol are constructed from smaller submessages using concatenation and encryption. The smallest submessage are the ones that do not contain any further submessage. These smallest submessages are called *atomic messages*. An atomic message can belong to any of the following four types.

1. The *Agent name* is used to refer to a participant in a protocol.

2. A *Nonce* is a randomly generated number. A nonce is generated only once during the course of a protocol. The purpose is to ensure that no one can predict the value of a nonce. A message containing a newly generated nonce is assumed to be fresh.

3. A *Key* is used to encrypt messages. Every key k has an inverse k^{-1}. However, for symmetric cryptography, the decryption key is the same as the encryption key, so $k = k^{-1}$.

4. *Data message* have no role in protocol functionality. It is simply intended to be communicated between the agents involved in a protocol.

Let M be the set of all messages such that it contains all possible atomic messages. The set M is constructed as follows:

Concatenation Two messages $m_1 \in M$ and $m_2 \in M$ can be paired together to form a new message $m_1.m_2 \in M$;

Encryption A message $m \in M$ can be encrypted with a key k to form an encrypted message $m_k \in M$.

Projection A message $m_1.m_2 \in M$ can be projected to form $m_1 \in M$ and $m_2 \in M$;

Decryption A message $m_k \in M$ can be decrypted using a key $k^{-1} \in K \in M$(in antisymmetric cryptography system) or $k \in K \in M$ (in symmetric cryptography system) to form $m \in M$ and $k \in K \in M$. The set $K \in M$ represents the set of all possible keys.

6.2 Syntax

We use *modal logic* to express notions of knowledge. Let A be the set of agents named 1,...,n. The set of *model operators* for A is $a_1, a_2, ..., a_n$. The set of primitive propositions Φ is a nonempty set describing basic facts about the world. To express a statement such as "Client 2 *knows* that the

key Z is fresh," we write $a_2\phi$. Here, $a_2 \in A$ represents client (or agent) 2 and $\phi \in \Phi$ is a primitive proposition stating that the key Z is fresh.

The argument to the primitive propositions are terms. The formal description of terms is as follows.

- If A is any agent's ID, then A is an agent term.
- If a is an agent's variable, then a is an agent term.
- If M is a message, then M is a message term.
- If m is a message variable, then m is a message term.
- If K is a key, then K is a key term as well as a message term. The term k signifies a key term but is also a message term because $k \in K \in M$
- If k is a key variable, then k is a key term as well as a message term.
- If $m1$ and $m2$ are message terms, then $m1.m2$ is a message term.
- If m and k are message terms, then m_k is a message term.

We define the set of formulas by closing off under negation, conjunction, and the modal operators. The *well-formed formulas* are built up from primitive propositions with modal and first-order logic as follows.

- if ϕ is an atomic proposition, then ϕ is a WFF.
- if ϕ is a WFF, then $\neg\phi$ is a WFF.

- if ϕ_1 and ϕ_2 are WFFs then $\phi_1 \wedge \phi_2$ is a WFF.

- if ϕ is a WFF and a is an agent term then $\exists a.\phi$ is a WFF. Here, $\exists a.\phi$ means that there exists some agent a_0 such that ϕ is true when you substitute a_0 for a in ϕ.

We also utilize standard propositional logic abbreviations to form more complicated formulas, such as:

- $\phi \vee \psi$ for $\neg(\neg\phi \wedge \neg\psi)$,

- $\phi \rightarrow \psi$ for $\neg\phi \vee \psi$,

- $\phi \leftrightarrow \psi$ for $(\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$,

- and $\forall a.\phi$ for $\neg\exists a.\neg\phi$.

A statement such as "Client 1 *knows* that the key Z is fresh and Client 2 *knows* that the key Z is not fresh," can be represented in our language as $a_1\phi \wedge a_2\neg\phi$. Here, $a_1$and a_2 are $\in A$ and represent client 1 and client 2 respectively. Also, $\phi \in \Phi$ is a primitive proposition stating that the key Z is fresh and $\neg\phi$ implies that the key Z is not fresh.

6.3 Semantics

In a multi-agent system, every agent is characterized by a local state at a given point in time.

- The local state of an agent contains all the information necessary to model the behavior of that agent at a given point in time. In our Asynchronous Message Passing (a.m.p.) system, the local state of an agent contains the values of variables, history of messages received and sent, and a list of internal actions performed.

- The global state of the system at any time t consists of all the local states of its agents at time t.

- A *run* of the system is defined as a function from time to the global states. A run is a complete description of how the system evolves over time.

- A *point* is a pair (r, t) comprising of a run r and time t.

- The global state $r(t)$ describes the state of the system at a point (r, t). $r(t) = (s_1, s_2, ..., s_n)$, where s_i = agent i's local state at point (r, t).

- A *round* takes place between two time points. For example, round t in run r takes place between time t and $t - 1$.

- The *history* of an agent $a \in A$, over M, at point (r, t), consists of $a's$ initial state followed by the sequence describing $a's$ actions up to time t. If a performs no actions in round t then its history at (r, t) is the same as its history at $(r, t - 1)$. In a.m.p. system, the agent a's history at any point (r, t) is its local state at point (r, t) and can be represented as $r_a(t)$. The history $r_a(t)$ describes everything that has happened in the run r up to time t from agent $a's$ point of view.

- Let E be the set of events. An event $e \in E$ occurs in $a's$ history in round t of run r if e is in $r_a(t)$ but not in $r_a(t - 1)$. In a security protocol, the primitive events are of the form $send(m, a, b)$(i.e., agent a sends m to agent b) and $receive(m, a, b)$(i.e., m is received by agent a from b). An agent can also perform *internal events* (or internal actions) specific to a security protocol, for example, fresh(m) (which checks wether the term m is fresh or not) etc.

The local states of the agents changes as they perform events (or actions) during the course of a protocol. The action performed by the participants as well as the change in their states are reflected in their histories. Actions also cause the global state of the system to change. A message $m \in M$ is *sent* by an agent $a_1 \in A$ in global state $r(t)$ and the global state changes to $r(t+1)$ if and only if:

- There is a session history $H_k(t)$ with $r_k(t)$ contains an action $send(m)$. Intuitively, this means that is a session history that indicates that a message m was sent.

Similarly, a message $m \in M$ is *received* by an agent $a_1 \in A$ in global state $r(t)$ and the global state changes to $r(t+1)$ if and only if:

- There is a session history $H_k(t)$ with $r_k(t+1)$ contains an action $receive(m)$. This means that is a session history that indicates that a message m was received.

We need to make sure that the global states of our system are consistent over time. This can be achieved if histories do not shrink over time and every message $m \in M$ received in round t corresponds to a message that was sent in some earlier round. Following three constraints are defined to ensure that the global states of the system are consistent.

C1. $r_a(t)$ is a history over M;

C2. for every event $receive(m)$ in $r_a(t)$ there exists a corresponding event $sent(m)$ in $r_b(t)$, for some $b \in A$;

C3. $r_a(0)$ is the empty sequence and $r_a(t+1)$ is either identical to $r_a(t)$ or the result of appending one event to $r_a(t)$.

C1 simply states that an agent's local state is its history. The constraint C2 says that every message received at round t corresponds to one that was sent earlier. C3 ensures that histories do not shrink

over time.

6.4 Theoretical Model for the 4-Way Handshake of 802.11i

After laying out the formal multi-agent framework, the next step is to translate the participants and communication of the 4-way handshake into the multi-agent framework. We utilize the *bundle* structure of Strand Space Model (SSM) [FHG99] to capture the evaluation of agents in time. Next, we use the Strand System notion of [HP03] to capture the evolution of bundles in time. Each time an event takes place for any agent $a \in A$, we write $B_1 \mapsto B_2$. With every such event, the mapping $B_1 \mapsto B_2$ grows and forms a chain $B_1 \mapsto B_2 \mapsto B_3$....

For a chain $C = B_1 \mapsto B_2$... and an agent $a \in A$, the history $hist_a^n(C)$, can be defined inductively. The initial history $hist_a^0(C)$ for agent $a \in A$ is $\langle\ \rangle$. The $hist_a^1(C)$ is $\langle\ \rangle$ if no event happens for a, otherwise, it can be obtained by appending $hist_a^0(C)$ to $e_{a,B_0 \mapsto B_1}$, where e_a is the event in $B_0 \mapsto B_1$ for the agent $a \in A$. Similarly, $hist_a^n(C)$ is $hist_a^{n-1}(C)$ if no event happened in $B_{n-1} \mapsto B_n$ for a and $hist_a^{n-1}(C).e_{a,B_{n-1} \mapsto B_n}$ if an event happens for agent a in $B_{n-1} \mapsto B_n$.

From the communication in Figure 5.3, the evolution of system is captured as follows. We denote access point with ap, client with cl, and environment with en. Environment en is assumed to have all the control over communication channel. Whenever an agent sends or receives a message, that message is sent or received through the environment. Also, for the sake of simplicity, we assume the time to be long enough so that a message is send by an agent and received by the environment (and vice versa) in one frame.

The initial histories of all agents is $hist_A^0(C) = \langle\rangle$.

The events for ap causes the bundles to grow like $e_{ap,B_0 \mapsto B_1}$, $e_{ap,B_1 \mapsto B_2}$, $e_{ap,B_2 \mapsto B_3}$,

$e_{ap,B_3 \mapsto B_4}$, $e_{ap,B_4 \mapsto B_5}$,$e_{ap,B_5 \mapsto B_6}$, $e_{ap,B_6 \mapsto B_7}$, $e_{ap,B_7 \mapsto B_8}$, ...

The corresponding histories at each point for ap are $hist^1_{ap}(C) = hist^0_{ap}(C).send(msg1)$,

$hist^2_{ap}(C) = hist^1_{ap}(C).no-op()$,..., $hist^8_{ap}(C) = hist^7_{ap}(C).receive(msg4)$.

Similarly, the client's histories can be given as $hist^1_{cl}(C) = hist^0_{cl}(C).no-op()$,

$hist^2_{cl}(C) = hist^1_{cl}(C).receive(msg1)$,..., $hist^7_{cl}(C) = hist^6_{cl}(C).send(msg4)$,

$hist^8_{cl}(C) = hist^7_{cl}(C).no-op()$.

In a similar fashion, environment's histories are given as:

$hist^1_{en}(C) = hist^0_{en}(C).receive(msg1)$, $hist^2_{en}(C) = hist^1_{en}(C).send(msg1)$,...,

$hist^7_{en}(C) = hist^6_{en}(C).receive(msg4)$, $hist^8_{en}(C) = hist^7_{en}(C).send(msg4)$.

6.5 Formal Proofs

We make use of the ideal cryptography and algebra freeness assumptions provided in Section 5.3.1.

6.5.1 Penetrator

In our model, the penetrator (or intruder or adversary or saboteur) is a user of the system that tries to impersonate himself as some other honest agent of the system. The penetrator is assumed to have the ability to eavesdrop on every communication and generate fake messages. Intuitively, every sent message is intercepted by the penetrator and all messages received by honest agents were actually sent by the penetrator. We also allow the penetrator to create and destroy messages. However, we make use of the general set of assumptions set forth by the researchers in our area and put a bound on the capability of every participant involved in a protocol. This set of assumption is described in Section 5.3.1.

6.5.2 Authentication of the Client and the Access Point

If a message was received by the client at some point in its history, then that message should be present in the access point's history at an earlier point.

Theorem 5: If a message "m" is received in the run r^n_{cl} by the client "cl", then that message "m" must have been sent by the access point "ap" at some earlier point than "n".

Proof 5: Proof of theorem 1 will imply that the client and the access point have completed their protocol run with matching variables in the same session with each other. The r^n_{cl} is basically the history $hist^8_{cl}(C)$ and is described in detail in Section 6.4. We need to prove that the message 2 (from Figure 5.3) originates in the history of the client.

Theorem 5.1: $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$ *originates in the access point's history.*

Proof 5.1: We assumed that the penetrator does not possess the secret keys of any other participant. We need to prove that any term encrypted with the shared PMK (i.e, K_{CA}) must originate only in the histories of the legitimate users of that shared PMK. In other words, we need to show that any regular (non-penetrator) node does not generate the PMK as an open message. We discuss the possible set of actions that a penetrator can take in the following lines.

$Generate - Message$: The penetrator can create a message of the type, $\langle +term \rangle$. This means that the penetrator emits a term without previously obtaining it from anywhere. In our case, the term is equal to $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$. Since any penetrator can not originate a term encrypted with the key K_{PMK}, this implies that $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$ was not originated in the penetrator's history.

$Flushing$: The implies flushing of a message and can be represented as $\langle -term \rangle$. Intuitively, this indicates that the penetrator received a term from somewhere and then flushed it. We claim that

$\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$ is not originated by the penetrator because an originating node is always a node where a message was sent. We do not need to worry about any history indicating the reception of a message as it does nt imply origination.

Tee: It can be represented as $\langle -term, +term, +term \rangle$. The tee shows that the penetrator received a term and then forwarded that term twice. As the penetrator received a term in the first node, the term could not have originated by the penetrator. So $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$ can not be originated by an action similar to Tee.

Concatenation: Concatenation is of the form $\langle -term1, -term2, +term1term2 \rangle$. The penetrator received two terms, concatenated them to form a new term and then forwarded the concatenated term. Considering our algebra to be free, no new term can be obtained by simply concatenating other terms. Hence, no new term is generated at the positive node of the concatenation strand. Instead, the penetrator is sending a concatenated term. So $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$ can not be formed by concatenation.

Separation: The trace of separation is $\langle -term1term2, +term1, +term2 \rangle$. Since penetrator is getting both the terms, term1 and term2, from the previous node, the separation ability of the penetrator lacks any positive originating node.

Key: Key emits the key $\langle +K \rangle$. The algebra freeness assumption guarantees that a key cannot be equal to any encrypted message. So $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$ cannot be generated by the penetrator.

Encryption: It can be written as $\langle -K, -term, +\{term\}_K \rangle$. The trace of encryption states that the $\{term\}_K$ is equal to $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$. Using algebra freeness, if two encrypted terms are equivalent, the only possibility is that the term is equal to $\{A, N_a, M, s+$

$1, MIC, RSNIE_a, GTK\}$ and K is equal to K_{CA}. It means that the penetrator receives the key K_{PMK}. We assume that any legitimate participant never sends any secret without encryption. Therefore, the penetrator does not produce the $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$.

Decryption: The decryption can be represented as $\langle -K^{-1}, -\{term\}_K, +term \rangle$. As the positive term is obtained by decrypting the previous node, no positive node serves as an originator.

After ruling out all the possible penetrator possibilities, it is safe to conclude that the term $\{A, N_a, M, s+1, MIC, RSNIE_a, GTK\}_{K_{CA}}$ originates on a regular node. Since all the messages are encrypted with symmetric keys and we assume that K_{CA} does not belong to the set of keys held by the penetrator, we say that no penetrator can sit in the middle and behave as a regular participant by simply forwarding the messages to the legitimate parties. Hence, the authentication messages arriving at any regular node must be originated on some non-penetrator nodes.

Similar to the client's authentication, the authentication of access point can be stated as:

Theorem 6: If a message "m" is received in the run r^n_{ap} by the access point "ap", then that message "m" must have been sent by the client "cl" at some earlier point than "n".

Proof 6: We start with proving a statement similar to the one we proved earlier.

Theorem 6.1: $\{C, N_c, M, s+1, MIC\}_{K_{CA}}$ originates in the client's history.

Proof6.1: We assumed that $K_{CA} \notin K_{pen}$, meaning that the penetrator does not has the possession of the secret key shared between two non-penetrator participants. By using a proof similar to proof 5.1, we can infer that $\{C, N_c, M, s+1, MIC\}_{K_{CA}}$ originates on a regular node in the bundle and can say that the term $\{C, N_c, M, s+1, MIC\}_{K_{CA}}$ originates in the access point's history. The basic notion

is the guarantee that if a message encrypted with a secret key is received by a participant, then that message must have originated on a regular node. However, we can not guarantee that a message sent by a legitimate party will always be received.

6.6 Summary

For the formal verification of distributed security protocols, it is important to capture the formal model of knowledge. We present a multi-agent architecture to model the knowledge and behavior of agents involved in a distributed system. In particular, our focus is to verify the authentication property of distributed security protocols. The presented architecture contains an explicit notion of multi-agents, which is lacking in some existing verification frameworks. We extend our contributions to present a verification technique. The presented verification technique is simple and defines an explicit involvement of every agent in the multi-agent environment. The proposed architecture and verification strategy are generic in nature. We also apply the proposed technique to analyze the 4-way handshake of the 802.11i protocol.

CHAPTER 7
FUTURE WORK

Our future work is divided into two main steps as explained in the following sections.

7.1 Additional Security Properties

We will attempt to include security properties such as secrecy, in future. Secrecy is the practice of hiding information from others. Modern cryptosystems have this property as a main design goal. In our future network, the system will provide proofs of verification for strong data confidentiality, integrity, and replay protection for every transmitted message. Data confidentiality and integrity help build a secure channel for the user to communicate in an insecure environment. They aim to provide an environment where only the communicating users are able to understand the received messages, and generate or modify valid messages. We aim to construct a trusted environment where replayed messages should be recognized and discarded even though they may pass the integrity check. These requirements could be satisfied by well-designed cryptographic functions and appropriate replay protection techniques.

7.1.1 A Formal Threat model

We plan to extend the formal threat model to security protocols other than 802.11i. Our threat model will be based on the properties described in Chapter 3. The threat model will include:

1. Eavesdropping: Eavesdropping occurs when an adversary can easily sniff and store all the traffic in a WLAN due to the characteristics of the wireless communication. An adversary

may learn partial or complete information from certain messages even when messages are encrypted. It is important to consider that this possibility exists if common message fields are predictable or redundant.

2. Message Insertion: This is possible because an adversary is capable of inserting a message into the wireless network with some moderate equipment. In our analysis, we have assumed that an adversary can generate any chosen packet, modify contents of a packet, and completely control the transmission of the packet.

3. Message Deletion and Interception: Similar to message insertion, we have assumed message deletion and message interception. In message deletion, an adversary is capable of removing a packet from the network before the packet reaches its destination. Message interception means that an adversary is able to control a connection completely.

4. Masquerading: An adversary can masquerade and use the malicious access point. An adversary is also able to install his own AP.

5. Session Hijacking: An adversary might hijack a session.

6. Man-in-the-Middle: An adversary can launch a man-in-the middle attack as discussed in 5.3.3.4. A more rigorous analysis is planned for future where these attacks will be evaluated on the final system.

7.2 Verification of a Wireless Mesh Network

A mesh network is a communications network in which there are at least two pathways to each node. If every node has a direct connection to every other node then the mesh network becomes a fully meshed network. Most mesh networks are partially meshed because a fully meshed network is very costly. In mesh networks, component parts can all connect to each other via multiple hops. Each node needs only to transmit as far as the next node and thus act as repeaters to transmit data from nearby nodes to peers that are too far away to reach, resulting in a network that can span large distances. The resulting infrastructure is reliable, decentralized, and fault tolerant.

Future work will deal with the application of our strategy to other distributed protocols and environments such as a Wireless Mesh Network (WMN). It is important to guarantee the security properties of the WMN to ensure a desired secure system. We state that a WMN can be understood as a Multi-Agent System (MAS) where each node represents an agent of the MAS. The purpose is to benefit from the existing state-of-the-art verification techniques applicable in the multi-agent domain. Our contributions will include:

1. translating the mesh network in a multi-agent framework,

2. representing the security properties as the MAS's formal specifications,

3. applying the Strand System verification strategy to prove the correctness of the MAS's security properties.

7.2.1 Virtual Private Network

We have restricted ourselves to the link layer level authentication. However, it is valuable to note that there are many other security mechanisms in industry, which extend their corporate Virtual Private Network (VPN) based on IPsec, SSL, or SSH to protect the wireless link. These solutions are also successful. They provide the end-to- end security in different layers, and can be implemented with link layer authentication together for better security. We plan to incorporate the verification of these systems in future as well.

CHAPTER 8
CONCLUSION

Used with a communication protocol, a security protocol provides secure delivery of data between two or more parties using cryptographic operations such as encrypt, decrypt, and so forth. Employing informal methods for the verification of security protocols has yielded poor results. In order to foil the vulnerabilities of a security protocol, one would need to employ more sophisticated modes of analyses. Formal methods are useful for the analysis of security protocols because they allow one to do both a thorough analysis of the different paths which an intruder can take, and to specify precisely the environmental assumptions that have been made. The application of formal methods to cryptographic protocols refers to mathematics or logic based techniques for the specification, development, and verification of these protocols.

We introduce an explicit notion of security objectives that needs to be defined before starting the formal verification process. The definition of these objectives is important because these are the methods that enforce security policies and mitigate security risks. We address the security of WLANs in terms of its security objectives. We present a formal model that provides basis for the formal verification of security protocols. The presented mathematical model is then used for the correctness proofs of a WLAN using the 802.11i protocol.

We also develop a modal logic and a proof based approach for the security protocol verification using the Strand Space framework. Our presented models use simple logical formulas to represent the security primitives. The presented approaches allow us to establish the formal proofs about a participant by looking at its own run of the protocol. In our proof based model, we present a generic set of proofs to establish the correctness of a security protocol. In this work, we restrict

ourselves to the verification of the authentication property of the security protocols.

After defining our formal models, we use our proof based system to perform the formal verification of the 802.11i protocol. We start by modeling the 802.11i protocol into our proof based system and then perform the formal verification of the authentication property. We then present the high level abstraction of a protocol run. We consider a penetrator, empower its capabilities, and evaluate the authentication of the proposed protocol. Given the constraints and suggestions of the proposed architecture, we state convincingly that any attempt to defy the authentication mechanism of the 802.11i protocol will not be successful. We further describe a situation where modifications to our model will lead to a successful man-in-the-middle attack.

We also present a multi-agent approach that includes an explicit notion of multi-agent. The Strand Space framework does not contain an explicit notion of multi-agents and assumes that all the information available to a principal is either supplied initially or is contained in messages received by that principal. However, an agent may combine information from different roles played by him in a protocol to launch a powerful attack. The presented approach addresses this issue by modeling the behavior of a distributed system as a multi-agent system. Our multi-agent model captures the combined information, the formal model of knowledge, and the belief of agents over time. We also apply the proposed technique to analyze the 4-way handshake of the 802.11i protocol.

LIST OF REFERENCES

[Aba99] M. Abadi. "Secrecy by typing in security protocols." *Journal of the ACM*, **46**(5):749–786, September 1999.

[ABV04] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. E. Levkowetz. "Extensible Authentication Protocol EAP. RFC 3748.", June 2004.

[AG99] M. Abadi and A.D. Gordon. "A calculus for cryptographic protocols: The spi calculus." *Information and Computation*, **148**(1):1–70, October 1999.

[Arb01] W. A. Arbaugh. "An inductive chosen plaintext attack against WEP/WEP2.", May 2001. Presentations to IEEE 802.11 TGi.

[AT91] Martín Abadi and M. Tuttle. "A Semantics for a Logic of Authentication." In *Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing*, pp. 201–216, 1991.

[BAN90a] Michael Burrows, Martín Abadi, and Roger Needham. "A Logic of Authentication." *ACM Transactions on Computer Systems*, **8**(1):18–36, February 1990.

[BAN90b] Michael Burrows, Martín Abadi, and Roger Needham. "Rejoinder to Nessett." *Operating Systems Review*, **24**(2):39–40, April 1990.

[BGW01] N. Borisov, I. Goldberg, and D. Wagner. "Intercepting mobile communications: the insecurity of 802.11." In *7th Annual International Conference on Mobile Computing and Networking,*, 2001.

[Bie90] P. Bieber. "A logic of communication in a hostile environment." In *Proceedings of the IEEE Computer Security Foundations Workshop III*, pp. 14–22, June 1990.

[BS03] J. Bellardo and S. Savage. "802.11 Denial-of-Service attacks: real vulnerabilities and practical solutions." In *USENIX Security Symposium*, pp. 15–28, August 2003.

[BV98] L. Blunk and J. Vollbrecht. "PPP Extensible Authentication Protocol (EAP). RFC 2284.", March 1998.

[CDV03] D. Chen, J. Deng, and P. Varshney. "Protecting Wireless Networks against a Denial of Service Attack Based on Virtual Jamming." In *Poster Session of the ACM MobiCom03,*, September 2003.

[Che05] H. Cheung. "FBI Teaches Lesson in how to break into Wi-Fi networks.", 2005.

[CHW03] N. Cam-Winget, R. Housley, D. Wagner, and J. Walker. "Security flaws in 802.11 data link protocols." *Communications of the ACM: Special Issue on Wireless networking security*, **46**(5), May 2003.

[CJM00a] E. Clarke, S. Jha, and W. Marrero. "Partial Order Reductions for Security Protocol Verification." In *Proceedings of the 6th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pp. 503–518. Springer-Verlag, 2000.

[CJM00b] E. Clarke, S. Jha, and W. Marrero. "Verifying security protocols with Brutus." *ACM Transactions on Software Engineering and Methodology (TOSEM)*, **9**(4):443–487, October 2000.

[CP97] T. Coffey and P.Saidha. "Logic for verifying public-key cryptographic protocols." *IEE Proc. Comput. Digit. Tech.*, **144**(1):28–32, January 1997.

[CSP92] E. Campbell, R. Safavi-Naini, and P. Pleasants. "Partial belief and probabilistic reasoning in the analysis of secure protocols." In *Proceedings of the IEEE Computer Security Foundations Workshop V*, pp. 84–91, 1992.

[DEk82] D. Dolev, S. Even, and R. karp. "On the security of Ping-Pong protocols." *Information and Control*, **55**(1-3):57–68, 1982.

[DY83] D. Dolev and A. C. Yao. "On the Security of Public Key Protocols." *IEEE Transactions on Information Theory*, **29**:198–208, March 1983.

[FC02] D. B. Faria and D. R. Cheriton. "DoS and authentication in wireless public access networks." In *First ACM Workshop on Wireless Security*, September 2002.

[FD01] B. Fleck and J. Dimov. "Wireless access points and ARP poisoning: wireless vulnerabilities that expose the wired network.", 2001. White paper by Cigital Inc.

[FHG98] F. J. T. Fabrega, J. C. Herzog, and J. D. Guttman. "Strand Spaces: Why is a Security Protocol Correct." In *Proceedings of the IEEE Symposium on Security and Privacy*, May 1998.

[FHG99] F. Thayer Fabrega, J. Herzog, and J. Guttman. "Strand Spaces: Proving security protocols correct." *Journal of Computer Security*, **7**(1):191–230, 1999.

[FM00] S. Fluhrer and D. McGrew. "Statistical analysis of the alleged RC4 keystream generator." In *Proceedings of the 7th International Workshop on Fast Software Encryption*, 2000.

[FMG04] Z. Furqan, S. Muhammad, and R. K. Guha. "A Heuristic State Space Search Model for Security Protocol Verification." In *Proceedings of the International Conference on E-Business And Telecommunication Networks (ICETE)*, pp. 113–118. INSTICC Press, 2004.

[FMG06] Z. Furqan, S. Muhammad, and R. K. Guha. "Formal Verification of 802.11i using Strand Space Formalism." In *Proceedings of the 5th International Conference on Networking (ICN)*. IEEE Computer Society Press, April 2006.

[FMG07] Z. Furqan, S. Muhammad, and R. K. Guha. "Authentication Analysis of the 802.11i Protocol." *International Journal of Information Technology*, **4**(1):61 – 67, 2007.

[FMS01] S. Fluhrer, I. Mantin, and A. Shamir. "Weaknesses in the key scheduling algorithm of RC4." In *Lecture Notes in Computer Science*, pp. 1–24, 2001.

[GF02] Joshua D. Guttman and F. Javier Thayer Fábrega. "Authentication tests and the structure of bundles." *Theoretical Computer Science*, **283**:333–380, June 2002.

[GFM07a] R. K. Guha, Z. Furqan, and S. Muhammad. "A Multi-agent Approach Toward the Security Analysis of the 802.11i Handshake Protocol." In *Proceedings of the International Symposium on Wireless Pervasive Computing (ISWPC)*, February 2007.

[GFM07b] R. K. Guha, Z. Furqan, and S. Muhammad. "A Tutorial on Wireless Network Security." In *Proceedings of the International Symposium on Wireless Pervasive Computing (ISWPC)*, February 2007.

[GNY90] Li Gong, Roger Needham, and R. Yahalom. "Reasoning About Belief in Cryptographic Protocols." In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pp. 234–248, May 1990.

[Gon91] L. Gong. "Handling infeasible specifications of cryptographic protocols." In *Proceedings of the IEEE Computer Security Foundations Workshop IV*, pp. 99–102, 1991.

[GS91] K. Gaarder and E. Snekkenes. "Applying a formal analysis technique to the CCITT X.509 strong two-way authentication protocol." *Journal of Cryptology*, **3**:81–98, January 1991.

[HM04] C. He and J. C. Mitchell. "Analysis of the 802.11i 4-Way Handshake." In *Proceedings of the Third ACM International Workshop on Wireless Security*, 2004.

[HM05] C. He and J. C. Mitchell. "Security analysis and improvements for IEEE 802.11i." In *Proceedings of the 12th Annual Network and Distributed System Security Symposium*, 2005.

[HP03] J. Y. Halpern and R. Pucella. "On the relationship between strand spaces and multi-agent systems." *ACM Transactions on Information and System Security*, **6**(1):43–70, 2003.

[HR04] M. Huth and M. Ryan. *Logic in Computer Science*. Cambridge Univ., 2004.

[IEE99] IEEE Standard 802.11. "IEEE Standards for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area network – Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.", 1999. 1999 Edition.

[IEE04a] IEEE P802.11i/D10.0. "Medium Access Control (MAC) security enhancements, amendment 6 to IEEE standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications.", 2004.

[IEE04b] IEEE Standard 802.1X. "IEEE standard for local and metropolitan area networks - Port-based Network Access Control - Revision of IEEE Std 802.1X-2001.", 2004. 2004 Edition.

[KBC97] H. Krawczyk, M. Bellare, and R. Canetti. "HMAC: Keyed-Hashing for Message Authentication. RFC 2104.", February 1997.

[Kem89] R. Kemmerer. "Using formal methods to analyze encryption protocols." *IEEE Journal on Selected Areas in Communications*, **7**(4):448–457, 1989.

[KG91] R. Kailar and V. D. Gilgor. "On belief evolution in authentication protocols." In *Proceedings of the IEEE Computer Security Foundations Workshop IV*, pp. 103–116, 1991.

[KMM94] R. A. Kemmerer, Catherine A. Meadows, and J. Millan. "Three Systems for Cryptographic Protocol Analysis." *Journal of Cryptology*, **7**(2):79–130, 1994.

[KV03] P. Kyasanur and N. Vaidya. "Detection and handling of MAC layer misbehavior in wireless networks." In *International Conference on Dependable Systems and Networks*, June 2003.

[KW99] Darrell Kindred and Jeannette M. Wing. "Theory Generation for Security Protocols." *TOPLAS*, **7**, 1999.

[Low96] G. Lowe. "Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR." In *Tools and Algorithms for the Construction and Analysis of Systems, 2nd International Workshop TACAS'96*, LNCS 1055, pp. 147–166. Springer Verlag, March 1996.

[Low97] G. Lowe. "A Hierarchy of Authentication Specification." In *Proceedings of the 1997 IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 31–43, 1997.

[Low99] G. Lowe. "Towards a completeness results for model checking security protocols." *Journal of Computer Security*, **7**(2), 1999.

[LR92] D. Longley and S. Rigby. "An automatic search for security flaws in key management schemes." *Computers and Security*, **11**(1):75–90, 1992.

[MB93] W. Mao and C. Boyd. "Towards formal analysis of security protocols." In *Proceedings of the Computer Security Foundations Workshop VI*, pp. 147–158, 1993.

[MC87] J. K. Millen, S. C. Clark, , and S. B. Freedman. "The Interrogator: protocol security analysis." *IEEE Transactions on Software Engineering*, **13**(2):274–288, February 1987.

[Mea94] Catherine A. Meadows. "The NRL Protocol Analyzer: An Overview." *Journal of Logic Programming*, **19/20**:19 pages, 1994.

[Mea03] C. Meadows. "Formal methods for cryptographic protocol analysis: Emerging issues and trends." *IEEE Journal on Selected Areas in Communications*, **21**(1):44–54, January 2003.

[MFGa] S. Muhammad, Z. Furqan, and R. K. Guha. "Analyzing Authentication in Kerberos-5 using Distributed Temporal Protocol Logic." *Accepted for Publication in the Journal of Information and Computing Science*.

[MFGb] S. Muhammad, Z. Furqan, and R. K. Guha. "A Logic Based Verification Framework for Authentication Protocols." *Accepted for Publication in the International Journal of Internet Technology and Secured Transactions (IJITST)*.

[MFG06a] S. Muhammad, Z. Furqan, and R. K. Guha. "Designing Authentication Protocols: Trends and Issues." In *Proceedings of the 5th International Conference on Networking (ICN), IEEE Computer Society Press*, April 2006.

[MFG06b] S. Muhammad, Z. Furqan, and R. K. Guha. "Logic-Based Formal Analysis of Cryptographic Protocols." In *14th IEEE International Conference on Networks (ICON 2006)*, September 2006.

[MFG06c] S. Muhammad, Z. Furqan, and R. K. Guha. "A Parameterized Analysis of Public-Key Protocols: Needham-Schroeder and Kerberos-5." In *14th IEEE International Conference on Networks (ICON 2006)*, September 2006.

[MFG06d] S. Muhammad, Z. Furqan, and R. K. Guha. "Understanding the Intruder through Attacks on Cryptographic Protocols." In *Proceedings of the 44th ACM Southeast Conference (ACMSE2006)*, pp. 667–672, March 2006.

[MGF04a] S. Muhammad, R. K. Guha, and Z. Furqan. "A Dynamic Simulation Model and Testing Techniques for Security Protocol Verification." *WSEAS Transactions on Computers*, **3**(5):1226–1231, November 2004.

[MGF04b] S. Muhammad, R. K. Guha, and Z. Furqan. "A Dynamic Simulation Model and Testing Techniques for Security Protocol Verification." In *Proceedings of 4th WSEAS Int. Conf. on Information Science, Communications And Applications (ISA 2004)*, April 2004.

[MMS97] John C. Mitchell, Mark Mitchell, and Ulrich Stern. "Automated Analysis of Cryptographic Protocols Using Murϕ." In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pp. 141–151, May 1997.

[Mos89] L. Moser. "A logic of knowledge and belief for reasoning about computer security." In *Proceedings of the IEEE Computer Security Foundations Workshop II*, pp. 57–63, June 1989.

[MRH04] V. Moen, H. Raddum, and K. J. Hole. "Weakness in the Temporal Key Hash of WPA." *ACM SIGMOBILE Mobile Computing and Communications Review*, **8**(2):76–83, April 2004.

[MS01] I. Mantin and A. Shamir. "A practical attack on broadcast RC4." In *Proceedings of the 8th International Workshop on Fast Software Encryption*, 2001.

[Nat01] National Institute of Standards and Technology. "FIPS Pub 197: Advanced Encryption Standard (AES).", November 2001.

[Nat06] National Institute of Standards and Technology. "SP 800-97: Guide to IEEE 802.11i: Establishing Robust Security Networks.", June 2006.

[Neo03] D. Neoh. "Corporate Wireless LAN: Know the risks and best practices to mitigate them.", December 2003. GSEC version 1.4b option 1.

[Nes90] D. M. Nessett. "A critique of the burrows, abadi and needham logic." *Operating Systems Review*, **24**(2):35–38, April 1990.

[NS78] Roger M. Needham and M. Schroeder. "Using Encryption for authentication in large networks of computers." *Communications of the ACM*, **21**(12):993–999, December 1978.

[ORS92] S. Owre, J. M. Rushby, and N. Shankar. "PVS: A prototype verification system." In *11th International Conference on Automated Deduction (CADE)*, LNCS 607, pp. 748–752. Springer Verlag, 1992.

[Pau98] Lawrence C. Paulson. "The Inductive Approach to Verifying Cryptographic Protocols." *Journal of Computer Security*, **6**:85–128, 1998.

[PD03] J. S. Park and D. Dicoi. "WLAN security: current and future." *IEEE Internet Computing*, **7**(5):60–65, 2003.

[Ran88] P. V. Rangan. "An axiomatic basis of turst in distrbiuted systems." In *Proceedings of the 1988 Symposium on Security and Privacy*, pp. 204–211, May 1988.

[RH93] A. D. Rubin and Peter Honeyman. "Formal Methods for the Analysis of Authentication Protocols." Technical Report CITI TR 93-7, University of Michigan, Ann Arbor, MI, USA, October 1993.

[RN95] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[SBP01] D. Song, S. Berezin, and A. Perrig. "Athena: a novel approach to efficient automatic security protocol analysis." *Journal of Computer Security*, **9**:47–74, 2001.

[Sch97] Steve Schneider. "Verifying Authentication Protocols with CSP." In *10th IEEE Computer Security Foundation Workshop*, pp. 3–17, 1997.

[SIR02] A. Stubblefield, J. Ioannidis, and A. Rubin. "Using the Fluhrer, Mantin, and Shamir attack to break WEP." In *Network and Distributed Systems Symposium*, 2002.

[Sne92] E. Snekkenes. "Roles in cryptographic protocols." In *Proceedings of the IEEE Computer Society Symposium on Security and Privacy*, pp. 105–119, 1992.

[SO94] P. Syverson and P. Van Oorschot. "On Unifying Some Cryptographic Protocol Logics." In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 14–28, 1994.

[SWA05] D. Stanley, J. Walker, and B. Aboba. "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs. RFC 4017.", March 2005.

[Syv90] P. Syverson. "Formal semantics for logics of cryptographic protocols." In *Proceedings of the IEEE Computer Security Foundations Workshop III*, pp. 32–41, June 1990.

[Wal00] J. R. Walker. "Unsafe at any key size; an analysis of the WEP encapsulation.", 2000. IEEE Document 802.11-00/362.

[WHF03] D. Whiting, R. Housley, and N. Ferguson. "Counter with CBC-MAC (CCM). RFC 3610.", September 2003.

[WL93] T. Y. C. Woo and S. S. Lam. "A Semantic Model for Authentication Protocols." In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 178–194, 1993.