ANALYSIS OF SECURITY PROTOCOLS FOR WIRELESS NETWORKS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF
ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Changhua He
December 2005

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

John C. Mitchell (Principal Advisor)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Fouad Tobagi (Co-Advisor)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Dan Boneh

Approved for the University Committee on Graduate Studies.

Abstract

Security is a serious concern in wireless networks. In order to eliminate the vulnerabilities in previous Standards, the IEEE 802.11i Standard is designed to provide security enhancements in MAC layer. The authentication process consists of several components, including an 802.1X authentication phase using TLS over EAP, a 4-Way Handshake to establish a fresh session key, and an optional Group Key Handshake for group communications. The objective of this work is to analyze IEEE 802.11i with respect to data confidentiality, integrity, mutual authentication, and availability.

Under our threat model, 802.11i appears to provide effective data confidentiality and integrity when CCMP is used. 802.11i may also provide satisfactory mutual authentication and key management, although there are some potential implementation oversights that may cause severe problems. On the other hand, we identified several Denial of Service attacks. Different solutions are proposed for these vulnerabilities, which result in an improved variant of 802.11i with a more effecient failure recovery mechanism. Some of the resulting improvements have been adopted by the IEEE 802.11 TGi in their final deliberation.

We used a finite-state verification tool, called Murϕ, to analyze the 4-Way Handshake component. Our result shows that finite-state verification is quite effective for analyzing security protocols. Furthermore, we adopted Protocol Composition Logic to conduct a correctness proof of 802.11i, including SSL/TLS as a component. The proof is modular, comprising a separate proof for each protocol component and providing insight into the networking environment in which each component can be reliably used. Finally, we showed that 802.11i can significantly reduce the complexity of designing a secure routing protocol when it is deployed in wireless ad hoc networks.

Contents

Abstract **v**

1 Introduction **4**

1.1 Security Requirements 5

1.1.1 Wired vs. Wireless 6

1.1.2 Link Layer vs. Upper Layer Authentication 7

1.2 Wireless Security Evolution 8

1.2.1 Data Confidentiality, Integrity, Authentication 8

1.2.2 Availability and DoS Attacks 12

1.3 Thesis Outline 14

2 IEEE 802.11i Security Analysis **17**

2.1 Wireless Threats 18

2.2 Data Confidentiality and Integrity 22

2.3 Authentication and Key Management 24

2.3.1 RSNA Establishment Procedure 24

2.3.2 RSNA Security Analysis 28

2.3.3 Security Level Rollback Attack 30

2.3.4 Reflection Attack 32

2.4 Availability 33

2.4.1 Known DoS Attacks and Defenses 34

2.4.2 Michael Algorithm Countermeasure 35

2.4.3 RSN IE Poisoning 37

2.4.4 4-Way Handshake Blocking 42
2.4.5 Failure Recovery 44
2.4.6 Improved 802.11i 45
2.5 Conclusion 47

3 Finite-State Verification **49**
3.1 Finite-State Verification 50
3.1.1 The Murϕ Model Checker 50
3.1.2 The Verification Procedure 52
3.1.3 Limitations 53
3.2 Verification of the 4-Way Handshake 54
3.2.1 The 4-Way Handshake 54
3.2.2 The Protocol Model 56
3.2.3 The Protocol Clarifications 57
3.3 DoS Attack 59
3.3.1 The DoS attack 59
3.3.2 Parallel Instances 61
3.3.3 Limitations 62
3.3.4 Practicality 63
3.4 Effective Defenses 65
3.4.1 Random-Drop Queue 65
3.4.2 Message 1 Authentication 67
3.4.3 Nonce Reuse 68
3.5 Conclusions 69

4 A Modular Correctness Proof **71**
4.1 Overview 73
4.1.1 The 802.11i Control Flow 73
4.1.2 The Proof Method 74
4.2 4-Way Handshake 77
4.2.1 Modelling 4-Way 78
4.2.2 Security Properties 79

4.2.3 Operating Environment 81
4.2.4 Improved 4-Way Handshake 82
4.3 TLS 84
4.3.1 Modelling TLS 84
4.3.2 Security Properties 86
4.3.3 Operating Environment 87
4.4 Group Key Handshake 88
4.4.1 Modelling Group Key Handshake 88
4.4.2 Security Properties 89
4.4.3 Operating Environment 91
4.5 Composition 92
4.5.1 Staged Composition 93
4.5.2 Composition of 802.11i 95
4.6 Conclusions 97

5 Using 802.11i in Ad Hoc Routing 99
5.1 Secure Routing 99
5.2 Dynamic Source Routing with 802.11i 101
5.2.1 The Original DSR 101
5.2.2 The Improved DSR 102
5.3 Security Analysis 105
5.3.1 DSR Property 105
5.3.2 One Malicious Node 107
5.3.3 Multiple Compromised Nodes 109
5.4 Availability 110
5.5 Conclusions 112

6 Conclusions 114
6.1 Deployment Considerations 114
6.2 Conclusions 116
6.2.1 IEEE 802.11i Security Analysis 117
6.2.2 Finite-State Verification 118

6.2.3 A Modular Correctness Proof 119
6.2.4 Using 802.11i in Ad Hoc Routing 120

A Protocol Composition Logic **122**
A.1 Programming Language 122
A.2 Protocol Logic 123
A.3 Proof System 125
A.3.1 Axioms for Protocol Actions 125
A.3.2 Axioms for PTK derivation 125
A.3.3 Possession Axioms 126
A.3.4 Encryption and Signature 127
A.3.5 Generic Rules 127
A.3.6 Sequencing rule 127
A.3.7 Preservation Axioms 128
A.3.8 Axioms and Rules for Temporal Ordering 128
A.3.9 Honesty Rule 128

B Proof of the 4-Way Handshake Guarantee **129**

Bibliography **135**

List of Tables

4.1 4-Way Handshake Program 78
4.2 4-Way Handshake Precondition and Invariants 82
4.3 TLS: Client and Server Programs 85
4.4 TLS Invariants 88
4.5 Group Key Handshake Programs 89
4.6 Group-Key Protocol Precondition and Invariants 92

List of Figures

2.1 RSNA Establishment Procedure 25
2.2 Security Level Rollback Attack 31
2.3 Reflection Attack on the 4-Way Handshake 32
2.4 TKIP MPDU Format 35
2.5 RSN Information Element Format 38
2.6 RSN IE Poisoning 39
2.7 4-Way Handshake Blocking 43
2.8 A Flow Chart of the Improved 802.11i 46

3.1 The Idealized 4-Way Handshake Protocol 55
3.2 The Simplified 4-Way Handshake Protocol 59
3.3 The one-message attack on the 4-Way Handshake protocol 60
3.4 Effectiveness of random-drop queue 66

4.1 The 802.11i Control Flow 75
4.2 Arrows-and-messages vs cords 76

5.1 DSR route discovery 103
5.2 DSR Shortest Path Property 106
5.3 The set of routes DSR can return 107
5.4 The set of returned routes with one malicious node 109
5.5 The set of returned routes with multiple malicious nodes 110
5.6 The rate limit problem 112

List of Acronyms

AAA	Authentication, Authorization, and Accounting
ACL	Access Control List
AES	Advanced Encryption Standard
AP	Access Point
ARP	Address Resolution Protocol
BSS	Basic Service Set
CBC-MAC	Cipher Block Chaining Message Authentication Code
CCA	Clear Channel Assessment
CCM	Counter with CBC-MAC
CCMP	Counter-mode/CBC-MAC Protocol
CRC	Cyclic Redundancy Checksum
DoS	Denial of Service
DSR	Dynamic Source Routing
DSSS	Direct Sequence Spread Spectrum
EAP	Extensible Authentication Protocol
EAPOL	Extensible Authentication Protocol Over Local Area Network
EAP-TLS	Extensible Authentication Protocol - Transport Layer Security
FCS	Frame Check Sequence
FHSS	Frequency Hopping Spread Spectrum
FSM	Finite-State Machine
GTK	Group Transient Key
ICMP	Internet Control Message Protocol
ICV	Integrity Check Value

IEEE	Institute of Electrical and Electronics Engineers
IPsec	Internet Protocol Security
IV	Initialization Vector
KCK	Key Confirmation Key
KEK	Key Encryption Key
LAN	Local Area Network
MAC	Media Access Control
MIC	Message Integrity Code
MitM	Man-in-the-Middle
MPDU	MAC Protocol Data Unit
MSDU	MAC Service Data Unit
NIC	Network Interface Card
MSK	Master Session Key
OFDM	Orthogonal Frequency Division Multiplexing
PCL	Protocol Composition Logic
PHY	PHYsical Layer
PMK	Pairwise Master Key
PMKID	Pairwise Master Key Identifier
PRF	Pseudo Random Function
PSK	Pre-Shared Key
PTK	Pairwise Transient Key
RADIUS	Remote Authentication Dial In User Service
RC4	Rivest Cipher 4 (Ron's Code)
RF	Radio Frequency
RSNA	Robust Security Network Association
RSN IE	Robust Security Network Information Element
SET	Secure Electronic Transaction
SIFS	Short Inter-Frame Space
SSH	Secure SHell
SSID	Service Set IDentifier
SSL	Secure Sockets Layer

TK	Temporal Key
TKIP	Temporal Key Integrity Protocol
TLS	Transport Layer Security
TPTK	Temporary Pairwise Transient Key
TSC	TKIP Sequence Counter
TSN	Transient Security Network
VPN	Virtual Private Network
WEP	Wired Equivalent Protocol
WiFi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WPA	WiFi Protected Access

Chapter 1

Introduction

Wireless networks represent a rapidly emerging area of growth and importance for providing ubiquitous networking connections. The common technologies can be classified into different categories according to the range of the service area. On a worldwide scale, telecommunication companies have been making significant progress in carrying voice and data traffic over their cellular networks; furthermore, the next generation infrastructure, under developpment all over the world, aims to provide higher bandwidth and better quality for multimedia traffic. In a metropolitan area, WiMAX (IEEE 802.16) can provide users with high-speed broadband access to the Internet. In a local area, WiFi (IEEE 802.11) enables users to establish wireless connections within a corporate or campus building. Moreover, in a personal area (often less than 10 meters), bluetooth (IEEE 802.15) can provide low-cost and short-range connectivity for portable devices.

The focus of this dissertation is Wireless Local Area Networks (WLAN) based on IEEE 802.11a/b/g specifications [51, 52, 53, 54]. Compared with the current cellular networks, a WLAN system has much higher transmission rates and shorter transmission range; hence, it is suitable for home networking, small business, and large corporations and has been widely deployed since IEEE 802.11b first appeared in 1999. However, along with the popularity of WLAN, security is a serious concern because the wireless medium is open for public access within a certain range.

In 2001, Shipley [95] invented the first "War-driving" in the Bay Area to find

existing wireless networks. By driving a car along certain route through a district, people can discover the operating Access Points (APs), the corresponding Service Set Identifier (SSID), and even the physical locations of the APs, with only moderate equipment [33]. Obviously these capabilities release sensitive information and unauthorized services to an outsider. Furthermore, if the discovered APs are not well-configured, the outsider is able to exploit bandwidth for free Internet access, steal confidential data for malicious usage, or install advanced attacks from this open base. Even worse, the legitimate user may be unaware of these activities because the outsider can physically stay inside his car along the road or in the parking lot. These dangers impose necessary requirements on the security of WLAN implementations.

1.1 Security Requirements

In a general network system, security has different contexts depending on different applications, among which the essential requirements are data confidentiality and integrity, authentication, and availability.

Data Confidentiality and Integrity The network MUST provide strong data confidentiality, integrity, and replay protection for every transmitted message. Data confidentiality and integrity, helping build a secure channel for the user to communicate in an insecure environment, mean that only the communicating users are able to understand the received messages, generate or modify valid messages. Furthermore, 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.

Mutual Authentication The network MUST provide mutual authentication, which means that the communicating peers authenticate each other's identity. If required, the authentication process should also combine with key generation, distribution and management to provide secret keys for the cryptographic function. Based on the

authentication results, flexible authorization and access control policies could be deployed to restrict the privilege of users.

Availability Availability is a form of robustness, which is another important category of security requirements. The network should be able to prevent an adversary from shutting down the connectivity for a legitimate individual or the entire system. In other words, Denial of Service (DoS) attacks should be eliminated, or at least mitigated.

1.1.1 Wired vs. Wireless

Since there are already many security technologies developed for wired networks, it is important to understand that there are more security challenges in wireless networks than in wired networks, due to their different characteristics.

First of all, wireless medium access is open to everybody within the transmission range. In wired networks, it is possible to secure the connection by hiding the wires inside walls and restricting the access to outlets. However, in WLAN it is difficult to shape and limit the radio transmission range; an outsider could access the traffic without any suspicion from the administrator because there is no need for the outsider to plug his device into a jack, or appear in sight of the administrator. For example, an attacker could be physically located miles away, interfering with the wireless network through high gain antennas.

Second, in wired networks, end users could have some confidence on the validity of the networks they connect to. For example, when a user plugs his device into a jack on the wall inside a corporation, it is known that the network access is provided by the corporation. However, in wireless networks, a user is completely blind about the connected network, because the user can only have a view of the network from the associated AP, which is invisible and might be malicious.

Third, the wireless links are more vulnerable to the environment than a wired one, causing a relatively high loss rate and bit error rate. Hence, protocols in wireless networks should be designed to allow message loss and damage, which gives an attacker more chances for attempt.

Fourth, the connections between wireless devices are adaptive and flexible according to the user mobility and link quality, which is the desired advantage over the wired networks, but causing a more complicated trust relationship.

Finally, the cryptographic operations should adapt to the computation and power restraints of wireless devices; the authentication and key management protocols should be scalable and ubiquitous to support user mobility; furthermore, due to the inherent vulnerabilities of wireless channels, it is much more difficult to defend against DoS attacks in a wireless environment.

1.1.2 Link Layer vs. Upper Layer Authentication

The 802.11 Standards deal with the Physical Layer (PHY) and Media Access Control (MAC) which is part of the Link Layer. The PHY specifications, including 802.11 FHSS (Frequency Hopping Spread Spectrum), DSSS (Direct Sequence Spread Spectrum), and OFDM (Orthogonal Frequency Division Multiplexing), defines the physical signal transmit, receive, and the CCA (Clear Channel Assessment). The MAC specification defines the protocol to access the shared media. For example, the frame scheduling, acknowledgement, retransmission, collision avoidance, and so on. The other Upper Layers in wireless networks are the same as in common wired networks. In order to satisfy the security requirements, the authentication mechanism could be implemented in either Link Layer or Upper Layers. In Wireless Local Area Networks, Link Layer authentication is adopted due to the following reasons. First, the original mechanism in 802.11 documentation aims to provide Wired Equivalent Privacy (WEP) to protect the wireless link, thus, other existing mechanisms in wired networks can still work as usual by treating the wireless link as a wired one. The more advanced mechanisms developped later also inherit this Link Layer authentication for reusing legacy devices. Second, it is fast, simple, and inexpensive to perform authentication in the Link Layer. Compared with Upper Layer authentication [3], the devices capable of processing link layer frames are fairly cheap; the delay of exchanging frames in the link layer is much smaller since there is no need to pass through

other upper layers and multiple links; and this provides a transparent implementation independent of the protocols running on top of the link layer. Furthermore, since the authentication is done at the edge of the system and at the beginning of the session, the clients do not need any network access prior to authentication. For example, the clients do not need to resolve names or obtain an IP address; the routing information does not interact with the authentication. This is good because allowing partial network access prior to authentication might introduce vulnerabilities into the system.

In this dissertation we focus on link layer authentication mechanisms. 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 end-to-end security in different layers, and can be implemented with link layer authentication together for better security. The details of these techniques are out of the scope of this dissertation.

1.2 Wireless Security Evolution

During the evolution of WLAN security, many efforts have been made to achieve data confidentiality, integrity and mutual authentication. There are some interesting lessons along the way, and the objectives are not met until IEEE 802.11i appeared. Section 1.2.1 is a review of the literature on proposed approaches and their deficiencies. Furthermore, availability has been ignored in some way which results in numerous DoS attacks in a WLAN system. Section 1.2.2 describes the possible DoS attacks.

1.2.1 Data Confidentiality, Integrity, Authentication

In order to provide data confidentiality equivalent to a wired network, the IEEE 802.11 Standard [51] originally defines Wired Equivalent Privacy (WEP). This mechanism adopts a common stream cipher known as RC4 (Rivest Cipher 4) [92], to encrypt

messages with a shared key. This key is concatenated with a 24-bit Initialization Vector (IV) to construct a per-packet RC4 key. In order to provide data integrity, WEP calculates an Integrity Check Value (ICV) over the MSDU (MAC Service Data Unit), which is a common Cyclic Redundancy Checksum (CRC). The frame body, together with the corresponding ICV, is encrypted using the per-packet key. In addition, two authentication mechanisms are defined: the Open System Authentication, which is actually a null authentication, and the Shared Key Authentication, which is a Challenge-Response handshake based on the shared key.

However, numerous studies have shown that data confidentiality, integrity, and authentication are not achieved through these mechanisms. First, the 40-bit shared key is too short for brute-force attacks [16, 82]. Though some vendors might support a longer key (128 bits, containing a 104-bit key and a 24-bit IV), it is still easy for an adversary to recover the plaintext traffic because the small IV size and the static shared key result in a high possibility of key stream reuse [16, 99], which trivially defeats any stream cipher. Furthermore, the concatenation of the IV and the shared key has inherent weakness for generating the per-packet RC4 key [39]; an adversary can discover the key by eavesdropping several million packets [97]. Moreover, because ICV is a linear and unkeyed function of the message [16], data integrity cannot be guaranteed; even without any knowledge of the key stream, an adversary is able to arbitrarily modify a packet without detection, or forge a packet with a valid ICV. This weak integrity also enables much easier plaintext recovery, as in IP redirection, reaction attacks [16], and inductive chosen plaintext attacks [7]. Finally, an adversary can trivially spoof the Shared Key Authentication [8, 16], through observing an authentication process of a legitimate station. Additionally, WEP does not implement any mechanism to prevent replay attacks. FBI agents have publicly demonstrated that they can break a 128-bit WEP key in about three minutes [22].

Although WEP fails to satisfy any security requirements, it is not practical to expect users to completely discard their devices with WEP already implemented. Hence, the WiFi Alliance proposed an interim solution, called WiFi Protected Access (WPA), to ameliorate the vulnerabilities while reusing the legacy hardware. WPA adopts a Temporal Key Integrity Protocol (TKIP) for data confidentiality, which still

uses RC4 for data encryption, but includes a key mixing function and an extended IV space to construct unrelated and fresh per-packet keys. WPA also introduces the Michael algorithm [50], a weak keyed Message Integrity Code (MIC), for improved data integrity under the limitation of the computation power available in the devices. Furthermore, in order to detect replayed packets, WPA implements a packet sequencing mechanism by binding a monotonically increasing sequence number to each packet.

In addition, WPA provides two improved authentication mechanisms. In one mechanism, the possession of a Pre-Shared Key (PSK) authenticates the peers; furthermore, a 128-bit encryption key and another distinct 64-bit MIC key can be derived from the PSK. Alternatively, IEEE 802.1X [55] and the Extensible Authentication Protocol (EAP) [15] can be adopted to provide a stronger authentication for each association, and generate a fresh common secret as part of the authentication process; all required keys can be derived from this shared secret afterwards.

TKIP is proposed to address all known vulnerabilities in WEP; it does enhance the security in all aspects. However, there are weaknesses in WPA due to the limitation of reusing the legacy hardware. Although the TKIP key mixing function has stronger security than the WEP key scheduling algorithm, it is not as strong as expected. It is possible to find the MIC key given one per-packet key; furthermore, the whole security is broken for the duration of a Temporal Key (TK) given two per-packet keys with the same IV32 [75]. This vulnerability does not mean that TKIP is insecure; however, it discloses that parts of TKIP are weak on their own. Furthermore, the Michael algorithm is designed to provide only 20 bits (or possible slightly more) of security in order to minimize the impact on the performance, which means an adversary can construct one successful forgery every 2^{19} packets. Thus, countermeasures are necessary to limit the rate of the forgery attempts [50]. However, this countermeasure may allow DoS attacks. In addition, the 802.1X authentication may be vulnerable to Session Hijacking and Man-in-the-Middle (MitM) attacks [70]. Though these attacks disappear when mutual authentications and strong encryption are used [23], there are deficiencies of using 802.1X, which is originally designed for a switched LAN, to implement in a shared media WLAN.

As a long-term solution, IEEE 802.11i [50] is proposed to provide an enhanced MAC layer security. 802.11i provides authentication protocols, key management protocols, and data confidentiality protocols that may execute concurrently over a network in which other protocols are also used. Under the assumption of upgrading the hardware, 802.11i defines a Counter-mode/CBC-MAC Protocol (CCMP) that provides strong confidentiality, integrity, and replay protection. In addition, an authentication process, combining the 802.1X authentication and key management procedures, is performed to mutually authenticate the devices and generate a fresh session key for data transmissions. Since 802.11i promises to be the right solution for wireless security, it should be able to prevent an adversary from advanced attacks even if the adversary could have the most powerful equipments and techniques for breaking into the system. In other words, an implementation of 802.11i protocols in a WLAN should be able to provide sufficient data confidentiality, integrity, and mutual authentication.

Some vendors have adopted a MAC-address-based Access Control List (ACL) for authorization. Specifically, only stations with their MAC addresses existing in the ACL list are able to access the network. This method is weak since an adversary can obtain valid MAC addresses easily by sniffing the traffic [8]. Another mechanism, called Closed System Authentication, is also proposed for access control, by disabling the SSID broadcast in the Beacon frame. The SSID is treated as a secret; only the user knowing the SSID can join the network. However, this approach is worthless because an adversary can obtain the SSID from the cleartext of Probe Request/Response frame or (Re)Association Request frame of a legitimate user [61]. In this dissertation, the requirements of authorization and access control are not considered; instead, mutual authentication is required, following which flexible authorization and access control policies could be implemented.

There are also many other approaches to secure a WLAN from other layers than MAC. From the Physical Layer (PHY), proper antenna selecting and positioning can reduce the signal leakage, thus, enhance the security [64]. It is also possible to implement a RF Firewall architecture to protect the WLAN [89], but this requires significant modifications on the 802.11 PHY. In addition, some vendors have adopted

different existing protocols to secure network connections through the IP or upper layer, such as IPsec, SSL, and SSH. As we have discussed in Section 1.1.2, these mechanisms have their own roles for security and solve the problem in different layers.

There are some novel approaches [37] appearing recently to reduce the complexity of the key management in a WLAN implementation, in which a wireless device authenticates itself to the system by presenting within a limited physical range controlled by the system, depicted by the transmission power control and CRC error omission. Although this approach appears to be only one-direction authentication (the client authenticates to the system), and it depends on the accuracy of localization, it is a valuable solution since it significantly reduces the complexity of setting up a WLAN system.

1.2.2 Availability and DoS Attacks

The past studies have extensively focused on the data confidentiality, integrity, and mutual authentication for wireless security. However, availability has not been considered sufficiently. Many DoS attacks have been disclosed on the WLAN systems from the Physical Layer to the Application Layer. Some might think that the DoS attacks are inevitable due to the physical characteristics of wireless links. However, since many DoS attacks can be mounted by an adversary with moderate equipments, and a successful DoS attack may facilitate other advanced attacks, such as Session Hijacking and Man-in-the-Middle (MitM), they should be considered to be real threats to a WLAN implementation. The key point to mitigate these attacks is to impose relatively higher cost for an adversary, e.g., more computation power, more message transmissions, or more memory consumption, which could make the DoS attacks impractical.

In the Physical Layer, a straightforward DoS attack is frequency jamming: an adversary can interfere the whole frequency band with a strong noise signal, blocking the legitimate data transmissions. This appears to be inevitable. Fortunately, it is relatively expensive because the adversary needs special equipments and huge power consumption to jam the whole spectrum. Also spread spectrum technology

can be adopted in wireless networks to make the frequency jamming more difficult. Furthermore, an adversary performing this attack can be easily detected and located by a network administrator. Therefore, it is reasonable to assume that an adversary will not try to launch this attack for common purposes except military. There exists another easier approach to mount a frequency jamming in a WLAN implementing Direct Sequence Spread Spectrum (DSSS) [11, 101]. By exploiting the Clear Channel Assessment (CCA) procedure, an adversary can cause all WLAN nodes within range to consider the channel busy and defer transmissions of any data. Particularly, most vendors do not remove the engineering function PLME-DSSSTESTMODE from their released products, which makes the attack more convenient through the off-the-shelf usage of a common wireless Network Interface Card (NIC). There are no complete solutions for this DoS attack yet. Fortunately, the attack only affects a WLAN system implementing CCA, which is in DSSS (e.g., 802.11b/g), but not in OFDM (e.g., 802.11a/g).

In the MAC layer, an adversary can scramble the channel by MAC preemptive jamming because the WLAN is designed to be cooperative. For example, the adversary can send out a short jamming noise in every time interval of SIFS (Short Inter-Frame Space, 10μs in 802.11b networks), which will surely collide with all the legitimate traffic, or cause the legitimate traffic to be deferred infinitely. However, this attack is not considered to be a real threat because the adversary needs to send out about 50,000 packets per second in an 11 Mbps 802.11b network [13]. As another possible attack, an adversary is able to transmit legitimate messages, without obeying the standard. Specifically, the adversary could use a smaller "backoff"time, in order to obtain an unfair allocation of the channel bandwidth. If the adversary adopts no "backoff", s/he may ultimately cause a DoS attack for legitimate users [58]. More DoS vulnerabilities arise from the unprotected management frames and control frames. An adversary is able to easily launch a DoS attack on a specific station or the entire Basic Service Set (BSS) by forging the Deauthentication, Disassociation, Traffic Indication Map (TIM), or Poll messages [13]. Furthermore, DoS attacks could be mounted by exploiting the virtual carrier-sense scheme through forging any frame,

especially RTS (Ready To Send) frame, with an extremely large value of NAV (Network Allocation Vector), which can fool the devices to consider the channel busy; thus, suppress the device from transmitting messages [13, 21]. Additionally, as in a wired LAN, an adversary can perform an ARP (Address Resolution Protocol) cache poisoning to mount a DoS attack if the adversary is able to access the network in someway [38].

Furthermore, if an IEEE 802.1X authentication is implemented for stronger authentication, the adversary has more choices to mount a DoS attack through forging EAP-Start, EAP-Logoff, and EAP-Failure messages. The adversary can also exhaust the space of the EAP packet identifier, which is only 8 bits long, by sending more than 255 authentication requests simultaneously [8].

In addition, due to the speed limitation of a WLAN comparing to a wired network, it is easy to perform a DoS attack from the IP or upper layer by simple network jamming, e.g., ICMP ping flooding from a high-speed wired network [79]. However, this attack could be prevented by appropriate filtering policies and traffic shaping in the Access Point. An adversary can also exploit the deficiencies of the upper layer protocols to mount a DoS attack; improvements on the upper layer protocol might be necessary for countermeasures.

As a summary, in the current WLAN system, DoS attacks are very easy to mount; furthermore, once an adversary successfully mounts a DoS attack, more advanced attacks, such as MitM [61], could be subsequently constructed. Therefore, it is necessary to deploy a security mechanism that can defend against DoS attacks. Since 802.11i does not emphasize such objective, it is definitely valuable if it can be improved to mitigate DoS attacks in the Link Layer.

1.3 Thesis Outline

As we have described, IEEE 802.11i is designed to solve all recognized problems in previous protocols. In this dissertation we analyzed the security of 802.11i protocols and explored the application of 802.11i in wireless ad hoc networks. The main results in Chapter 2, Chapter 3, Chapter 4 can also be found in our published papers [41,

40, 42], respectively.

Chapter 2 describes 802.11i in detail and proposes our informal analysis results on 802.11i. We found a Denial of Service (DoS) attack on the 4-Way Handshake and proposed a simple solution, which eliminates the vulnerability inherently and requires no modifications on the packet format. We also found a reflection attack on the 4-Way Handshake, which violates the mutual authentication. Furthermore, due to the vulnerability of the Robust Security Network Information Element (RSN IE), we found a possible security level rollback attack which will rollbacks the security of the whole system to the lowest level; we also described an easy DoS attack based on RSN IE Poisoning. These vulnerabilities have gained attentions in several industrial product groups. We also found that it is practical to launch a DoS attack by exploiting the TKIP Michael countermeasure mechanism. Finally, we proposed an improved architecture for 802.11i to address all the vulnerabilities discussed in this chapter.

Chapter 3 discusses the finite-state verification methodology for security protocols, based on a model checker called Murϕ. Using 802.11i 4-Way Handshake as an example, we described the approach to verify the 4-Way handshake and how the vulnerability was found. We originally identified the 4-Way Blocking attack through this approach, which shows that the methodology is quite effective on finding bugs in protocols. In order to eliminate this vulnerability, we studied four different solutions and analyzed their effectiveness. Our final solution was adopted by IEEE 802.11 TGi (*Task Group i*) committee, which involves reusing the nonce in the supplicant (wireless station) side.

Chapter 4 conducts a modular correctness proof of 802.11i which uses EAP-TLS as the authentication method. We used Protocol Composition Logic (PCL) to prove the authentication and key secrecy properties. The proof is modular, comprising a separate proof for each protocol section and a correctness proof for the composition of all sections. Our results shows that 802.11i can provide session authentication and key secrecy under the appropriate environment; furthermore, our analysis shows that PCL is suitable for analyzing large protocols. We also proved a new staged composition theorem to handle the complicated control flow of 802.11i when different failure recovery mechanisms are considered. Throughout the proof, we identified the

specified invariants for the security properties, corresponding to the operating environments, which can help the deployment and implementations of 802.11i. Moreover, the proof of TLS/SSL has independent interest since it is widely used in Internet.

Chapter 5 discusses the application of 802.11i to wireless ad hoc networks. When every node is honest, We can simply build a secure ad hoc network by using 802.11i to establish confidential channels among neighbors and running a common routing protocol on top of that. However, this mechanism does not work in practice since one or several compromised nodes could destroy the security of the whole system. We analyzed the influence of link layer security mechanism on the routing algorithm using Dynamic Source Routing (DSR) as an example. We achieved a secure routing scheme based on per-hop confidential links and a slightly improved version of DSR. Our results show that a link layer security mechanism (i.e., 802.11i) can make it much easier to design a secure routing protocol.

Chapter 6 describes some suggestions for avoiding possible vulnerabilities when deploying 802.11i in wireless networks, and concludes the dissertation.

Chapter 2

IEEE 802.11i Security Analysis

IEEE 802.11i [50], an IEEE standard ratified on June 24, 2004, is designed to provide enhanced security in the Medium Access Control (MAC) layer for 802.11 networks. The 802.11i specification defines two classes of security algorithms: Robust Security Network Association (RSNA), and Pre-RSNA. Pre-RSNA security consists of Wired Equivalent Privacy (WEP) and 802.11 entity authentication. RSNA provides two data confidentiality protocols, called the Temporal Key Integrity Protocol (TKIP) and the Counter-mode/CBC-MAC Protocol (CCMP), and the RSNA establishment procedure, including 802.1X authentication and key management protocols. While Pre-RSNA algorithms are originally defined in IEEE 802.11 specification [51], and are included in 802.11i just for backward compatibility, RSNA algorithms are proposed to provide scalable, flexible and strong securities in wireless networks.

This chapter describes the 802.11i specification briefly and analyzes the security aspects, considering data confidentiality, integrity, mutual authentication, and availability. Our analysis suggests that 802.11i is a well-designed standard for data confidentiality, integrity, and mutual authentication, promising to improve the security of wireless networks. At the same time, some vexing Denial of Service (DoS) attacks remain. We reviewed the known DoS attacks and describe appropriate countermeasures. We also described two new DoS attacks - RSN Information Element (RSN IE) Poisoning and 4-Way Handshake Blocking - and present countermeasures for these.

Furthermore, we analyzed the failure-recovery strategy in 802.11i and discussed associated tradeoffs. Finally we outlined an improved version of 802.11i that addresses all the vulnerabilities discussed in this chapter. In proceeding through the analysis of 802.11i, we also described several implementation considerations and suggested ways to avoid a range of security problems.

The chapter is organized as follows. Section 2.1 describes threats in wireless networks. Section 2.2 analyzes the data confidentiality and integrity protocols of 802.11i. Section 2.3 analyzes the RSNA establishment procedure and indicates possible implementation mistakes. Section 2.4 discusses the practical DoS attacks and proposes an improved version of 802.11i. Section 2.5 concludes the chapter.

2.1 Wireless Threats

In order to analyze security protocols for wireless networks, it is important to characterize the likely capabilities of any adversary. From the Link Layer of a WLAN, there are three possible types of frames: Management Frames, Control Frames, and Data Frames. Any manipulation of these frames that directly or potentially jeopardizes data confidentiality, integrity, mutual authentication, and availability will be considered a threat. In this section, we outline some forms of attack that we consider and evaluate in our analysis.

Threat 1. Passive Eavesdropping and Traffic Analysis
Due to the characteristics of wireless communication, an adversary can easily sniff and store all the traffic in a WLAN. Even when messages are encrypted, it is important to consider whether an adversary may learn partial or complete information from certain messages. This possibility exists if common message fields are predictable or redundant; further, encrypted messages may be generated upon the requests from the adversary itself. In our analysis, we consider whether recorded packets and/or knowledge of the plaintext can be used to reveal the encryption key, decrypt complete packets, or gather other useful information through traffic analysis techniques.

Threat 2. Message Injection and Active Eavesdropping

An adversary is capable of inserting a message into the wireless network with moderate equipment, such as a device with a common wireless Network Interface Card (NIC) and some relevant software. Although the firmware of most wireless NICs may limit the interface for composing packets complying to the 802.11 standard, an adversary is still able to control any field of a packet using known techniques [13]. Hence, it is reasonable to assume that an adversary can generate any chosen packet, modify contents of a packet, and completely control the transmission of the packet. If a packet is required to be authenticated, the adversary may be able to break the data integrity algorithm to make a valid packet. The adversary can also insert a replayed packet, if there is no replay protection or the adversary is able to avoid it. Furthermore, by inserting some well-chosen packets, the adversary might be able to learn more information from the reaction of the system through active eavesdropping.

Threat 3. Message Deletion and Interception

We assume that an adversary is able to do message deletion, which means that an adversary is capable of removing a packet from the network before the packet reaches its destination. This could be done by interfering with the packet reception process on the receiver's antenna, for example, by causing CRC (Cyclic Redundancy Checksum) errors so that the receiver drops the packet. This process is similar to ordinary packet errors due to noise, but may be instigated by an adversary.

Message interception means that an adversary is able to control a connection completely. In other words, the adversary can capture a packet before the receiver actually receives it, and decide whether to delete the packet or forward it to the receiver. This is more dangerous than the eavesdropping and message deletion. Furthermore, it differs from eavesdropping and replaying, because the receiver does not get the packet before the adversary forwards it. Message interception may seem difficult in wireless LANs because the legitimate receiver might detect a message as soon as the adversary does so. However, a determined adversary DOES have some potential ways to achieve message interception. For example, the adversary can use a directional antenna to delete a packet on the receiver side by causing collisions, while

simultaneously using another antenna to receive the packet itself. Since message interception is relatively difficult to achieve, we only consider this possibility when the damage caused is relatively severe. Note that it is not necessary for the adversary to perform a Man-in-the-Middle (MitM) attack in order to intercept packets.

Threat 4. Masquerading and Malicious AP

Because the plaintext MAC addresses are included in all packets transmitted through wireless links, an adversary can learn valid MAC addresses by eavesdropping. The adversary is also capable of modifying its MAC address to any value because most firmware provides the interface to do so. If a system uses MAC address as the only identification of the wireless devices, the adversary can masquerade as any wireless station by spoofing its MAC address; or can masquerade as an Access Point (AP) by spoofing its MAC address and functioning appropriately through appropriate freeware (e.g., HostAP [62]). An adversary is also able to install his own AP, with a forged MAC address and a spoofed SSID. Alternatively, without masquerading as others, it is possible for a malicious AP to provide a strong signal and attempt to fool a wireless station into associating with it and leaking credentials or private data.

Threat 5. Session Hijacking

We consider that an adversary may be able to hijack a legitimate session after the wireless devices have finished authenticating themselves successfully. Here is one possible scenario for achieving this. First, the adversary disconnects a device from an existing session, and then masquerades as this device to obtain possible connections without the attention of the other device. In this attack, the adversary is able to receive all packets destined to the hijacked device and send out packets on behalf of the hijacked device. This attack could conceivably circumvent any authentication mechanism in the system. However, if data confidentiality and integrity protocols are used, the adversary must break them in order to read encrypted traffic and send out valid packets. Thus this attack against authentication can be prevented by sufficiently powerful data confidentiality and integrity mechanisms.

Threat 6. Man-in-the-Middle

This attack is different from message interception because the adversary must participate in communication continuously. If there is already a connection between a wireless station and the AP, the adversary must break this connection first. Then, the adversary masquerades as the legitimate station to associate with the AP. If the AP adopts any mechanisms to authenticate the station, the adversary must be able to spoof the authentication. And finally, the adversary must masquerade as the AP to fool the station to associate with it. Similarly, if the station adopts some mechanism to authenticate the AP, the adversary must spoof the AP's credentials. Another possible approach for the adversary to launch a MitM attack is ARP cache poisoning, as in a wired LAN [38].

Threat 7. Denial of Service

WLAN systems are quite vulnerable to DoS attacks. An adversary is capable of making the whole Basic Service Set (BSS) unavailable, or disrupting the connection between legitimate peers. Using characteristics of wireless networking, an adversary may launch DoS attacks in several ways. For example, forging the unprotected management frames (e.g., Deauthentication and Disassociation), exploiting some protocol weaknesses, or straightforward jamming of the frequency band will deny service to legitimate users. However, we only consider DoS attacks that require reasonable effort on the part of the adversary. For instance, deleting all packets, using the message deletion techniques described in *Threat 3*, consumes considerable resources and may not be considered a relevant DoS attack because it is just like a frequency jamming.

Threats 1, *2*, and *3* attack all three types of frames in the Link Layer, possibly breaking data confidentiality and integrity of a WLAN. *Threats 4*, *5*, and *6* defeat mutual authentication; generally they arise from compositions of *Threats 1*, *2*, and *3* on management frames. *Threat 7* interferes with availability, and could result from *Threats 1*, *2*, and *3* on any type of frames. The following sections will analyze the effectiveness of 802.11i for defending against these threats; appropriate suggestions or modifications are proposed if 802.11i cannot eliminate these threats.

2.2 Data Confidentiality and Integrity

IEEE 802.11i defines three data confidentiality protocols: Wired Equivalent Privacy (WEP), Temporal Key Integrity Protocol (TKIP), and Counter-mode/CBC-MAC Protocol (CCMP). The vulnerabilities of WEP and TKIP have been studied extensively [8, 7, 16, 19, 39, 75, 82, 97, 99]. Therefore, this section will focus on analyzing CCMP. Note that a fresh Temporal Key (TK) is assumed to be shared between the peers before executing any data confidentiality protocols.

Unlike the RC4 stream cipher used in WEP and TKIP, CCMP uses the CCM (Counter with CBC-MAC) operation mode [100] of the AES encryption algorithm [78] with a 128-bit key and a 128-bit block size. CCMP combines the counter mode (CTR) for data confidentiality and the Cipher Block Chaining Message Authentication Code (CBC-MAC) for data integrity, using an 8-octet MIC (Message Integrity Code) and a 2-octet Length field. We assume that a 128-bit key is secure against brute-force attacks on AES. With AES, it is possible to use a single 128-bit key to encrypt all packets, eliminating the problems of key scheduling algorithms associated with WEP and TKIP. CCMP also provides MIC protection over both the frame body and nearly the entire header in a MAC frame, which prevents an adversary from exploiting the MAC headers. In addition, CCMP uses a 48-bit Packet Number (PN) to prevent replay attacks and construct a fresh nonce for each packet. The sufficient space of PN eliminates any worry about PN re-usage during an association.

A possible vulnerability might arise from the fact that CCM uses the same key for both confidentiality and integrity. However, CCM appears unproblematic because it guarantees that the space of the CTR never overlaps with the space of the CBC-MAC initialization vector. If AES behaves like a pseudo-random permutation, which is a plausible assumption, and the cipher operates on two separate spaces, which is guaranteed by CCM, then the outputs of the cipher will be independent.

Another possibility is pre-computation attacks. While a 128-bit key is considered secure against brute-force attacks; CCMP uses an incremental PN to construct nonces, and the PN is initialized to one for every fresh TK. This allows a common pre-computation attack. An adversary might compute a table offline for one specific

nonce and 2^{64} possible keys. Then the adversary starts to observe the online messages encrypted with this specific nonce and an unknown key. On average the adversary could find an overlap of keys after observing 2^{64} messages with the specific nonce and different keys, obtaining the TK for that session. This pre-computation attack reduces the key space from 2^{128} to 2^{64}, which is possible to be broken practically for a block cipher. However, since CCMP also includes the source MAC address in constructing the nonce, once the adversary chooses a specific value to build the offline table, the same nonce will only appear for the specific station. Furthermore, since a PN will never repeat for the same TK, the adversary may need to wait for refreshed TKs, which means different sessions, in order to observe more messages with the same nonce. Therefore, the combination of the PN and the MAC address requires that an adversary keep observing 2^{64} different sessions for a specific station in order to break one TK. Additionally, 2^{64} entries may consume significant resources to store and search efficiently. Hence, this is not considered a practical attack.

Although there has been some criticism [88], analysis suggests that CCM provides a level of confidentiality and authenticity comparable to other authenticated encryption modes, such as OCB mode [57]. Hence, it is reasonable to believe that, once CCMP is implemented, an adversary is not able to break the data confidentiality and integrity without the knowledge of the key. Furthermore, an adversary cannot obtain useful information about the key through analyzing the cipher text even if the corresponding plaintext is known.

For completeness, we discuss the threats listed in Section 2.1 in order. With regard to *Threat 1*, eavesdropping and traffic analysis, an adversary may eavesdrop on traffic, but it cannot decrypt the packets because it has no way to discover the TK. Furthermore, since the IP header of the messages is encrypted, the adversary can only obtain limited information through traffic analysis, compared to a higher layer encryption method such as IPsec and SSL. However, the adversary does have some ways to discover useful information, because the MAC header is not encrypted; the packet size and frequency are observable. Fortunately, in most scenarios such information leakage is not considered to be harmful.

Threat 2 on data frames is completely eliminated because a strong MIC prevents

an adversary from inserting a forged data message. Data modifications are similarly prohibited by the MIC. Further, a replayed packet will be discarded silently because the corresponding PN is out of order. For *Threat 3*, the adversary is able to delete a packet in any case; however, this can be handled by the retransmission mechanism or higher layer protocols. On the other hand, the adversary is also able to intercept a packet and forward this packet to the receiver later. The forwarded packet could have been correctly encrypted with a valid MIC; however, the receiver is likely to recognize this as an out-of-order packet and discard it silently.

In summary, against *Threats 1*, *2*, and *3*, CCMP appears to provide satisfactory data confidentiality, integrity, and replay protection for data packets, as intended. However, since management frames and control frames are neither encrypted nor authenticated by the Link Layer encryption algorithm, they are still vulnerable to these threats. In addition, not surprisingly, CCMP requires hardware upgrades and might have some impacts on performance.

2.3 Authentication and Key Management

Prior work has shown that 802.11 entity authentication (Open System Authentication and Shared Key Authentication) are completely insecure [8, 16]. Therefore, 802.11i defines the Robust Security Network Association (RSNA) establishment procedure to provide strong mutual authentications and generate fresh TKs for the data confidentiality protocols. This section will analyze the RSNA handshakes in detail.

2.3.1 RSNA Establishment Procedure

802.11i RSNA establishment procedure consists of 802.1X authentication and key management protocols. Three entities are involved, called the Supplicant (the wireless station), the Authenticator (the Access Point), and the Authentication Server (de facto a RADIUS server [87]). Generally, a successful authentication means that the supplicant and the authenticator verify each other's identity and generate some shared secret for subsequent key derivations. Based on this shared secret, the key

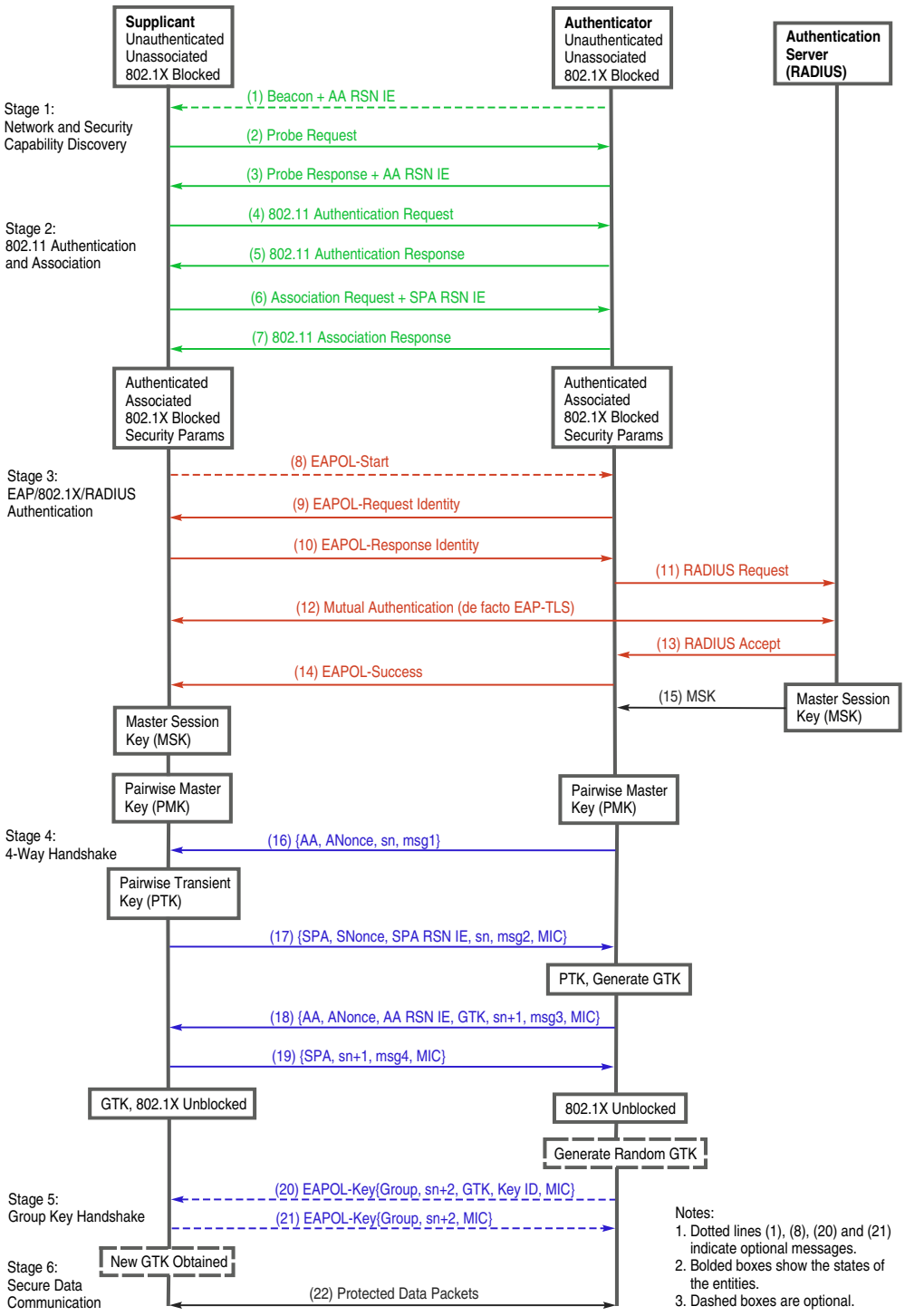

Figure 2.1: RSNA Establishment Procedure

management protocols compute and distribute usable keys for data communication sessions. The authentication server can be implemented either in a single device with the authenticator, or through a separate server, assuming the link between the authentication server and the authenticator is physically secure. The complete handshakes of establishing a RSNA are shown in Figure 2.1. For the purpose of analysis, these steps can be divided into 6 stages as follows.

Stage 1. Network and Security Capability Discovery

This stage consists of messages numbered (1) to (3). The AP (Access Point) either periodically broadcasts its security capabilities, indicated by RSN IE (Robust Security Network Information Element), in a specific channel through the Beacon frame; or responds to a station's Probe Request through a Probe Response frame. A wireless station may discover available access points and corresponding security capabilities by either passively monitoring the Beacon frames or actively probing every channel.

Stage 2. 802.11 Authentication and Association

This stage consists of messages numbered (4) to (7). The station chooses one AP from the list of available APs, and tries to authenticate and associate with that AP. Note that 802.11 Open System Authentication is included only for backward compatibility, and a station should indicate its security capabilities in the Association Request. After this stage, the station and the AP are in authenticated and associated state. However, the authentication achieved so far is weak, and will be supplemented by further steps. At the end of this stage, the 802.1X ports remain blocked and no data packets can be exchanged.

Stage 3. EAP/802.1X/RADIUS Authentication

This stage consists of arrows numbered (8) to (14). The supplicant and the authentication server execute a mutual authentication (de facto EAP-TLS [5]), with the authenticator acting as a relay. After this stage, the supplicant and the authentication server have authenticated each other and generated some common secret, called the Master Session Key (MSK). The supplicant uses the MSK to derive a Pair-wise

Master Key (PMK); The AAA key material on the server side is securely transferred to the authenticator, indicated by message (15), to derive the same PMK in the authenticator. Note that this stage might not be present if the supplicant and the authenticator are configured using a static Pre-Shared Key (PSK) for the PMK, or when a cached PMK is available during a Re-Association.

Stage 4. 4-Way Handshake

This stage consists of messages numbered (16) to (19). Regardless of whether the PMK is derived from Stage 3, configured using a PSK, or reused from a cached PMK, the 4-Way Handshake must be executed for a successful RSNA establishment. The supplicant and authenticator use this handshake to confirm the existence of the PMK, verify the selection of the cipher suite, and derive a fresh Pairwise Transient Key (PTK) for the following data session. Simultaneously, the authenticator might also distribute a Group Transient Key (GTK) in message (18). After this stage, a fresh PTK (and maybe GTK) is shared between the authenticator and the supplicant; the 802.1X ports are unblocked for data packets.

Stage 5. Group Key Handshake

This stage consists of arrows numbered (20) and (21). In the case of multicast applications, the authenticator will generate a fresh Group Transient Key (GTK) and distribute this GTK to the supplicants. These handshakes might not be present if a fresh GTK has been distributed in Stage 4. These handshakes may be repeated multiple times using the same PMK.

Stage 6. Secure Data Communication

This stage is indicated by arrow (22). Using the PTK (or GTK) and the negotiated cipher suite from above handshakes, the supplicant and the authenticator construct secure communication channels to transmit data packets using certain data confidentiality protocol.

With these handshakes, the supplicant and the authenticator can mutually authenticate each other and establish a secure session. Section 2.3.2 analyzes this process in detail. These RSNA handshakes appear to have no significant vulnerabilities; however, they could be broken in case of inappropriate implementations. Furthermore, if Pre-RSNA and RSNA co-exists in a system, an adversary might be able to cancel the entire RSNA security by a security level rollback attack, which is described in Section 2.3.3.

2.3.2 RSNA Security Analysis

Based on the complete RSNA establishment procedure described above, we will analyze the security of 802.11i considering each possible threat separately. Since the management frames are not protected in a WLAN, an adversary is capable of interfering with Stages 1 and 2 of the RSNA establishment. More specifically, Stages 1 and 2 are vulnerable to *Threats 1*, *2*, *3*, and *4*. An adversary can send spoofed security capabilities and topological views of the network to a supplicant on behalf of an authenticator. Once this occurs, the supplicant will be forced to use inappropriate security parameters to communicate with the legitimate authenticator, or associate with a malicious AP. Alternatively, an adversary can also forge Association Requests to the authenticator with possibly weak security capabilities, which might cause problems if no further protections are adopted. Fortunately, these threats are eliminated in Stage 3 if a strong mutual authentication is implemented. The authentication mechanism (e.g., EAP-TLS) should prevent an adversary from forging, modifying, and replaying authentication packets, eliminating *Threats 1*, *2*, and *3*. In addition, since credentials other than MAC addresses must be provided for successful mutual authentication, *Threat 4* is not possible. After the peers have authenticated each other and exchanged some secret after Stage 3, the subsequent handshakes are also resistant to these threats. In the case that Stage 3 is omitted because a PSK or a cached PMK is used, the peers can authenticate each other by verifying possession of the shared key in Stage 4, again preventing *Threats 1*, *2*, *3*, and *4*. In addition, Stage 4 can also verify the security capability negotiations.

Threat 5 may exist even if a strong authentication mechanism is implemented. After a legitimate station has completed a successful authentication, the adversary could disconnect a station by forging Deauthentication or Disassociation messages, and resume the session with the AP on behalf of the legitimate station. There are two possibilities to consider. First, if the session allows the adversary only to accept packets, this appears to be just eavesdropping, which is prevented by data confidentiality mechanisms. Second, if the session requires the adversary for interaction, the adversary will need to obtain the authentication information, such as the PTK, in order to generate acceptable traffic. On the whole, *Threat 5* poses no more danger than eavesdropping and DoS attacks on the station.

Threat 6 is possible in a WLAN if no authentication mechanisms are implemented. An adversary could establish two separate connections to the supplicant and the authenticator to construct a MitM attack [61]. First, the adversary forges the Deauthentication frames to disconnect a station from a legitimate AP. Then, the victim station will automatically probe for a new AP after several failed retries, and eventually associate with the malicious AP, maybe on a different channel. At last, the adversary associates with the legitimate AP on behalf of the legitimate station. However, when 802.11i is implemented with a strong mutual authentication mechanism, like EAP-TLS, the adversary might not be able to authenticate itself to the station or the AP because it does not possess appropriate credentials. Of course the adversary can forward credentials between the AP and the station; but since the authentication packets cannot be modified or replayed, the adversary can only act as a relay, which causes no more damage than eavesdropping. However, if the mutual authentication mechanism is not appropriately implemented [9], the adversary will be able to launch a MitM attack and learn the PMK. Though this vulnerability is considered as a weakness of the specific mutual authentication protocol instead of 802.11i, any implementer of 802.11i should consider this problem carefully.

In summary, if the complete RSNA handshakes are performed, the authentication and key management process appear to be secure. However, since the adversary could interfere with Stage 1 and 2, it might be able to fool the authenticator and the supplicant, and prevent completion of the RSNA; this is described as a Security Level

Rollback Attack in Section 2.3.3. In addition, some implementations might also allow a reflection attack in Stage 4; the details are described in Section 2.3.4. Furthermore, although we assume the link between the authenticator and the authentication server is secure, an adversary may still be able to discover the shared secret in RADIUS by offline dictionary attacks [4]. When a 256-bit PSK is used as a PMK, this PSK might be derived from a passphrase [77], which makes the PSK vulnerable to dictionary attacks. An implementation should carefully choose a good passphrase or directly use a 256-bit random value to eliminate this vulnerability.

2.3.3 Security Level Rollback Attack

When Pre-RSNA and RSNA algorithms are both used in a single WLAN, an adversary can launch a Security Level Rollback Attack, avoiding authentication and disclosing the default keys. Some might argue this is not a real vulnerability, because 802.11i explicitly disallows Pre-RSNA algorithms when RSNA is used. However, 802.11i does define a Transient Security Network (TSN) supporting both Pre-RSNA and RSNA algorithms, and this situation might naturally appear in a WLAN implementation. In general, new WLAN implementations may try to support Pre-RSNA algorithms in order to support migration to RSNA. In other words, a supplicant might enable accesses to both RSNA and Pre-RSNA capable networks to ensure Internet access under mobility; simultaneously, an authenticator might be configured in a similar way to provide services to various supplicants. This hybrid configuration will degrade the security of the entire system to the lowest level.

Figure 2.2 shows an attack to roll back the security level. In this figure, the green lines represent legitimate message exchanges and the red lines indicate messages sent by the adversary. In this attack, the adversary impersonates the authenticator, forging the Beacon or Probe Response frames to the supplicant, and indicating that only Pre-RSNA (WEP) is supported. Alternatively, the adversary can impersonate the supplicant, forging the Association Request frame in a similar way. As a result, the supplicant and the authenticator will establish a Pre-RSNA connection, even though both of them could support RSNA algorithms. Since there is no cipher suite

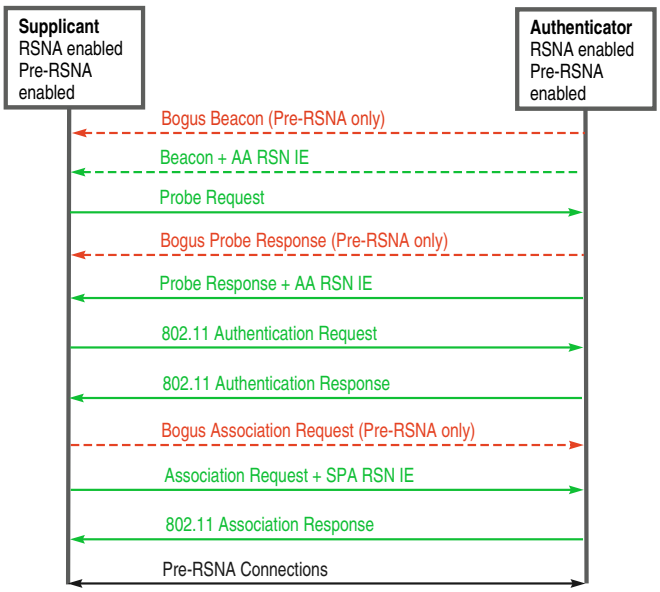

Figure 2.2: Security Level Rollback Attack

verification in Pre-RSNA, the supplicant and the authenticator will not be able to detect the forgery and confirm the cipher suites. Even worse, the adversary is able to disclose the default keys by exploiting the weakness of WEP, which then completely undermines the security. This attack is practically feasible because the adversary could either perform as a MitM or forge the beginning management frames in a timely way, that is, Beacon or Probe Response frame to the supplicant and Association Request frame to the authenticator.

The solution is not complicated. In the simplest approach, both the authenticator and the supplicant could allow only RSNA connections. However, this is only acceptable when security is a strict requirement for the whole system. In most scenarios, TSN might be a better choice to provide services to more supplicants. Therefore, the supplicant and the authenticator could allow both Pre-RSNA and RSNA connections, but deploy appropriate policies on the choice of the security level. Specifically, the supplicant should decide whether it wants a more confidential connection (using RSNA), or it wants more availability of Internet access (using Pre-RSNA). In any

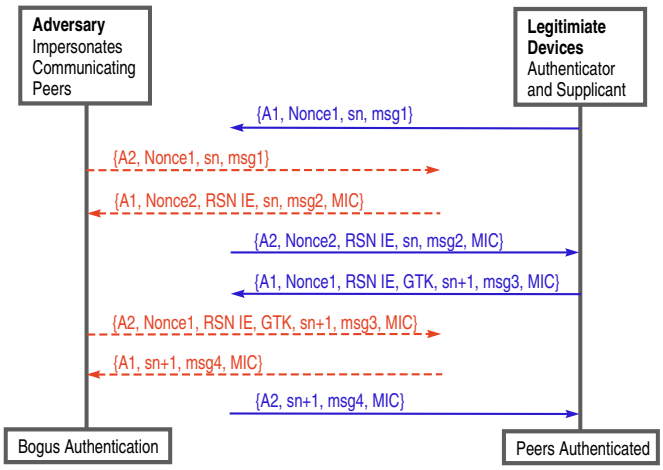

Figure 2.3: Reflection Attack on the 4-Way Handshake

event, the supplicant should have a chance to deny the Pre-RSNA algorithms, prior to initiating a connection, either manually or through some form of policy configuration. The authenticator could limit Pre-RSNA connections to only insensitive data. While this policy might cause some inconvenience, it may be worth the security it provides. It is absolutely unreliable to allow the devices to choose a security level transparently, because the authenticator and supplicant have no knowledge of the authenticity of Stages 1 and 2.

2.3.4 Reflection Attack

In Stage 4, the 4-Way Handshake uses symmetric cryptography to protect the integrity of the messages. Since the authenticator and the supplicant both know the shared PMK, they are the only two parties that are able to calculate correct MICs and compose valid messages. This fact supports authentication. However, if a device is implemented to play the role of both the authenticator and the supplicant under the same PMK, an adversary can launch a common reflection attack to this device, as shown in Figure 2.3. When the device initializes a 4-Way Handshake as an authenticator, the adversary will initialize another 4-Way Handshake, with the same

parameters but with the victim device acting as the intended supplicant. Once the victim device is fooled to compute messages as a supplicant, the adversary could use these messages as valid responses to the 4-Way Handshake initialized by the victim.

Naturally this scenario will not appear in an infrastructure network, because a legitimate device will never implement the role of both the authenticator and the supplicant. However, in ad hoc networks, a possible use of 802.11i could allow each device to serve both roles to distribute their own GTKs. This makes a reflection attack possible. Some might argue that this is not a real vulnerability because the adversary could not decrypt the following data packets without the appropriate key materials (like in Session Hijacking of *Threat 5*). However, it is still valuable to point out the problem because the attack violates mutual authentication; moreover, sometimes the adversary can store the encrypted data for further analysis. In order to eliminate this, the implementer should either limit a device to only one role, or require separate roles to have different PMKs.

2.4 Availability

IEEE 802.11i appears not to emphasize availability as a primary objective, leaving many DoS vulnerabilities even if the strongest data confidentiality and authentication protocols are used. Prior research has found numerous DoS attacks on a WLAN from the Physical Layer to the Application Layer [8, 11, 13, 21, 38, 58, 79, 101]. Compared to DoS attacks in the Physical Layer, DoS vulnerabilities in 802.11i appear to be more severe for several reasons. First, an adversary can launch an 802.11i attack much more easily than a physical layer attack, with only moderate equipment. Second, it is much more difficult for a network administrator to detect and locate these attacks. Furthermore, layer abstraction is a very important concept in networks, requiring each layer to provide independent functionality separately. A more robust 802.11i could help migration to other Physical Layer specifications in the future, which might be secure against DoS attacks. Therefore, it is valuable to strengthen 802.11i against DoS vulnerabilities.

Section 2.4.1 reviews known DoS attacks, analyzes their effects on 802.11i, and

proposes corresponding defenses. Section 2.4.2 discusses the practicality of the DoS attack on the Michael algorithm countermeasure in TKIP. Section 2.4.3 describes the RSN IE Poisoning attack and proposes a feasible repair with minor modifications to the algorithm. Section 2.4.4 explains a DoS attack on Message 1 of the 4-Way Handshake. Section 2.4.5 explores the efficiency of different failure-recovery strategies. Section 2.4.6 proposes an improved version of 802.11i to address all of these vulnerabilities.

2.4.1 Known DoS Attacks and Defenses

Since the management frames and control frames are unprotected in a WLAN, an adversary can easily forge these frames to launch a DoS attack. Among the management frame attacks, the most efficient attack is to forge and repeatedly send Deauthentication or Disassociation frames. In control frames, the most severe problem lies in the virtual carrier-sense mechanism like RTS frame [13, 21]. Unfortunately, these attacks persist even if 802.11i is used to protect the WLAN. It might be possible to adopt a Central Manager to handle these frames specifically and identify the forged frames by their abnormal behavior [32]. However, this requires extra functionality in the authentication server, and the server needs to keep the state of all supplicants. This increases the workload of the server and might be infeasible. Another approach is to respond to Deauthentication and Disassociation frames by restarting a 4-Way Handshake, with the result of the 4-Way Handshake indicating whether these frames are forged [76]. This method could restrict the impact of forged Disassociation and Deauthentication to the 802.11 MAC. However, this does not prevent the attack because periodically forcing a 4-Way Handshake could be an effective DoS attack. Based on these considerations, authenticating management frames appears to be a better approach. This is also described in Section 2.4.6 as an improvement to 802.11i. On the other hand, authenticating control frames might be inefficient and add too much overhead because the control frames could appear frequently; it might be better to handle forged control frames by checking the validity of the virtual carrier-sense according to the knowledge of the specific frames.

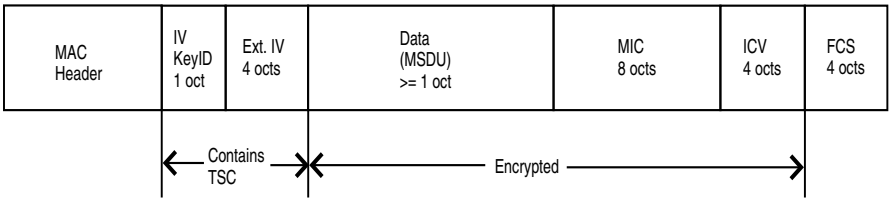

Figure 2.4: TKIP MPDU Format

There are several DoS attacks that exploit the unprotected EAP messages in 802.1X authentication. Specifically, an adversary can forge EAPOL-Start messages repeatedly to prevent the 802.1X authentication from succeeding, forge EAPOL-Success message to maliciously bring up the 802.1X data port in the supplicant without authentication, and forge EAPOL-Failure message and EAPOL-Logoff message to disconnect the supplicant. Fortunately, these vulnerabilities can be eliminated in 802.11i by simply ignoring these messages. This does not affect the functionality and logic of the protocol. The outcome of the subsequent 4-Way Handshake could take the role of EAPOL-Success and EAPOL-Failure to indicate the authentication result; EAPOL-Logoff could be replaced by Deauthentication to disconnect a client; and EAPOL-Start is not necessary for the protocol.

An adversary can also launch a DoS attack on the AP by flooding forged Association Request frames. This will exhaust the EAP Identifier space, which is only 8 bits long (0-255). This vulnerability can be addressed by careful consideration during implementation. Since an EAP Identifier is only required to be unique within a single 802.11 association, it is not necessary for the AP to deny new connection requests when the EAP Identifier space has been exhausted. Particularly, the AP can adopt a separate EAP Identifier counter for each association.

2.4.2 Michael Algorithm Countermeasure

In addition to these known DoS attacks, the countermeasure associated with the Michael algorithm (discussed in the 802.11i standard) is also vulnerable to DoS attacks. As a data confidentiality protocol, TKIP adopts the Michael algorithm to

provide MIC protection for every MSDU (MAC Service Data Unit). The TKIP MPDU (MAC Protocol Data Unit) format is shown in Figure 2.4. The Michael algorithm is designed to provide only 20 bits (or possible slightly more) of security due to the limited computation power in legacy devices. This means it is possible for an adversary to construct a successful forgery after 2^{19} attempts. Therefore, TKIP implements the following countermeasures to limit the rate of the forgery attempts from an adversary. The first Michael MIC failure is logged as a security-relevant matter. Once two failures are detected within 60 seconds, the transmission and reception will cease for 60 seconds. Furthermore, the authenticator could re-key or deauthenticate the supplicant; the supplicant should send out a Michael MIC Failure Report frame and deauthenticate itself afterwards.

As shown in the following calculation, the countermeasure ensures that a successful forgery could occur only every half year, which makes the forgery practically useless. In an 802.11b network, an adversary could send out approximately 2^{12} messages per second. Therefore, the adversary is able to make a successful forgery in about 2 minutes (approximately 2^7 seconds) without implementing the countermeasure. However, if countermeasures are deployed to limit the rate, for example, 2 forgery attempts per minute, the attacker is limited to make one successful forgery every 6 months (approximately 2^{18} minutes). Unfortunately, the countermeasure leaves an obvious DoS vulnerability: an adversary can send out unsuccessful forgery attempts to cause two Michael MIC failures and shutdown a connection. In order to prevent this DoS attack, the protocol checks the FCS (Frame Check Sequence), ICV (Integrity Check Value), TSC (TKIP Sequence Counter) and MIC sequentially. A MIC failure is only logged when the frame has been received with correct FCS, ICV, TSC but an invalid MIC. Checking FCS and ICV can detect packet errors caused by noise, while checking TSC can detect replayed packets. Moreover, if the adversary modifies the TSC, the per-packet key will be modified simultaneously, which causes packet decryption to fail before a log of MIC failure. Hence, checking FCS, ICV, TSC and MIC in a strict order could make DoS attacks more difficult. However, an adversary is still able to launch such an attack through interception. Furthermore, the inappropriate TSC update strategy might make this attack more convenient.

In *Threat 3*, an adversary is able to intercept a message before the receiver hears it. Through this approach, the adversary can obtain a packet with a valid TSC value. Keeping the TSC field unchanged, the adversary is capable of modifying some bits of the packet and updating the corresponding FCS and ICV fields to make them consistent, due to weakness in the ICV algorithm. Then the adversary has obtained the desired packet because the packet can pass the check for FCS, ICV, and TSC, but with an invalid MIC. By sending out this packet, the adversary could force a Michael MIC failure in the receiver side, and eventually launch a DoS attack. Even worse, since 802.11i suggests updating the TSC until an MSDU passes the Michael MIC check, if the receiver implementation resumes communication after 60 seconds without re-keying, the adversary can simply forward one modified message repeatedly because the TSC is still valid after the first failure. On the other hand, if the receiver re-keys the system or deauthenticates, the adversary will have enough time to construct the next modified packet and block the communication. Of course this is not that easy because *Threat 3* requires considerable work from the adversary. However, it is quite practical because the time required for re-keying and re-authentication is sufficient for an adversary to construct the packet.

This attack can be mitigated by careful consideration in the implementation. First, re-keying and deauthentication are not necessary when availability is an objective. The authenticator and the supplicant should just cease for 60 seconds, and then resume communication. Second, the TSC should be updated once a packet passes the check of FCS, ICV, and TSC, even if the Michael MIC failure occurs. Note that in this case the retransmitted packets must use a fresh TSC. These modifications will make the DoS attack more difficult; however, they do not eliminate the vulnerability. Fortunately, this attack disappears with TKIP when CCMP is implemented for data confidentiality.

2.4.3 RSN IE Poisoning

Another possible category of DoS attacks on 802.11i involves the RSN IE (RSN Information Element) verification mechanism. As shown in Figure 2.5, RSN IE contains

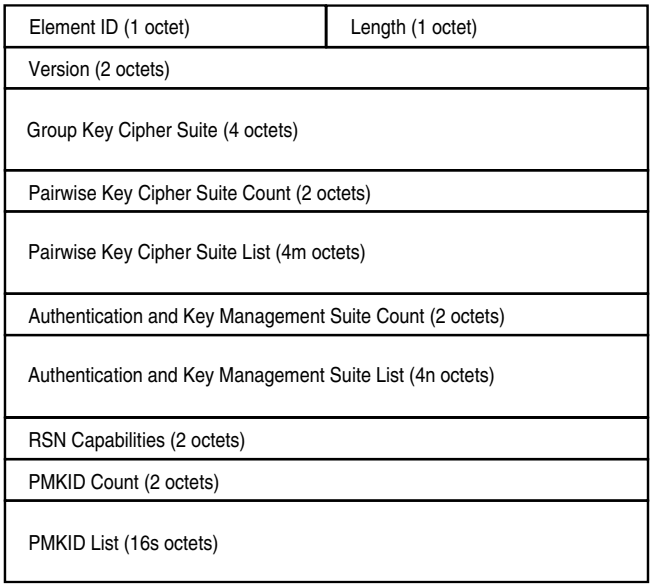

Figure 2.5: RSN Information Element Format

authentication and pairwise key cipher suite selectors, a single group key cipher suite selector, an RSN Capabilities field, the PMKID (Pairwise Master Key Identifier) count, and the PMKID list. The authenticator should insert the supported RSN IEs in the Beacon and Probe Response, and the supplicant should insert its chosen RSN IE in the (Re)Association Request. The authenticator and the supplicant use the negotiated security suites to perform the authentication and key management protocol, and use the negotiated cipher suites to encrypt data communications. In order to confirm the authenticity of the RSN IEs, the supplicant is required to include the same RSN IE in *Message 2* of the 4-Way Handshake as in (Re)Association Request. The authenticator is also required to include the same RSN IE in *Message 3* of the 4-Way Handshake as in the Beacon or Probe Response. After receiving a *Message 2*, the authenticator will bit-wise compare the RSN IE in the message with the one it receives in the (Re)Association Request from the supplicant, in order to confirm that they are exactly the same. The supplicant will bit-wise compare the RSN IE in *Message 3* with the one it receives in Beacon or Probe Response. If the RSN IEs are not

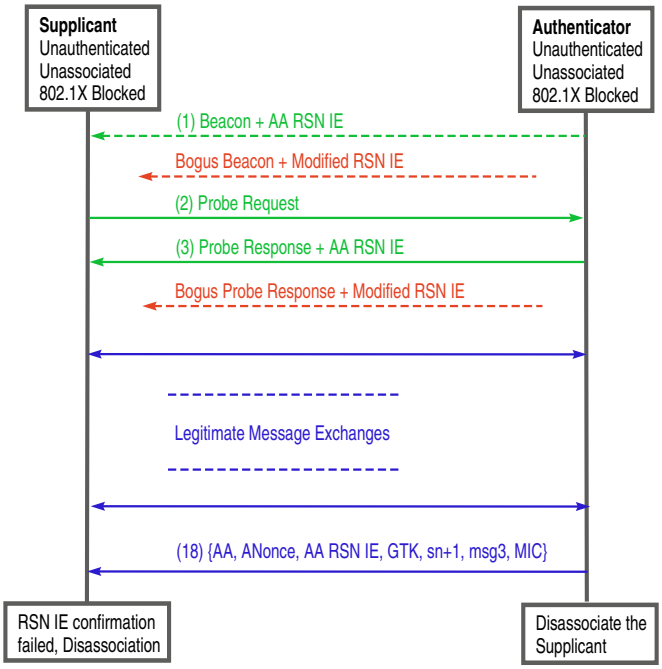

Figure 2.6: RSN IE Poisoning

exactly the same, the supplicant and the authenticator will deauthenticate each other and a security error should be logged. This confirmation process prevents an adversary from tricking the supplicant and the authenticator into using a weaker security scheme by forging the RSN IE negotiations. However, as a result, it is vulnerable to DoS attacks.

In *Message 2* of the 4-Way Handshake, the authenticator verifies the MIC before the RSN IE, which is the correct order; but in *Message 3*, the supplicant checks the RSN IE before the MIC verification, and aborts if the RSN IE is unmatched. Consequently, an adversary can easily modify the RSN IE in *Message 3* to cause the handshake to fail. This vulnerability could almost be considered a typo in the 802.11i documentation. However, even if the check order is correct, there is another fundamental attack to cause the RSN IE confirmation process to fail, which is shown in Figure 2.6.

An adversary can easily eavesdrop on the Beacon frames of a legitimate authenticator, modify several bits in the frame that are "insignificant", where "insignificant" means that, the modification of these bits will not effect the validity of the frame and the selection of the authentication cipher suites. For example, the *Reserved* bits and the *Replay Counter* bits in the RSN Capabilities field are "insignificant". The adversary then broadcasts this forged Beacon to poison the knowledge of RSN IEs in the supplicants. Because this forged Beacon only modifies "insignificant" bits, the supplicant and the authenticator are still able to continue the authentication and key management using the effective security suites. However, the 4-Way Handshake will never succeed because the RSN IE confirmation will fail. Accordingly, when the supplicant uses an active scan instead of a passive one, the adversary can forge a Probe Response with the modified RSN IE, which requires the adversary to interfere with the handshake in a more timely way. The adversary can also forge a (Re)Association Request with modified RSN IE to poison the knowledge of the authenticator, but this approach is less efficient.

Based on analysis above, an adversary can always launch a DoS attack by RSN IE poisoning. This attack is different from the Security Level Rollback Attack, because the adversary does not aim to establish a successful connection; it simply blocks the protocol execution. This attack is considered to be harmful because it is quite easy for an adversary to implement and it will affect all the supplicants simultaneously when the forged Beacon is allowed. Furthermore, because the supplicant and the authenticator are unaware of the RSN IE poisoning, they might continue to exchange considerable number of messages, e.g., messages (2) to (18) in Figure 2.1, until the 4-Way Handshake fails. In other words, the legitimate entities do substantial work, while the adversary is able to successfully interfere with little work. This wastes the resources of the authenticator and the supplicant; moreover, the adversary will have more time to periodically repeat its attacks.

This weakness is exploitable for three reasons. First, the management frames like Beacon, Probe Response, and (Re)Association Request are not protected. Second, there are a number of message exchanges between the RSN IE negotiation and confirmation, which consume resources and leave more time for the adversary. Third, the

bit-wise comparison in the 4-Way Handshake might be unnecessarily strict to confirm RSN IE. The vulnerability may be addressed accordingly. Authenticating the management frames is a good approach. However, in some scenarios it might not be considered acceptable to authenticate the Beacon and Probe Response frames because the authenticator and the supplicant share no secret at the beginning. Therefore, an acceptable approach is to do the RSN IE confirmation as soon as possible to avoid wasting message exchanges and make the attack less disruptive. Considering that the authentication server may know the security parameters that the authenticator supports, it can help to confirm the RSN IE much earlier, such as in the 802.1X authentication. However, this would require an EAP method supporting cipher suite negotiations (e.g., EAP-TLS), and considerable modifications to the existing standard might be necessary.

Alternatively, this attack can be mitigated by loosening the condition of the RSN IE confirmation. In other words, the authenticator and the supplicant can ignore the differences of the "insignificant" bits in the corresponding RSN IEs, while keeping the negotiation secure. Actually in a RSN IE, only the authentication and key management suite selector is essential for the subsequent handshakes, because the authenticator and the supplicant are always able to securely negotiate the encryption cipher suites after they finish the authentication and share some secret. If an adversary does not change the authentication and key management suite selector, the RSN IE could be accepted because the correct authentication has been executed. Afterwards, the authenticator and the supplicant can use the authenticated RSN IE in the 4-Way Handshake for the subsequent data encryptions. On the other hand, if the adversary modifies the authentication and key management suite selector, this can be detected at the beginning of the association. The association fails and the supplicant retries quickly without continuing message exchanges. In the worst case, this modification can be prevented in the 4-Way Handshake.

2.4.4 4-Way Handshake Blocking

The 4-Way Handshake is an essential component of the RSNA establishment. Its purpose is to confirm the possession of the shared PMK (Pairwise Master Key) in the authenticator and the supplicant, and derive a fresh PTK (Pairwise Transient Key) for subsequent data communication. In the handshake, the authenticator and the supplicant generate their own nonces and send them to each other. The PTK is derived from the shared PMK, the nonces, and the MAC address of the peers. *Message 1* and *3* carry the nonce generated by the authenticator; *Message 2* carries the nonce generated by the supplicant, and *Message 4* is an acknowledgment to indicate the handshake is successfully completed. While *Message 2*, *3*, and *4* are authenticated by the fresh PTK, *Message 1* is unprotected. In order to prevent an adversary from affecting the PTK through forging *Message 1*, 802.11i adopts a Temporary PTK (TPTK) to store the newly generated PTK until *Message 3* is verified. However, this approach does not prevent DoS attacks on *Message 1*.

The supplicant must accept all *Message 1*s it receives in order to ensure that the handshake can complete in case of packet loss and retransmission. This allows an adversary to cause PTK inconsistency between the supplicant and the authenticator by sending a forged *Message 1* with a different nonce value between the legitimate *Message 1* and *Message 3*. In order to accommodate the forged *Message 1*s, the supplicant has to store all the responding nonces and the derived PTKs. Only after a *Message 3* with a valid MIC is received, the supplicant can install the corresponding correct PTK for data communications and discard all others. Obviously, an adversary is able to launch a memory DoS attack by sending out numerous forged *Message 1*s, as shown in Figure 2.7. This attack is serious because it is simple for the adversary to perform, and a successful attack will cancel all efforts in the previous authentication process.

There are several approaches to address this vulnerability. First, the supplicant can implement a queue with a random-drop policy. This method helps to mitigate the vulnerability, but does not eliminate it. Second, *Message 1* can be authenticated to defend against this attack, because the authenticator and the supplicant

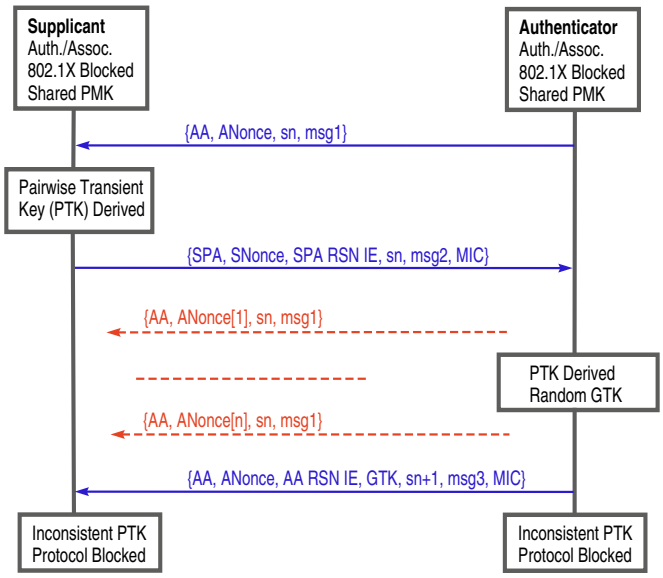

Figure 2.7: 4-Way Handshake Blocking

have already finished the authentication and shared some secret. However, this requires some modifications to the message format; moreover, the authenticator needs to include a monotonically increasing sequence number in each *Message 1* in order to prevent replays. Third, the supplicant can inherently eliminate this attack by re-using the same nonce for all received *Message 1*s until a 4-Way Handshake completes successfully. The supplicant only stores one nonce, calculates a PTK based on this stored nonce and the nonce in the received message, then verify the MIC. This approach only requires minor modifications on the algorithm in the supplicant side; the supplicant need store no more than one nonce, eliminating the possible memory exhaustion. However, the supplicant will consume more computation power because it needs to calculate the same PTK twice for the received *Message 1* and *3*, given that the received nonces and derived PTKs are not stored. The supplicant has to make a decision on the tradeoff between the memory and the CPU consumption. As a combined solution, the supplicant can reuse the same nonce for all *Message 1*s to eliminate the memory DoS vulnerability, and store one entry of the derived PTK to improve the performance. A more detailed discussion appears in Chapter 3 and [40].

2.4.5 Failure Recovery

The 802.11i design decisions appear to emphasize other security objectives over availability. Once security-related events or timeouts occur, the specification always suggests Deauthentication or Disassociation. This mechanism reduces information leakage and prevents further attacks, but it also increases the possibility of potential DoS attacks. Therefore, different failure recovery schemes should be specified when DoS attacks are considered significant. A better failure recovery does not inherently prevent the DoS attacks; however, it will make the protocol more efficient, and cause more difficulties for an adversary trying to launch an attack. For example, in the Group Key Handshake, a timeout will cause the authenticator to disassociate or deauthenticate the supplicant. The supplicant then reassociates with the same AP or scans the channels for another AP, which is quite time consuming. Moreover, since the authenticator can only install the GTK after all the supplicants have done so, the reassociation delay of one supplicant will affect all others. Alternatively, if the supplicant and the authenticator just retry the Group Key Handshake or the 4-Way Handshake, they could resume the connection more quickly. On the other hand, if the Group Key Handshake timeout is due to the unavailability of the supplicant, e.g., the supplicant moving out of the range of the authenticator, retrying the Group Key Handshake or the 4-Way Handshake wastes more time compared to directly disassociating from the current AP and reassociating with another AP. This is a tradeoff the protocol implementer must make according to the networking environment.

This tradeoff could be generalized to any trust relationship discovery process in a malicious environment. Assuming a protocol instance is running between a pair of stations; at some point the instance fails due to some active attacks. If the protocol restarts from the beginning, this protocol will be vulnerable to a so-called “defensive DoS attack”. That means, if an adversary has the capability of causing the protocol to fail at some point, it can launch a DoS attack by periodically doing that. Since each time the legitimate peers need to waste some message exchanges to recover, the adversary will have more time to construct the attack. On the other hand, if the protocol recovers from the nearest point, the adversary might not have sufficient time to construct the “defensive DoS attack”; however, the protocol could be vulnerable

to another type of DoS attack, so-called "captured DoS attack". That means, if the adversary has the capability of propelling the protocol to some point with a legitimate user, it is possible for the adversary to capture the user for more time before the user could find a legitimate peer. Hence, the selection of the recovery point depends on the assumptions of the networking scenario and the capability of the adversary.

Specifically, in 802.11i it might be reasonable to assume that it is difficult to forge an 802.1X authentication. Therefore, once the handshakes between the supplicant and the authenticator have proceeded beyond the 802.1X authentication, the recovery point could be chosen to the nearest point to improve the efficiency of the protocols, because both entities involving in the communications should be legitimate. On the other hand, if a supplicant and an authenticator do not finish an 802.1X authentication, it might be better to restart the protocol from the beginning. Of course, in a high mobility environment, a failure might occur more frequently because one entity is unavailable; thus, retrying the nearest point might waste more time. However, since the channel scanning time is significantly larger than the protocol execution time, retrying the nearest point will not increase the delay too much compared to the total recovery delay.

2.4.6 Improved 802.11i

Based on the above analysis, we propose an improved version of 802.11i to make it more DoS resistant. Note that due to the physical vulnerability of wireless links, DoS attacks always exist through frequency jamming, network jamming, or other exploits. However, we improve the 802.11i protocol to achieve DoS resistance in the Link Layer. A flow chart of the improved 802.11i is shown in Figure 2.8.

1. In order to eliminate the DoS attacks by Association Request flooding, it is better to perform authentication before association; this idea is originally developed in [36]. Specifically, in 802.11i, it is possible to replace the 802.11 entity authentication with 802.1X authentication; a secure association can occur after the 802.1X authentication. This improvement might require modifications to the existing 802.1X implementations; however, the advantages appear worth it.

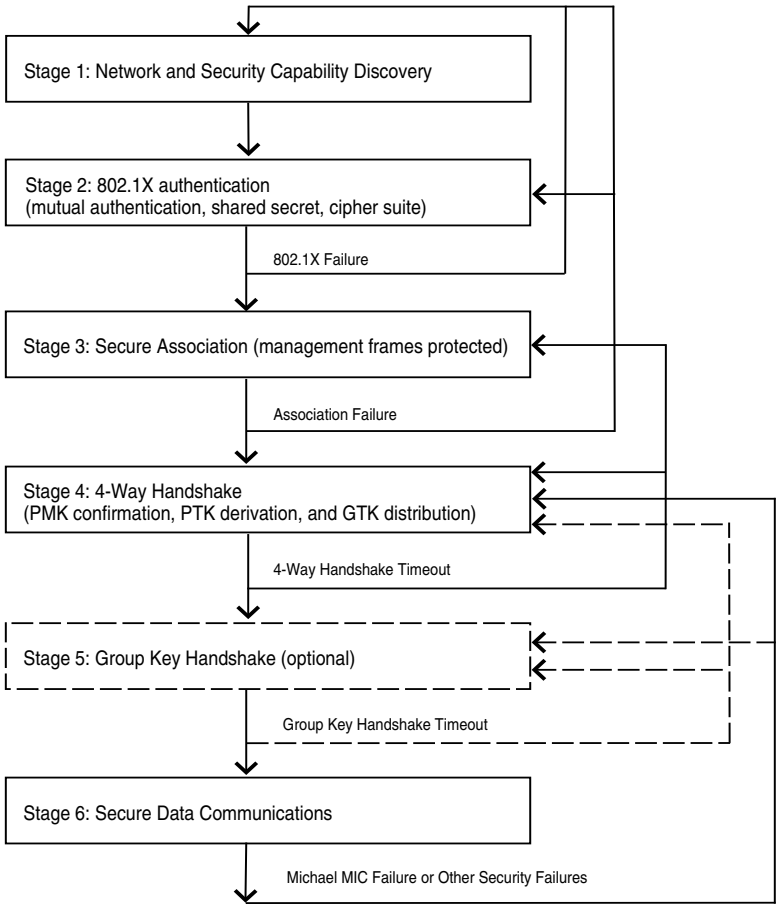

Figure 2.8: A Flow Chart of the Improved 802.11i: The original Stage 2 and 3 in Figure 2.1 are switched here; Possible recovery points for failures at different stages are indicated by arrows; Dashed lines indicate Stage 5 is optional.

2. The authentication and key management suite negotiation should be verified as soon as possible. Otherwise, the subsequent handshakes might waste time because the negotiated security suites could be forged by an adversary. Specifically, when an 802.1X authentication is adopted, the peers can verify their security parameters during the 802.1X authentication; when a PSK or a cached PMK is used, the peers can verify the information through a secure (Re)Association.

3. The management frames should be authenticated to improve security, and some control frames can also be authenticated if necessary. Authenticating these frames should be performed as soon as possible. Once an authentication process is completed successfully, the derived common secret could be used to authenticate the subsequent management frames, especially the (Re)Association Request/Response frame. Through this approach the vulnerabilities of most of the management frames are eliminated, except the Beacon and Probe Request/Response frames, which cannot be authenticated because the common secret is unavailable. Of course, key asynchrony and state abnormality should be carefully considered to avoid blocking the protocols.

4. An appropriate failure-recovery scheme is implemented to improve the efficiency of the overall protocol. In the infrastructure networks with low mobility, assuming the 802.1X provides strong authentication, the protocol will recover from the nearest point if 802.1X has been completed successfully, while the protocol will recover from the beginning if an 802.1X has not yet finished.

Through these improvements, the 802.11i vulnerabilities discussed in this chapter could be eliminated. Note that while the improvements result in DoS robustness, they may require more modifications to the existing implementations.

2.5 Conclusion

This chapter analyzes the IEEE 802.11i protocols for data confidentiality, integrity, mutual authentication, and availability. Under the threats we consider, 802.11i appears to provide effective data confidentiality and integrity when CCMP is used. This

requires a legacy WEP user to upgrade the hardware. Furthermore, 802.11i adopts a RSNA establishment procedure for mutual authentication and key management, which appears to be satisfactorily secure. However, several vulnerabilities might arise in a real implementation. If the mutual authentication mechanism is not implemented appropriately, there might be a Man-in-the-Middle attack that reveals the shared secret. If a passphrase is used to generate a 256-bit PSK, an adversary might be able to find the passphrase through dictionary attacks. An adversary is also able to discover the shared RADIUS secret through dictionary attacks. Furthermore, if Pre-RSNA and RSNA algorithms are implemented in a system simultaneously without careful considerations, an adversary is able to perform a Security Level Rollback Attack to force the communicating peers to use WEP, which is completely insecure. Moreover, if a wireless device is implemented to play the role of both the authenticator and the supplicant, an adversary can construct a reflection attack on the 4-Way Handshake. This scenario naturally appears in ad hoc networks.

Availability is another important security property in wireless networks. Since availability is not the primary design goal, 802.11i appears vulnerable to DoS attacks even if RSNA is implemented. We review the known DoS attacks and propose solutions appropriate to 802.11i. It appears that a better way to eliminate management frame vulnerabilities is to authenticate them. Furthermore, we find and analyze some new DoS attacks arising with 802.11i. First, we analyze the practicality of the DoS attack on the Michael algorithm countermeasures. Here, eliminating re-keying and updating the TSC carefully appear to provide significant improvements. Second, we describe a new DoS attack through RSN IE poisoning. Several repairs are discussed; relaxing the condition for RSN IE verification seems the preferred approach because it only requires minor modifications to the algorithm. Third, a DoS attack on the unprotected *Message 1* of the 4-Way Handshake is described and the corresponding defenses are proposed. Fourth, tradeoffs in the failure-recovery strategy are discussed and an efficient failure recovery for 802.11i is proposed, based on the characteristics of wireless networks. Finally, we integrate all the improvements to construct a DoS resistant variant of the 802.11i protocols.

Chapter 3

Finite-State Verification

In this chapter we showed the procedure for the finite-state verification of security protocols using Murϕ [30], and elaborated the details of the Denial of Service (DoS) attack on the 4-Way Handshake protocol in 802.11i, which have been briefly discussed in Section 2.4.4. We found a significant and unnecessary DoS attack - the 4-Way Blocking attack, and investigated several possible repairs. We provided these basic attacks and repairs to the IEEE 802.11 TGi; the third repair in Section 3.4.3 was adopted. Note that there exist other possible DoS attacks against different layers of 802.11 networks, e.g., Physical Layer [11], MAC Layer [13] or upper layers [8, 79]. However, in this chapter, we focus on analyzing the DoS attacks in the MAC Layer, particularly in the 4-Way Handshake; our repairs do not aim to prevent DoS attacks in other layers.

Through our formal verification process, we also identified the functionality of each field in the handshake messages, achieved a simplified protocol that has as strong an authentication as the original one under our model, and identified some fields in the original protocol as potentially redundant. Our analysis results support and clarify passages in the IEEE 802.11i documentation, providing insights that are useful for understanding and/or implementing the protocol.

This chapter is organized as follows. Section 3.1 describes the Murϕ model checker. Section 3.2 explains the finite-state verification of the 4-Way Handshake. Section 3.3 analyzes the DoS attack in detail and discusses the practicality of the attack, using an

11Mbps 802.11b network as an example. Section 3.4 proposes several possible repairs and evaluates their effectiveness. Section 3.5 concludes the chapter.

3.1 Finite-State Verification

Finite-State Verification, commonly called "model checking", is a very useful methodology to analyze security protocols. The basic idea is to define a Finite-State Machine (FSM) following the protocol and specify corresponding security properties, then exhaustively search all execution sequences and check whether the security properties hold in each state. Note that the finite-state analysis can NOT proof the security of a protocol, because the FSM model may simplify certain details in the protocol and is inherently limited to a bounded number of participants and sessions. During our study we use Murϕ for the verification of the 4-Way Handshake, which is described in detail in the following sections.

3.1.1 The Murϕ Model Checker

Murϕ [30] is a verification tool that will exhaustively search all execution sequences of a nondeterministic finite-state system. It has been successfully applied to several industrial protocols, especially in the domains of multiprocessor cache coherence protocols and multiprocessor memory models [31, 96, 103]. Mitchell et. al. [72] also applied this tool to the verification of small security protocols, such as the Needham-Schroeder public key protocol, the Kerberos protocol and the TMN cellular telephone protocol. Subsequently, Mitchell et. al. [73] adopted a "rational reconstruction" methodology to analyze the SSL 3.0 handshake protocol using Murϕ. Here we overviewed the Murϕ model checker and referred the reader to [72, 73] for more details.

In order to verify security protocols using Murϕ, we need to formulate a protocol model, add an attacker to the system, state the desired security properties, and run the protocol for some specific choice of system size parameters. The Murϕ model checker uses explicit state enumeration (either breadth-first or depth-first search) to automatically check whether all reachable states of the model satisfy the given

properties. If any specified properties are violated, a trace of messages will be output, identifying the steps involved in this successful attack. Note that the reached states are stored in a hash table to avoid redundant work when a state is revisited; the memory available for this hash table determines the largest tractable problem.

The Murϕ language for the protocol description is a simple high-level language for describing nondeterministic Finite-State Machines. The *state* of the model consists of the values of all global variables, which are initialized in the *startstate* statements, and updated by the transition from one state to another according to *rules*. Each *rule* has a boolean condition and an action which may be executed if the condition is true. The action is a program segment that is executed atomically and typically changes global variables to yield a new state. Most Murϕ models are nondeterministic since *states* typically allow execution of more than one action.

Murϕ has no explicit notion of processes. Nevertheless a process can be implicitly modeled by a set of related *rules*. The parallel composition of two processes in Murϕ is simply done by using the union of the *rules* of the two processes. Each process can take any number of steps (actions) between the steps of itself and the other. The resulting computational model is that of asynchronous, interleaving concurrency. Parallel processes communicate via shared variables; there are no special language constructs for communication.

The Murϕ language supports scalable models, which means that we are able to change the size of the model by simply changing constant declarations. When developing protocols, we typically starts with a small protocol configuration. Once this configuration is correct, we gradually increases the protocol size to the largest value that still allows verification to complete. In many cases, an error in the general (possibly infinite state) protocol will also show up in a down-scaled (finite state) version of the protocol. Murϕ can only guarantee correctness of the downscaled version of the protocol, but not correctness of the general protocol.

The desired properties of a protocol can be specified in Murϕ by *invariants*, which are boolean conditions that have to be true in every reachable *state*. If a *state* is reached in which some *invariant* is violated, Murϕ prints an error trace - a sequence of *states* from the start state to the state exhibiting the problem. The error trace

should be analyzed manually to determine whether this is a successful attack caused by the protocol vulnerabilities.

3.1.2 The Verification Procedure

Here we briefly listed the steps of analyzing security protocols using Murϕ model checker.

1. Formulate the protocol. In this step we abstract the protocol and describe it in the Murϕ language, including defining an explicit message format and global variables, identifying the key steps and primitives, and instantializing the rules. Obviously the formulation depends on the protocol specifications and will be different for various protocols.

2. Specify the attacker model. While the honest participants act following the protocol, an adversary will do whatever it can to break the system. For example, the adversary can eavesdrop and replay every message, intercept and delete messages, and generate messages using any combination of initial knowledge about the system or parts of overheard messages. These describe the capability of the adversary, which are generally similar for all the protocols.

3. State the desired correctness condition. The conditions specify the security properties under considerations, which must hold in all reachable states. Any violation of the conditions in any state could be a potential attack.

4. Run the protocol. The protocol model should be executed by specifying a suitable number of participants. Note that the states of the system increase exponentially along with the system size; therefore, a general idea is to start from a small system size, then increase step by step. If a relatively large system size is required, which may cause the system to run for many hours or terminate inconclusively by exceeding available memory, specific techniques should be considered to decrease the memory occupation.

5. Analyze results, improve model and repeat. When the protocol model is executed, it is very possible that some error trace will be output. We should

manually analyze the result to distinguish whether this is caused by our abstraction or it is a real vulnerability of the protocol. If it is caused by our neglect of specific details of the protocol, we should improve the Murϕ model accordingly and repeat the above steps to find a "real" attack.

3.1.3 Limitations

The Murϕ verification is very useful for protocol analysis; however, it can not prove the correctness due to the physical limitation of the number of states. Furthermore, the intruder model is limited in its capabilities and does not have all the power that a real-life intruder may have. In the following, we discuss examples of these limitations.

First, our intruder model ignores both computational and number-theoretic properties of cryptographic functions. As a result, it cannot perform any cryptanalysis whatsoever. If it has the proper key, it can read an encrypted message (or forge a signature). Otherwise, the only action it can perform is to store the message for a later replay. We do not model any cryptographic attacks such as brute-force key search (with a related notion of computational time required to attack the encryption) or attacks relying on the mathematical properties of cryptographic functions.

Second, although Murϕ defines a nondeterministic finite-state machine, it has no notion of probability. Therefore, we do not model "propagation" of attack probabilities through our finite-state system (e.g, how the probabilities of breaking the encryption, forging the signature, etc. accumulate as the protocol progresses). We also ignore that the intruder may learn some probabilistic information about the participants' keys by observing multiple runs of the protocol.

Third, we treat the cryptographic functionalities (e.g., the encryption, secret keys, fresh nonces, etc.) as atomic entities in our model. The intruder cannot break such data into separate bits and cannot perform an attack to disclose part of the confidential information. Furthermore, we do not consider the detailed implementation of the cryptographic functions.

There are still other limitations with our model, e.g., we cannot model the timing analysis. Although these limitations exist, we still consider the finite-state verification

very valuable since it can help us find subtle problems in the protocols; furthermore, our analysis shows the value of this methodology.

3.2 Verification of the 4-Way Handshake

3.2.1 The 4-Way Handshake

As we have described in Chapter 2, regardless of whether the PMK is derived from the EAP/802.1X/RADIUS handshakes, based on a PSK, or reused from a cached PMK, the authenticator may begin a 4-Way Handshake by itself or upon request from the supplicant in order to successfully establish a RSNA. This key management protocol confirms the existence of the PMK, the liveness of the peers, and the selection of the cipher suite. The protocol also generates a fresh Pairwise Transient Key (PTK) for each subsequent session, synchronizes the installation of PTKs into the MAC, and in the case of multicast applications transfers the Group Transient Key (GTK) from the authenticator to the supplicants. After a successful 4-Way Handshake, a secure communication channel between the authenticator and the supplicant can be constructed for subsequent data transmissions, based on the shared PTK and/or GTK. The 4-Way Handshake may be repeated using the same PMK to refresh the PTKs.

The detailed message exchanges of the 4-Way Handshake are shown, at an abstract level, in Figure 3.1. *SUPP* represents the Supplicant and *AUTH* represents the Authenticator; *SPA* and *AA*, *SNonce* and *ANonce*, represent the MAC address and nonces of the supplicant and authenticator, respectively; *sn* is the sequence number; *msg1, 2, 3, 4* are indicators of different message types; $\text{MIC}_{PTK}()$ represents the Message Integrity Code (MIC) calculated for the contents inside the bracket with the fresh PTK. While MAC is commonly used in cryptography to refer to a Message Authentication Code, the term MIC is used instead in connection with 802.11i because MAC has another standard meaning, Medium Access Control, in networking.

Message 1: AUTH → SUPP
AA, ANonce, sn, "msg1"
Message 2: SUPP → AUTH
SPA, SNonce, sn, "msg2", MIC_{PTK}(SNonce, sn, "msg2")
Message 3: AUTH → SUPP
AA, ANonce, sn+1, "msg3", MIC_{PTK}(ANonce, sn+1, "msg3")
Message 4: SUPP → AUTH
SPA, sn+1, "msg4", MIC_{PTK}(sn+1, "msg4")

Figure 3.1: The Idealized 4-Way Handshake Protocol

The fresh PTK is derived from the shared PMK through a Pseudo Random Function with output length X (PRF-X), say, PTK = PRF-X(PMK, "Pairwise key expansion" ‖ Min{AA, SPA} ‖ Max{AA, SPA} ‖ Min{ANonce, SNonce} ‖ Max{ANonce, SNonce}), and divided into KCK (Key Confirmation Key), KEK (Key Encryption Key) and TK (Temporary Key). Note that the MIC is actually calculated with KCK, which is only part of PTK. However, we do not distinguish them here because this appears to be unrelated to the authentication process.

We ignore some fields of the messages in this abstracted version of the message exchange, primarily because such fields are not essential for authentication, although they could improve the security in some sense. First, in the original protocol, a PMKID is included in *Message 1* to indicate the corresponding PMK used in the handshake. This PMKID field can improve the security until it is transmitted in the wireless link for the first time; the details will be discussed in Section 3.3.3. Second, the RSN IE (Information Element) fields are included in *Message 2* and *Message 3* to negotiate the cipher suite and avoid a version rollback attack. Third, an encrypted GTK is sent together in *Message 3* in the case of multicast applications.

When the 4-Way Handshake protocol runs as intended, a communicating authenticator - supplicant pair execute exactly one run of the protocol and share one valid PTK after the handshake. The authenticator can refresh the PTK either periodically or upon the request from the supplicant by running another 4-Way Handshake with the same PMK.

The authenticator and the supplicant will silently discard any received message that has an unexpected sequence number or an invalid MIC. When the supplicant does not receive *Message 1* within the expected time interval after a successful 802.1X authentication, it will disassociate, de-authenticate and try the same or another authenticator again. Note that the supplicant does not use any other timeout during the 4-Way Handshake. On the other hand, the authenticator will timeout and retry the message if it does not receive the expected reply within the configured time intervals. Furthermore, the authenticator will de-authenticate the supplicant if it does not receive a valid response after several retries.

While these operations sound reasonable, packet loss must be taken into account, as well as the possibility of malicious messages from an attacker. While the authenticator can initialize only one handshake instance and accept only the expected response, the supplicant must accept all messages in order to allow the handshake to proceed. Hence, an attacker can easily interfere with the handshake protocol by inserting a forged *Message 1*. This leads to more severe vulnerabilities than might be expected. We find this vulnerability by Murϕ modeling, with details discussed in the following sections.

3.2.2 The Protocol Model

In our Murϕ model, we consider the idealized 4-Way Handshake protocol in Figure 3.1. Because the PMK is assumed to be secure, the most important properties here are the PTK consistency and freshness. For simplicity, we assume that the cryptographic functions cannot be broken unless the key is disclosed.

The authenticator and the supplicant are programmed to follow the protocol. Each pair of authenticator and supplicant shares a PMK and tries to execute a given number of 4-Way Handshake sessions sequentially. The attacker is able to masquerade as any participant in the system by forging the MAC address. However, the attacker is assumed not to know the shared PMK of any pair of honest participants. The attacker can also eavesdrop on every message, remember nonces and MICs of each message, insert forged messages, and replay stored messages. Furthermore, the

attacker can compose *Message 1* from stored nonces, and respond to every message with an arbitrary combination of known nonces and MICs. It is not obviously easy for the attacker to intercept and block delivery of a message transmitted over a wireless link; however, we assume the attacker is capable of doing so because any message might be lost in a wireless environment, and this has the same effect on the protocol.

The system size parameter indicates the number of authenticators, supplicants and attackers in the system and the number of sequential sessions each pair of participants can execute. We executed the Murϕ model with different fields enabled in the message format, identified the functionality of each field, and achieved a simplified message format, keeping the same properties as the original one. The details will be discussed in Section 3.2.3. Furthermore, we found a DoS attack using *Message 1* that will block the protocol very easily. Note that although our verification process often reveals attacks, failure to find attacks does not imply that the protocol is completely secure, because the Murϕ model may simplify certain details and is inherently limited to the configurations of the small number of entities and the capabilities of the attackers.

3.2.3 The Protocol Clarifications

Starting from the simplest message format (only nonces included), we increase the complexity of the messages until the complete protocol in Figure 3.1 is reached. For each different message format, the Murϕ model checks all possible executions and finds the attacks that arise due to the absence of certain fields. Through this approach, we identify the functionality of each field in the message. We will not describe this process in detail. Instead, we summarize the outcome for specific fields, in some cases verifying the claims in the documentation of the standard, and in other cases revealing ineffective or redundant mechanisms.

First, the message flag, which is a combination of the Key ACK, Key MIC, and Secure bits in the Key Information field, is necessary and should be protected by the MIC field in the message. This flag makes *Message 1*, *2*, *3*, *4* distinguishable; otherwise, the attacker can easily use MICs in *Message 2* and *Message 3* to forge a

valid *Message 4*, using *Message 2* to forge a valid *Message 3* or vice versa. Furthermore, the authenticator should only generate messages of type 1 and 3; the supplicant should only generate messages of type 2 and 4.

Second, nonces are used to make every message fresh and derive the fresh PTK. These should be generated in an unpredictable and globally unique way. Otherwise, the protocol might be vulnerable to replay attacks or pre-computation attacks. Fortunately, the proposed nonce generation algorithm in the standard appears to satisfy these requirements with high probability.

Third, the sequence number does not appear to be necessary for any security objectives in the 4-Way Handshake. Replay attacks are prevented by the freshness of nonces and PTKs. Furthermore, the sequence number does not provide any performance improvement because eventually the MIC field must be checked if the attacker modifies the sequence number to a valid value. Including the sequence number will provide minor performance improvement only when the attacker blindly replays messages without modifying the sequence number, or when messages arrive out of order. Hence, we consider this field to be largely redundant.

Fourth, the MAC addresses of the authenticator and the supplicant do not appear to be necessary for the authentication process. In particular, it may not be necessary to include these addresses in the PTK derivation. From the documentation, the MAC addresses are used to bind the PTK to the peers. However, by establishing a PMK successfully, the shared PMK has already bound all the following keys with the peers. If the PMK is based on a PSK shared by a group of users, the fresh nonces will bind the PTK to the peers. Including MAC addresses in the PTK derivation does not add any more security to the keys cryptographically.

Based on these clarifications, we achieve a simplified protocol with a simpler message format than the one in Figure 3.1. Under our Murϕ model, the protocol shown in Figure 3.2 has the same authentication properties. Here the PTK is derived without AA and SPA, say, PTK = PRF-X(PMK, "Pairwise key expansion" $\|$ Min{ANonce, SNonce} $\|$ Max{ANonce, SNonce}).

Message 1: AUTH → SUPP
ANonce, "msg1"
Message 2: SUPP → AUTH
SNonce, "msg2", MIC_{PTK}(SNonce, "msg2")
Message 3: AUTH → SUPP
ANonce, "msg3", MIC_{PTK}(ANonce, "msg3")
Message 4: SUPP → AUTH
"msg4", MIC_{PTK}("msg4")

Figure 3.2: The Simplified 4-Way Handshake Protocol

3.3 DoS Attack

In addition to verifying the functionality of each field in the message format, our Murϕ model finds an attack on *Message 1*, easily causing PTK inconsistency between the authenticator and the supplicant. In this situation, protocol execution will be blocked, eventually leading to an authenticator timeout and a subsequent de-authentication of the supplicant. To avoid such an attack, the supplicant needs to keep all the received nonces and the corresponding PTKs, which ultimately leads to a DoS attack. Section 3.3.1 describes the attack in detail; Section 3.3.2 explains the inherent cause of the attack; Section 3.3.3 analyzes some limitations of the attack due to detailed implementations, and Section 3.3.4 illustrates the practicality of the attack.

3.3.1 The DoS attack

Because the attacker is capable of impersonating the authenticator, composing a *Message 1*, and sending to the supplicant, there is a simple one-message attack that causes PTK inconsistency, as shown in Figure 3.3. The attacker sends a forged *Message 1* to the supplicant after *Message 2* of the 4-Way Handshake. The supplicant will calculate a new PTK corresponding to the nonces for the newly received *Message 1*, causing the subsequent handshakes to be blocked because this PTK is different from the one in the authenticator. The attacker can determine the appropriate time to send out *Message 1* by monitoring the network traffic or just flooding *Message 1*

Message 1: AUTH → SUPP
AA, ANonce, sn, "msg1"
Message 2: SUPP → AUTH
SPA, SNonce, sn, "msg2", MIC_{PTK}(SNonce, sn, "msg2")
Message 1': Attacker → SUPP
AA, ANonce', sn, "msg1"
[*Supplicant generates SNonce', derives a new PTK' from SNonce' and ANonce'*]
Message 3: AUTH → SUPP
AA, ANonce, sn+1, "msg3", MIC_{PTK}(ANonce, sn+1, "msg3")
[*PTK and PTK' not consistent, MIC not verified, Protocol blocked*]

Figure 3.3: The one-message attack on the 4-Way Handshake protocol

with some modest frequency.

This attack arises from the vulnerability of *Message 1*. In the 802.11i documentation [50] the designer seems to be aware of possible problems in *Message 1* and proposes the following solution to defend against the attacks. The supplicant stores both a Temporary PTK (TPTK) and a PTK, and updates TPTK upon receiving *Message 1*, updating PTK only upon receiving *Message 3* with a valid MIC. In this solution the attacker cannot drive the supplicant to change its shared PTK because it is not feasible to forge *Message 3*. However, this approach works only when different handshake instances (one between the supplicant and the authenticator, others between the same supplicant and the attacker) are executed sequentially; that is, the forged *Message 1* does not intervene between the legitimate *Message 1* and *Message 3* of the 4-Way Handshake. Obviously this will not prevent the problem shown in Figure 3.3 because the supplicant still cannot correctly verify the MIC in *Message 3* from the authenticator.

We can modify this approach slightly to store two possible keys in TPTK and PTK and try verifying the MIC in *Message 3* with either TPTK or PTK. This modification solves the problem in Figure 3.3; however, the attacker can still cause problems for the supplicant side by sending out more forged messages with different nonces rather than only one. Therefore, in order to assure the handshake is non-blocking with the legitimate authenticator, the supplicant must use sufficient memory

to store all the received nonces and the derived PTKs, until it finishes a handshake and obtains a legitimate PTK. Once the supplicant receives *Message 3*, it can use the PTK corresponding to the nonce in the message to verify the MIC. The derivation of PTKs might not lead to a CPU exhaustion attack because PTK calculations are not computationally expensive. However, a memory exhaustion attack always exists because the number of *Message 1*s can theoretically be unbounded. Though this memory exhaustion attack occurs on the supplicant side, which is not as severe as if it were the server, this is still a problem because it is quite easy for the attacker to forge and flood *Message 1*s.

3.3.2 Parallel Instances

Some may discount this DoS vulnerability as arising from insufficient modelling of the characteristics of wireless networks. Specifically, we have assumed that an attacker may intercept messages and possibly interfere with the delivery of messages. The following arguments show that the vulnerability arises instead from the need to engage in the parallel execution of multiple instances of the handshake protocol.

It is feasible for the authenticator (initiator) to have at most one active handshake in progress with each supplicant. The authenticator expects a correct response for every message it sends out. It can discard an unexpected response and retry the previous message or terminate the handshake if the expected response is not received during a given time interval and certain number of retries.

However, the supplicant (responder) cannot use a similar strategy. More specifically, if the supplicant is configured to be stateful and expects some specific message reply, packet loss or malicious messages from an attacker can cause deadlock and block the protocol. The following arguments are intended to clarify this statement. Assume that the supplicant discards unexpected messages in the intermediate stage of a handshake execution. Consider the case in which the supplicant accepts *Message 1* and sends out *Message 2*, but this message is lost. The authenticator will never get the expected response (*Message 2*); thus, it retries *Message 1* after a timeout. However, the supplicant will discard this retried *Message 1* because it is expecting

a *Message 3*. On the other hand, an attacker can simply initialize a handshake by sending a forged *Message 1* to cause the supplicant to be blocked for the legitimate *Message 1* from the authenticator. Therefore, in the intermediate stage of a handshake, the supplicant must allow any *Message 1* to ensure the protocol to proceed.

The arguments above show that the supplicant must allow multiple handshake instances to run in parallel. In other words, the supplicant should allow *Message 1* at any stage. This makes the one-message attack and DoS attack unavoidable. However, some fields and mechanisms we omitted when abstracting the protocols can defend against the attack in some sense, although they cannot inherently eliminate the attack. The following section will analyze two limitations from the PMKID and the Link Layer Data Encryption.

3.3.3 Limitations

Our basic analysis is based on the idealized handshake protocol shown in Figure 3.1. In our Murϕ model we have not considered the effect of including PMKID in *Message 1* or the involvement of Link Layer Data Encryption for subsequent handshakes.

A PMKID is included in *Message 1* and transmitted in clear text at the beginning of the 4-Way Handshake. It is calculated as PMKID = HMAC-SHA1-128(PMK, "PMK Name" ‖ AA ‖ SPA). With the assumption that PMK is secure, this PMKID is not disclosed to any attacker until the first time it is sent through vulnerable wireless links. This limits the attacker to a DoS attack only after the PMKID is seen in the link. When a PSK is configured for PMK, the attacker may learn the PMKID easily because the PMK, thus the PMKID, is unchanged for a substantial period of time. During a re-authentication process, the supplicant tries to use the cached PMK to communicate with the authenticator; it is possible for the attacker to know the corresponding PMKID earlier if the attacker keeps monitoring the network for some time (The same PMKID might be transmitted in the previous *Message 1*; at least the supplicant might include it in the re-association request message). In both cases, it is easy for the attacker to construct a forged *Message 1*. However, when the 802.1X authentication is used to establish a PMK dynamically, the PMKID will be different

for every session; hence, the attacker cannot know the PMKID until it sees *Message 1* from the authenticator. As a summary, including PMKID in *Message 1* makes the attack more difficult, but it does not eliminate the attack. Instead of blindly flooding *Message 1*, the attacker has to read *Message 1* from the authenticator first, forge its own *Message 1* with the PMKID, and flood the messages.

When sequential 4-Way Handshakes occur under the same PMK, with the exception of the first handshake, all the following sessions are protected by the Link Layer Data Encryption. This mechanism can substantially improve the security of the protocol. All the handshake messages are transmitted in an EAPOL-Key format, which are encapsulated into data frames. Once the supplicant and the authenticator have some shared PTK, the following data frames will be protected by that PTK through encryption and authentication code. With the reasonable assumption that data encryption and MIC computation are cryptographically perfect, and replays can be detected, the attacker will not be able to intercept the transmissions. This mechanism ensures that the subsequent handshake sessions (with the same PMK) are secure except the first one. The attacker needs to catch up with the first 4-Way Handshake session and construct the attack. Obviously, it causes more difficulties for the attacker.

As a result, the Link Layer Data Encryption will limit the attacks to occur only before the first PTK is established, that is the first 4-Way Handshake protocol instance. Furthermore, when 802.1X is used to set up a PMK, the PMKID included in *Message 1* can limit the attacks to occur only after the first legitimate *Message 1* is seen in the wireless link. Combinations of the Data Encryption and the PMKID mitigate the vulnerability to only a limited duration; thus, the attacker has to interfere with the protocol in a more timely way. However, these mechanisms do not eliminate the inherent vulnerability of the protocol; the attack is still possible.

3.3.4 Practicality

When a PSK, or cached PMK, is configured to a current PMK, the attacker can interfere with the 4-Way Handshake either before *Message 1* or after *Message 1*

because it knows the corresponding PMKID from previous eavesdropping. Therefore, the attacker may just send out a forged *Message 1* periodically, the attack succeeding if one of the forged *Message 1*s falls between the legitimate *Message 1* and *Message 3*. Even when 802.1X authentication is used to establish the PMK dynamically, the attacker can still construct the attack after seeing the legitimate *Message 1* from the authenticator. The attacker can succeed with a memory DoS attack by increasing the frequency of sending forged *Message 1*s. In any case, this is different from a trivial network jamming or frequency jamming, and it is hard for the administrator to eliminate the attack because the attacker sends regular messages in a regular way. The following calculations indicate why this attack is practical and how it is different from a network jamming.

Assume that a basic *Message 1* is sent (only PMKID included in the Key Data field of the frame) through 802.11b networks [53]. A MSDU, consisting of 30 bytes MAC header, 6 bytes EAPOL header, 117 bytes EAPOL-Key frame, and 4 bytes checksum are given to the Physical Layer to transmit. Consider the short PLCP frame and an 11Mbps data rate, the DSSS preamble and header are transmitted in 96 μs, the data are transmitted in 114 μs. Even more count in DIFS (50μs), SIFS (10μs) and ACK (96μs + 10μs), *Message 1* is sent and ACKed in 376 μs. Assume that a timeout in the authenticator is set to the default value (100 ms). If the attacker has a full control on the Network Interface Card (NIC), and no random backoff time is inserted between two consecutive messages, a total of 265 *Message 1*s can be sent. Even if the random backoff time is included, on average 310 μs, it is still possible for the attacker to send about 145 messages. This number may be much larger if the timeout is set to a larger value, if we take into account the multiple retry times, and if the network accommodates a higher data rate like 54Mbps in 802.11a/g networks. Among the possible number of *Message 1*s, the attacker only needs one of them to reach the supplicant in order to block the protocol. Moreover, the large number of messages is sufficient for launching a DoS attack on the supplicant side in general.

3.4 Effective Defenses

This kind of DoS attack also exists in some other protocols where the responder needs to store states, i.e. IKE [6, 65] and TCP [86, 93]. We can simply repair it by transmitting both ANonce and SNonce in *Message 3*; this improves the protocol to a stateless one [6, 10, 65]. However, even with that improvement, the 4-Way Handshake might be still vulnerable to replay attacks. On the other hand, once some mechanism is implemented to defend against the replay attack, our repairs in Section 3.4.2 can already make the handshake secure. Furthermore, adding ANonce to *Message 3* significantly change the format of the packet. Hence, we do not suggest this approach. Instead, we propose three other approaches that do not require significant modifications of the packet format or the protocol itself.

3.4.1 Random-Drop Queue

The problem here is similar to the well-studied TCP SYN flooding DoS attacks [86, 93], which can be mitigated in some known ways. The supplicant can keep a queue of all the initiated, but incomplete, handshake instances. According to the calculations in Section 3.3.4, the queue size might be too large for the supplicant; the situation becomes even worse if a longer timeout period or a higher data rate is implemented. Therefore, a feasible improvement would be to implement the queue with a random-drop policy. The supplicant maintains a certain size of queue, say, Q entries to store the states. Once all entries in the queue are filled, one of them is randomly replaced by the new state. Denote that the number of malicious *Message 1*s is n between the legitimate *Message 1* and *Message 3*, the probability that the handshake will be blocked by the malicious messages is P, we have $P = 1 - (1 - 1/Q)^n$, as shown in Figure 3.4.

From Figure 3.4, when $Q = 1$, the attacker can block the handshake with probability 1 by inserting only one message. When Q increases, the attacker needs to insert more messages in order to block the handshake with a high probability. However, increasing Q could be quite expensive and performance reductive for the supplicant. Furthermore, even a queue of size 10 is not going to help very much because the

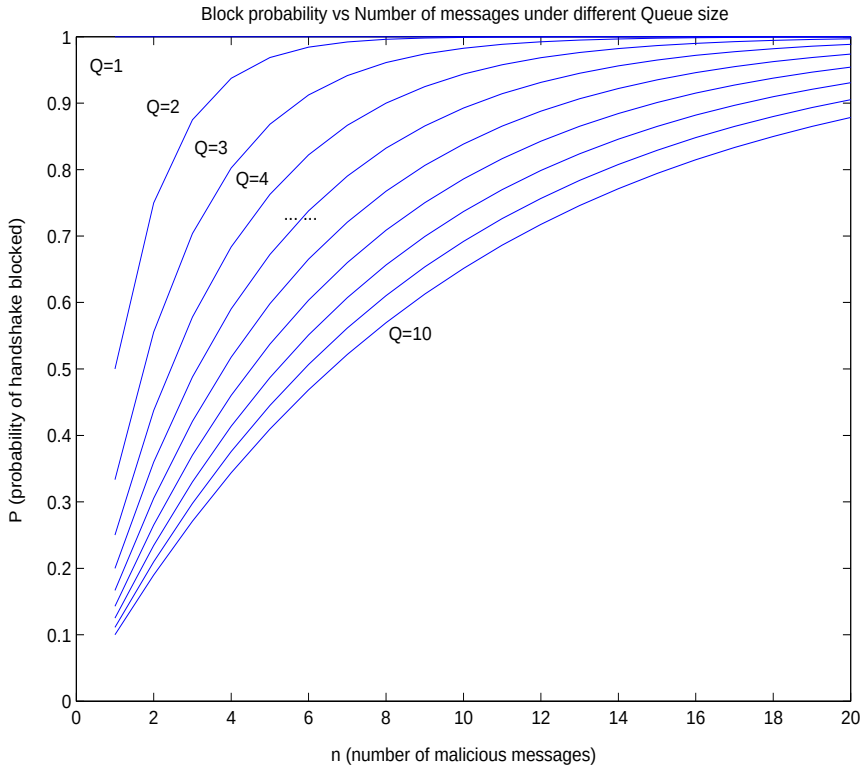

Figure 3.4: Effectiveness of random-drop queue

attacker can block the handshake with probability above 0.8 by inserting 16 messages. That is a trivial number of messages, compared to the total possible number of *Message 1*s the attacker can insert between the legitimate *Message 1* and *Message 3*.

3.4.2 Message 1 Authentication

Since there is already some common secret (PMK) shared between the authenticator and the supplicant, another possible repair is to add a MIC to *Message 1*, which will prevent the attacker from forging that message. In order to exploit the same hardware or software as in processing other messages, we can derive a trivial PTK based on the PMK and some specific values of nonces (e.g., 0), then calculate the MIC with this derived PTK. Note that after a MIC is added, *Message 1* and *Message 3* are still distinguishable by the Secure bit.

If the PMK is dynamically generated through an 802.1X authentication process, this would solve the problem. However, if a PSK or a cached PMK is used for the current PMK, the authenticated *Message 1* is still vulnerable to replay attacks since the PMK is static for a relatively long time. Therefore, the authenticator should keep a monotonically increasing sequence counter to defend against the replay attacks. One global sequence counter per authenticator appears to work for all supplicants. The supplicant can detect the replayed messages by comparing the counter of a received message against the counter of the largest-numbered previous message.

Fortunately, the requirement that the counter must be monotonically increasing appears feasible since there are apparently 8 octets set aside for this sequence counter. In fact, there appears to be sufficient space in the message format so that clock time could be used as the counter value, eliminating the possible problem of counter rollover. Furthermore, this specific sequence counter is also consistent with its usage in the group key handshakes and imposes no significant influences on other parts of the standard. Note that we need not worry about the synchronization of the clock time since only the local time in the authenticator side is used.

3.4.3 Nonce Reuse

The third repair is to eliminate the intermediate states on the supplicant side. Specifically, the supplicant can re-use the values of SNonce until a legitimate handshake is completed and a shared PTK is achieved between the supplicant and the authenticator. In other words, the supplicant does not update its nonce responding to each received *Message 1* until *Message 3* is received and verified. Note that there are no requirements for the authenticator to re-use the values of ANonce, because the legitimate ANonce will ultimately reach the supplicant via a valid *Message 3*.

In this approach the supplicant only needs to remember one SNonce of its own, which eliminates the memory DoS attack. Although it is still possible for the attacker to send out forged *Message 1*s with different nonces, the supplicant need not store every received ANonce and the corresponding PTK. It merely derives a PTK from the stored SNonce and the received ANonce, then computes a MIC from the derived PTK and sends out the corresponding *Message 2*. Upon receiving *Message 3*, the supplicant will again derive a PTK from the stored SNonce and the received ANonce, then verify the MIC using the derived PTK. Once the MIC is verified, *Message 4* is sent out and the corresponding PTK can be used as the session key.

This approach is a robust solution to the memory exhaustion attack; however, it uses more computation on the supplicant side. Specifically, the PTK is calculated twice for each received nonce: the first time when *Message 1* is received, and the second time when *Message 3* is received. If the computation power is poor for some devices, flooding *Message 3* might cause a CPU exhaustion attack, or substantially decrease the performance because the supplicant needs to re-compute the PTK first, then verify the MIC.

In practice, the CPU load of calculating the PTK varies considerably depending on the implementation and the type of CPU [Moore, email communications, May 11, 2004]. Generally, the PTK calculation is about 1.5 times slower than the calculation of MIC. However, the calculation of PTK can be improved about 4 5 times by merging the loops and pre-calculating intermediate results; the MIC calculation can be improved about 1.5 2 times by caching intermediate results. Hence, the PTK calculation does add to the CPU load, but not as much as another MIC calculation.

Of course, the supplicant can store all the received nonces and the derived PTKs to handle the computation load, but then obviously the memory exhaustion attack recurs. There is a tradeoff here that the supplicant needs to make between the memory consumption and the CPU consumption. If the environment is such that most of the messages are expected to be legitimate, the supplicant can store one copy of the derived PTK and received ANonce, and use them to verify the MIC in received *Message 3* directly. The supplicant re-computes the PTK only if the nonce in the message does not match the stored ANonce. This combined approach seems to be the most reasonable solution to the 4-Way Handshake problems.

3.5 Conclusions

The 4-Way Handshake protocol in IEEE 802.11i has been analyzed using Murϕ. We described the procedure for finite-state verification of security protocols, and our results showed that this is a very effective approach. We identified the functionality of each field in the messages, in some cases supporting assertions made in the protocol documentation, and in a few cases suggesting alternatives. A simplified protocol was presented that has the same authentication properties as the original one under our Murϕ model. Most significantly, we found and analyzed an effective DoS attack on *Message 1* in the protocol. As in many other case studies, this attack can be prevented by an extremely simple modification to the protocol. However, this clear improvement in the protocol was not apparent before our security analysis.

The protocol is supposed to allow only one active handshake at any time and to generate a shared PTK between a corresponding supplicant and authenticator. However, upon analysis we showed that the supplicant must allow multiple handshakes to execute in parallel, in order to ensure protocol completion in the presence of packet loss. This leads to vulnerabilities, allowing an attacker to block the handshake by simply inserting one forged message. Furthermore, the attacker can construct a memory DoS attack if the supplicant is implemented to store the states of all incomplete handshakes. When 802.1X authentication is implemented, the PMKID included in

Message 1 will limit the attacker to constructing this attack only after the first *Message 1* is seen in the link; Link Layer Data Encryption can protect the subsequent sessions after the first PTK is established. When both of these mechanisms are implemented, this attack can only be performed between *Message 1* and *Message 3* of the first legitimate 4-Way Handshake instance. These implementations cause more difficulties, but the attacker can still launch the attack if it keeps monitoring and intercepting the network in a timely way. However, since the attack can be prevented simply and completely, there is no need to live with a protocol subject to this attack.

We discussed and compared three repairs. First, we could use a random-drop queue of a certain size to avoid memory exhaustion without any modifications to the protocol. However, our calculations indicated that the protocol remains quite vulnerable with reasonable queue sizes. Second, with a more significant change, a MIC calculated from the PMK could be added to *Message 1* to prevent the attacker from forging *Message 1*. This remedy also requires a monotonically increasing sequence counter to be implemented in the authenticator side to prevent replay attacks. The local clock time of the authenticator appears to be a simple increasing counter that would do the job. Third, without any modification to the protocol itself, we could simply reuse the nonces in the supplicant until one legitimate 4-Way Handshake is completed. This approach inherently eliminates the vulnerability but might consume more computation power in the supplicant. We suggested the combined approach that reuses nonces and stores one entry of the received nonce and derived PTK.

Chapter 4

A Modular Correctness Proof

In the previous chapters, we have identified several vulnerabilities in IEEE 802.11i and verified the security properties using Murϕ. However, it is not a sufficient proof of the 802.11i correctness because the finite-state verification only deals with a bounded configuration. In this chapter, we present a formal correctness proof of the 802.11i protocols using *Protocol Composition Logic (PCL)* [34, 35, 25, 28, 26, 27, 68], which we believe is the first complete proof of 802.11i for an unbounded number of participants and sessions. Furthermore, the formal proof for the TLS protocol has independent interest since TLS is widely used independent of 802.11i (e.g. [1]).

Our proof consists of separate proofs of specific security properties for 802.11i components - the TLS authentication phase, the 4-Way Handshake protocol and the Group Key Handshake protocol. Using a new form of PCL composition principle, formulated as *staged composition* in this chapter, we combine the component proofs to show that any staged use of the protocol components achieves the stated security goals. It follows that the components compose securely for a range of failure recovery control flows, including the improvements proposed in Chapter 2 and [41]. The general result also proves security for other configurations presented in the 802.11i specifications, including the use of a Pre-Shared Key (PSK) or cached Pair-wise Master Key (PMK). In addition to devising a new composition principle for PCL, we also extend the logic to handle local memory associated with reusing generated nonces.

The memory feature is needed to prove correctness of an unusual feature of the improved 4-Way Handshake protocol [41] that involves reusing a nonce to avoid a Denial of Service (DoS) attack.

In previous work, PCL has been proved sound for protocol runs that use any number of principals and sessions, over both symbolic models and (for a subset of the logic at present) over more traditional cryptographic assumptions [24], which implies security for an unbounded number of participants and sessions. An advantage of PCL is that each proof component identifies not only the local reasoning that guarantees the security goal of that component, but also the environmental conditions that are needed to avoid destructive interference from other protocols that may use the same certificates or key materials. These environment assumptions are then proved for the steps that require them, giving us an invariant that is respected by the protocol. In formulating the proof, we identify the precise conditions that will allow other protocols to be executed in the same environment without interfering with 802.11i. Moreover, our proof provides certain insights into the component protocols.

Our results also suggest that PCL is suitable for compositional analysis of large protocols, yielding assurance and guidelines for implementation and deployment. Among the methods and security studies carried out in recent years, such as [90, 18, 66, 98, 69, 74, 72, 71, 83, 2], the closest to our study appear to be Paulson's investigations [84, 85] of TLS and SET [94]. Some advantages of our approach over Paulson's inductive method are a compositional proof method and a higher level of abstraction. Paulson's method involves direct reasoning about an inductively defined set of protocol execution traces, built from the protocol specification and attacker actions. However, PCL has an axiomatic system that abstracts away arguments about traces and eliminates the need to reason explicitly about attacker actions. This feature, along with the Floyd-Hoare style specification, makes proofs in PCL concise and readable. Because the semantic soundness of PCL shows that each of the axioms and inference rules are correct for arbitrary protocol runs in the presence of a symbolic attacker, the PCL proof system provides the same semantic guarantees as Paulson's method, without requiring a set of lemmas to be reproved for each protocol that is studied.

The rest of the chapter is organized as follows. Section 4.1 briefly describes the IEEE 802.11i control flow and the Protocol Composition Logic (PCL). Sections 4.2, 4.3 and 4.4 present the analysis of the 4-Way Handshake, the TLS protocol, and the Group Key Handshake, respectively. Section 4.5 describes the staged composition principle which takes into account the structure of complicated control flows and proves the safe composition of various components of 802.11i. Finally, Section 4.6 concludes the chapter.

4.1 Overview

4.1.1 The 802.11i Control Flow

As we have described in Chapter 2, the IEEE 802.11i Standard [50] defines a "Robust Security Network Association"(RSNA) for data confidentiality, integrity, and mutual authentication. In this chapter, we focus on proving the correctness of the RSNA establishment procedures - the mutual authentication and key management protocols. A typical RSNA establishment procedure starts by executing an EAP authentication between the supplicant and the authentication server, typically using EAP-TLS, with the authenticator acting as a relay. After the successful completion of the EAP-TLS session, the supplicant and the authentication server verified each other's identity and agreed on a shared secret. Then the authentication server moves the secret to the authenticator; the authenticator and supplicant derive a shared Pair-wise Master Key (PMK) from this secret. Afterwards, the authenticator and the supplicant execute a session of the 4-Way Handshake protocol, from which a fresh Pair-wise Transient Key (PTK) is derived to secure subsequent data traffic. Note that, in practice, the authentication server can be implemented either in a single device with the authenticator, or through a separate server. In the latter case, it is assumed that the link between the authentication server and the authenticator is physically secure. Therefore, while modelling the protocol, it is safe to make a simplifying assumption that the authentication server and the authenticator are the same principal.

While the typical run described above is relatively straightforward, the complete

specification is much more complicated due to additional optional components and alternative configurations. For example, 802.11i can adopt other EAP methods for authentication, such as password-based authentication, instead of EAP-TLS. Moreover, in the case of multicast applications, the authenticator can also distribute a fresh group key to all supplicants in a group after the PTK has been established.

In this chapter, we focus on the complete RSNA establishment procedure that consists of four components: TLS, 4-Way Handshake, Group Key Handshake, and data sessions. These components are designed to be executed sequentially; however, in order to prove security properties of this procedure, we also have to consider all other possible executions. For example, 4-Way Handshakes can be periodically re-run to refresh the PTK. Furthermore, failure of one component leads to other possible execution sequences. In the original 802.11i specification, the entire sequence is restarted if one component fails, as shown in Figure 4.1(a). As observed in [41], this failure recovery mechanism is quite inefficient and may be improved as shown in Figure 4.1(b). Therefore, we formulate our proof in a way that demonstrates the desired security properties for both control flow graphs.

4.1.2 The Proof Method

We use Protocol Composition Logic (PCL) [34, 25, 28, 27] to prove correctness of the 802.11i protocols. This section contains a brief discussion of PCL relevant to this analysis.

Modelling protocols A protocol is defined by a set of roles, each specifying a sequence of actions to be executed by an honest agent. Protocol roles are represented using a simple "protocol programming language" based on *cords* [34]. Figure 4.2 shows a simple two message protocol in the informal arrows-and-messages notation and the formal programs for roles of the same protocol using the cords notation. Program **Init** describes the actions of a *thread* X executing the initiator role in the protocol with *agent* $\hat{Y}$ as the responder. The possible protocol actions include nonce generation, signatures and encryption, communication steps, and decryption and signature verification via pattern matching. Programs can also depend on input

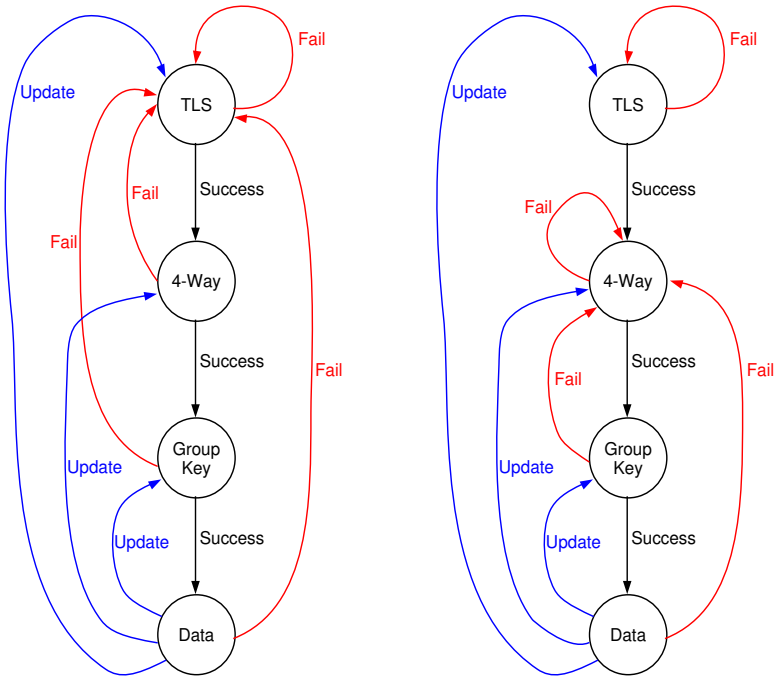

Figure 4.1: The 802.11i Control Flow

$$\begin{array}{rcl} X \to Y & : & x \\ Y \to X & : & x, SIG_Y(X,x) \end{array}$$

$$\begin{array}{rcl} \textbf{Init} & = & (X,\hat{Y})[\texttt{new } x; \texttt{send } \hat{X},\hat{Y},x; \texttt{receive } \hat{Y},\hat{X},x,s; \texttt{match } s/SIG_Y(X,x)]_X \\ \textbf{Resp} & = & (Y)[\texttt{receive } \hat{X},\hat{Y},x; \texttt{send } \hat{Y},\hat{X},SIG_Y(X,x)]_Y \end{array}$$

Figure 4.2: Arrows-and-messages vs cords

parameters (typically determined by context or the result of set-up operations) and provide output parameters to subsequent operations.

Protocol logic and the proof system The syntax of the logic and informal descriptions of the logical predicates is given in [35, 27]. Most protocol proofs use formulas of the form $\theta[P]_X\phi$, which means that starting from a state where formula θ is true, after actions P are executed by the thread X, the formula ϕ is true in the resulting state. Formulas ϕ and ψ typically make assertions about temporal order of actions (useful for stating authentication) and/or the data accessible to various principals (useful for stating secrecy).

The proof system extends first-order logic with axioms and proof rules for protocol actions, temporal reasoning, knowledge, and a specialized form of invariance rule called the *honesty rule*. The honesty rule is essential for combining facts about one role with inferred actions of other roles. Intuitively, if Alice receives a response from a message sent to Bob, the honesty rule captures Alice's ability to use properties of Bob's role to reason about how Bob generated his reply. In short, if Alice assumes that Bob is honest, she may use Bob's role to reason from this assumption.

Compositional proof method Following the modular design of the protocol, we extensively use the compositional approach developed in [28, 27] and prove properties of the whole protocol by combining proofs of its parts.

We separately prove security guarantees of the form $\Gamma \vdash \theta[P]_X\phi$ for 4-Way Handshake, TLS, and the Group Key Handshake in Sections 4.2, 4.3, and 4.4 respectively. Here Γ includes invariants of the specific protocol component. Assuming that these

invariants are satisfied, the respective components possess the stated properties. For each component, invariants are proved using the honesty rule.

In order to prove properties of the complete protocol, we combine guarantees provided by the different components. For example, the 4-Way Handshake provides authentication assuming that the Pair-wise Master Key established by TLS is a shared secret between the supplicant and the authenticator. The combined protocol consisting of a TLS session followed by a 4-Way Handshake therefore provides authentication. Technically, sequential composition involves matching a *postcondition* of one protocol to a *precondition* of the other, as well as checking that the two protocols satisfy each other's invariants. Ensuring that the protocol components compose safely given the error handling mechanisms in Figure 4.1 requires an extension of the existing composition theorems. The new composition theorem as well as its application to 802.11i is presented in Section 4.5.

We do not present an analysis of the data confidentiality protocol in this chapter. Since the current logic is based on an execution model assuming idealized cryptography, a correctness proof of the data transfer protocol is unlikely to be very informative. However, it could be prudent to study the data confidentiality protocol in another manner.

4.2 4-Way Handshake

In this section, we prove security properties of the original 4-Way Handshake and the modified 4-Way Handshake protocol proposed in Chapter 2 and [40, 41]. Note that the 4-Way Handshake generates the Pairwise Temporary Key (PTK) for data confidentiality protocols and the Group Key Handshake Protocol, using a pre-established secret shared between the authenticator and the supplicant. The pre-established secret, called the Pair-wise Master Key (PMK), may be set up via mutual authentication protocols (de facto EAP-TLS), or it may be pre-configured as a Pre-Shared Key (PSK).

4WAY : AUTH	=	$(X, \hat{Y}, pmk)$
		[`new` x; `send` $\hat{X}, \hat{Y}, x$, *"msg1"*;
		`receive` $\hat{Y}, \hat{X}, z$; `match` z/y, *"msg2"*, $mic1$;
		`match` $HASH_{pmk}(x, y)/ptk$;
		`match` $mic1/HASH_{ptk}(y,$ *"msg2"*$)$;
		`send` $\hat{X}, \hat{Y}, x$, *"msg3"*, $HASH_{ptk}(x,$ *"msg3"*$)$;
		`receive` $\hat{Y}, \hat{X}, w$;
		`match` $w/$ *"msg4"*, $mic2$; `match` $mic2/HASH_{ptk}($ *"msg4"*$)]_X$
4WAY : SUPP	=	(Y, pmk)
		[`receive` $\hat{X}, \hat{Y}, z$; `match` z/x, *"msg1"*;
		`new` y; `match` $HASH_{pmk}(x, y)/ptk$;
		`send` $\hat{Y}, \hat{X}, y$, *"msg2"*, $HASH_{ptk}(y,$ *"msg2"*$)$;
		`receive` $\hat{X}, \hat{Y}, w$;
		`match` w/x, *"msg3"*, mic; `match` $mic/HASH_{ptk}(x,$ *"msg3"*$)$;
		`send` $\hat{Y}, \hat{X},$ *"msg4"*, $HASH_{ptk}($ *"msg4"*$)]_Y$

Table 4.1: 4-Way Handshake Program

4.2.1 Modelling 4-Way

During the handshake, the authenticator and supplicant generate fresh nonces, then derive a fresh PTK based on the shared PMK, the nonces, and their MAC addresses. They authenticate the key material generated using keyed hashes. The authenticator and supplicant roles of the 4-Way Handshake are described formally using the programming language introduced in the previous section, which are listed in Table 4.1. We describe the authenticator program **4WAY** : **AUTH** in details below.

This program has three input parameters - the authenticator and the supplicant identifiers, and the PMK represented by the variable pmk. The first action executed by the authenticator X involves generating a fresh nonce x using the action `new` . Then X sends out $Message$ 1 to Y, which contains the nonce x and the string "msg1". In practice, message indicators are represented by sequences of bits, but we use strings here for readability. Authenticator X waits to receive a response from Y and then checks whether the received message is the expected $Message$ 2 using

a `match` action, which verifies the Message Integrity Code (MIC) based on the *ptk* derived. If *Message* 2 is valid, X sends out *Message* 3 including the nonce x, the string "msg3" and the MIC, and waits for the response. Once a valid *Message* 4 is received and verified, X completes the 4-Way Handshake and subsequently uses the derived *ptk*.

Note that in the 802.11i specifications, the PTK is divided into several parts: KCK (Key Confirmation Key) for computing MIC, KEK (Key Encryption Key) for encrypting the group key, and TK (Temporary Key) for protecting data packets. For expository convenience, we use *ptk* to refer to all of these parts.

4.2.2 Security Properties

The desired security properties for the 4-Way Handshake as identified by the standard (see Section 8.4.8 in [50]) are:

1. Confirm the existence of the PMK at the peer.

2. Ensure that the security association key (PTK) is fresh.

3. Synchronize the installation of session keys into the MAC.

4. Transfer the GTK from the Authenticator to the Supplicant.

5. Confirm the selection of cipher suites.

The definitions below formalize two security properties called *key secrecy* and *session authentication*. Items 1, 2, 3, and 5 are captured by *session authentication*; moreover, *session authentication* can be asserted only when the *key secrecy* is guaranteed. We discuss item 4 in Section 4.4. These conditions state the security guarantees for the authenticator precisely. The guarantees for the supplicant are analogous, but omitted due to space constraints. Informally, the formula $\phi_{4way,sec}$ below states that only the authenticator and the supplicant possess the PTK.

Definition 4.2.1 (4-way key secrecy)
The 4-Way Handshake is said to provide key secrecy if $\phi_{4way,sec}$ holds, where

$$\begin{aligned}\phi_{4way,sec} ::= &\ \mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset \\ &((\mathsf{Has}(\hat{Z}, ptk) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{Y})) \wedge \\ &\mathsf{Has}(\hat{X}, ptk) \wedge \mathsf{Has}(\hat{Y}, ptk)\end{aligned}$$

The formula below formalizes a standard notion of authentication called *matching conversations* [14]. It guarantees that the two principals have consistent views of protocol runs. It follows that they agree on terms such as the cipher suite and the freshly generated PTK.

Definition 4.2.2 (4-way authentication)
The 4-Way Handshake is said to provide session authentication for the authenticator if $\phi_{4way,auth}$ holds, where

$$\begin{aligned}\phi_{4way,auth} ::= &\ \mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset \\ &\exists Y. ActionsInOrder(\\ &\mathsf{Send}(X, \hat{X}, \hat{Y}, Message\ 1), \mathsf{Receive}(Y, \hat{X}, \hat{Y}, Message\ 1), \\ &\mathsf{Send}(Y, \hat{Y}, \hat{X}, Message\ 2), \mathsf{Receive}(X, \hat{Y}, \hat{X}, Message\ 2), \\ &\mathsf{Send}(X, \hat{X}, \hat{Y}, Message\ 3), \mathsf{Receive}(Y, \hat{X}, \hat{Y}, Message\ 3), \\ &\mathsf{Send}(Y, \hat{Y}, \hat{X}, Message\ 4), \mathsf{Receive}(X, \hat{Y}, \hat{X}, Message\ 4))\end{aligned}$$

The main result of this section is Theorem 1, which states the security guarantees for the authenticator. A proof of this theorem appears in Appendix B. A similar guarantee holds for the supplicant. We omit the theorem and the proofs for the supplicant due to space constraints.

Theorem 1 (4-Way Authenticator Guarantee)
(i) On execution of the authenticator role, key secrecy *and* session authentication *are*

guaranteed if the formulas in Table 4.2 hold. Formally,

$$\Gamma_{4way,1} \wedge \Gamma_{4way,2} \vdash \theta_{4way}[4WAY{:}AUTH]_X \phi_{4way,auth} \wedge \phi_{4way,sec}$$

(ii) $\Gamma_{4way,1}$ *is invariant of the 4-Way Handshake;* $\Gamma_{4way,2}$ *is an assumption on the environment. Formally,* $4WAY \vdash \Gamma_{4way,1}$

The theorem states that starting from a state in which the precondition holds, if the authenticator role is executed, then in the resulting state the desired authentication and secrecy properties are guaranteed. The precondition θ_{4way} listed in Table 4.2 requires that the PMK is only sent out under encryption. The authenticator deduces its security properties based on the actions that it performs, the properties of certain cryptographic primitives and knowledge of the behavior of an honest supplicant. (By definition, an honest principal behaves in accordance with the protocol.) In the case of the 4-Way Handshake, the expected behavior of an honest principal is captured by the formulas $\Gamma_{4way,1}$ and $\Gamma_{4way,2}$ listed in Table 4.2. The first statement of the theorem states that, assuming these formulas hold, the security property is guaranteed.

Invariants are generally proved by induction over programs using the *Honesty Rule*. However, the detailed proofs are omitted here due space limitations. The second part of the theorem states that $\Gamma_{4way,1}$ is an invariant of the 4-Way Handshake protocol. The formula $\Gamma_{4way,2}$ states that no principal performs the roles of both the supplicant and the authenticator. This is a reasonable assumption for a typical 802.11i deployment. As noted in [41], security of 802.11i may be compromised if this condition is violated.

4.2.3 Operating Environment

In this section, we discuss the characteristics of an operating environment in which the 4-Way Handshake Protocol provides security guarantees. The first goal is to identify the class of protocols that may run concurrently with the 4-Way Handshake without degrading its security; the second goal is to identify other application scenarios in which the 4-Way Handshake may be deployed safely. Our analysis provides insight in both directions.

$$
\begin{array}{lcl}
\theta_{4way} & := & \mathsf{Has}(\hat{X}, pmk) \wedge \mathsf{Has}(\hat{Y}, pmk) \wedge \mathsf{NonceSource}(Y, pmk, ENC_{\hat{X}}(pmk)) \\
\Gamma_{4way,1} & := & \mathsf{Computes}(\hat{X}, HASH_{pmk}(x, y)) \supset \\
 & & \quad \neg(\mathsf{Send}(\hat{X}, m) \wedge \mathsf{Contains}(m, HASH_{pmk}(x, y)) \\
\Gamma_{4way,2} & := & (\mathsf{Honest}(\hat{X}) \wedge \mathsf{Receive}(X, Message\ 1) \supset \neg\mathsf{Send}(X, Message\ 3)) \wedge \\
 & & (\mathsf{Honest}(\hat{X}) \wedge \mathsf{Send}(X, Message\ 1) \supset \\
 & & \quad \neg(\mathsf{Send}(X, Message\ 2) \wedge \mathsf{Send}(X, Message\ 4)))
\end{array}
$$

Table 4.2: 4-Way Handshake Precondition and Invariants

As discussed above, in order to provide the *key secrecy* and *session authentication* properties, the 4-Way Handshake must be executed in an environment where the formulas listed in Table 4.2 are satisfied. $\Gamma_{4way,1}$ states that the supplicant and the authenticator derive the PTK locally and do not reveal it. It is possible that protocols executing concurrently with the 4-Way Handshake may share state and have access to these shared secrets. In this case, our formulas require that these protocols do not reveal these secrets. Note that the shared PMK should also be kept secure; we will discuss that in the TLS invariants.

$\Gamma_{4way,2}$ states the requirement that an honest principal does not execute both the authenticator and the supplicant roles. It expresses this by requiring that a principal that performs actions corresponding to one role does not perform actions corresponding to the other. If $\Gamma_{4way,2}$ does not hold, a simple reflection attack can be demonstrated [41]. It is highly unlikely that this condition would be violated in a wireless LAN environment, as we do not expect a laptop to play the role of an authenticator, or an access point to play the role of a supplicant. However, 802.11i may be deployed in an ad-hoc network environment, in which case it is conceivable that nodes play both roles and violate this assumption.

4.2.4 Improved 4-Way Handshake

The 4-Way Handshake, described in the previous section suffers from a DOS vulnerability [40, 41]. The vulnerability results from the lack of any authentication in

Message 1, which allows the attacker to block the supplicant role. It is possible to work around this flaw by allowing an arbitrary number of sessions, but this may result in a memory exhaustion attack since the supplicant must store all the nonces that it generates for various sessions. A modification that involves nonce re-use is discussed in [40, 41] and has been adopted by the 802.11i standards committee. The modified supplicant program is listed in the pseudo-code that follows, while the authenticator program is the same as the one in Table 4.1.

Algorithm 4.2.1: MOD-4-WAY:SUPP(Y, pmk)

`new` y;
repeat
`receive` $\hat{X}, \hat{Y}, z$;
if `match` z/x, *"msg1"*;
then
`match` $HASH_{pmk}(x, y)/ptk$;
`send` $\hat{Y}, \hat{X}, y$, *"msg2"*, $HASH_{ptk}(y,$ *"msg2"*$)$;
until
`match` z/x, *"msg3"*, $HASH_{ptk}(x,$ *"msg3"*$)$;
`send` $\hat{Y}, \hat{X}$, *"msg4"*, $HASH_{ptk}($ *"msg4"*$)$;

It is easy to see that the modified protocol cannot be blocked by the attacker. Also re-using the nonce until one 4-Way Handshake completes allows the supplicant to avoid storing state, which prevents memory exhaustion. Since the suggested fix involves nonce reuse, and nonces are generally used to provide freshness guarantees, it is not obvious that the *authentication* property is preserved under this modification. The main result of this section is the following theorem.

Theorem 2 (Modified 4-Way Guarantee)
For the authenticator, the session authentication *and the* key secrecy *guarantees for the modified protocol are identical to Theorem 1.*

Intuitively the authenticator guarantee continues to hold because nonce re-use occurs within the *repeat - until loop* of a single session. The formalization of authentication - matching conversations - usually refers to all the actions of a principal in a session. Since an attacker can inject a forged $Message\ 1$ an arbitrary number of times, and the supplicant will respond to it, we exclude certain messages from the matching conversation. Authentication in this form is implied by our proof. The supplicant guarantees are identical to the original 4-Way Handshake.

4.3 TLS

The TLS/SSL [29] protocol provides end-to-end security and is widely deployed on the Internet in various security and e-commerce systems. IEEE 802.11i suggests EAP-TLS, which is an encapsulated version of TLS protocol, to mutually authenticate the supplicant and the authenticator, and to derive a shared secret key (PMK). In addition to proving TLS secure in isolation, we improve previous analysis of TLS [84, 74, 72, 71] by identifying conditions under which other protocols may run concurrently without introducing vulnerabilities. Identifying such conditions appears valuable given the wide deployment of SSL/TLS. Note that we use the terms *client* and *server* for TLS protocol participants, as in the TLS documentation [29]. When TLS is used with 802.11i, the *client* corresponds to the *supplicant* and the *server* can reside in the *authenticator*.

4.3.1 Modelling TLS

TLS has many possible modes of operation. We restrict our attention to the mode where both the server and the client have certificates, since this mode satisfies the mutual authentication property requirement of the 802.11i Standard.

The **TLS** : **Client** and **TLS** : **Server** programs are described in Table 4.3, where $VerSU$ represents the protocol version and cipher suite, K_y is the server's public key, Vx is the client's verification key. We use the `match` action to check signatures, verify keyed hashes and perform decryption. Note that the term *handShake1*

$$
\begin{array}{rcl}
\textbf{TLS} : \textbf{Client} & = & (X, \hat{Y}, VerSUx)[\\
& & \texttt{new } Nx; \texttt{ send } \hat{X}, \hat{Y}, Nx, VerSUx; \\
& & \texttt{receive } \hat{Y}, \hat{X}, Ny, VerSUy, cert; \\
& & \texttt{match } cert/SIG_{\hat{CA}}(\hat{Y}, Ky); \\
& & \texttt{new } secret; \\
& & \texttt{send } \hat{X}, \hat{Y}, SIG_{\hat{CA}}(\hat{X}, Vx), SIG_{Vx}(handShake1), \\
& & ENC_{Ky}(secret), HASH_{secret}(handShake1, \textit{“client”}); \\
& & \texttt{receive } \hat{Y}, \hat{X}, hash; \\
& & \texttt{match } hash/HASH_{secret}(handShake2, \textit{“server”});]_X \\
\textbf{TLS} : \textbf{Server} & = & (Y, VerSUy)[\\
& & \texttt{receive } \hat{X}, \hat{Y}, Nx, VerSUx; \texttt{ new } Ny; \\
& & \texttt{send } \hat{Y}, \hat{X}, Ny, VerSUy, SIG_{\hat{CA}}(\hat{Y}, Ky); \\
& & \texttt{receive } \hat{X}, \hat{Y}, cert, sig, encsec, hash; \\
& & \texttt{match } cert/SIG_{\hat{CA}}(\hat{X}, Vx); \\
& & \texttt{match } sig/SIG_{Vx}(handShake1); \\
& & \texttt{match } encsec/ENC_{Ky}(secret); \\
& & \texttt{match } hash/HASH_{secret}(handShake1, \textit{“client”}); \\
& & \texttt{send } \hat{Y}, \hat{X}, HASH_{secret}(handShake2, \textit{“server”});]_Y
\end{array}
$$

Table 4.3: TLS: Client and Server Programs

and *handShake2* represent the concatenation of all the terms sent and received by a principal up to the point it is used in the program.

4.3.2 Security Properties

The properties that TLS [29] ought to satisfy include:

1. The principals agree on each other's identity, protocol completion status, the values of the protocol version, cryptographic suite, and the *secret* that the client sends to the server. For server $\hat{Y}$, communicating with client $\hat{X}$, this property is formulated in Definition 4.3.1.

2. The *secret* that the client generates should not be known to any other principal other than the client and the server. For server $\hat{Y}$ and client $\hat{X}$, this property is formulated in Definition 4.3.2.

Definition 4.3.1 (TLS Authentication)
TLS is said to provide session authentication for the server role if $\phi_{tls,auth}$ holds, where

$$\begin{aligned}\phi_{tls,auth} ::= {} & \mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset \\ & \exists X.ActionsInOrder(\\ & \mathsf{Send}(X,\hat{X},\hat{Y},m1), \mathsf{Receive}(Y,\hat{X},\hat{Y},m1), \\ & \mathsf{Send}(Y,\hat{Y},\hat{X},m2), \mathsf{Receive}(X,\hat{Y},\hat{X},m2), \\ & \mathsf{Send}(X,\hat{X},\hat{Y},m3), \mathsf{Receive}(Y,\hat{X},\hat{Y},m3), \\ & \mathsf{Send}(Y,\hat{Y},\hat{X},m4))\end{aligned}$$

and m1, m2, m3, m4 represent the corresponding TLS messages in Table 4.3.

Definition 4.3.2 (TLS Key Secrecy)
TLS is said to provide secrecy if $\phi_{tls,sec}$ holds, where

$$\begin{aligned}\phi_{tls,sec} ::= {} & \mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset \\ & \mathsf{Has}(\hat{X}, secret) \wedge \mathsf{Has}(\hat{Y}, secret) \wedge \\ & (\mathsf{Has}(\hat{Z}, secret) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{Y})\end{aligned}$$

We use the proof system to prove guarantees for both the client and the server. Due to space constraints, we list only the guarantee for the authenticator in Theorem 3. The client guarantee is similar. The secrecy of the exchanged key material in TLS is established by combining local reasoning based on the client's actions with global reasoning about actions of honest agents. Intuitively, a client that generates the secret only sends it out either encrypted with an honest party's public key or uses it as a key for a keyed hash (this is captured by the predicate NonceSource). Furthermore, no honest user will ever decrypt the secret and send it in the clear. Specifically, an honest party can send the secret in the clear only if it receives it in the clear first (this is captured by the TLS invariant $\Gamma_{tls,2}$). Secrecy follows directly from these two facts.

Theorem 3 (TLS Server Guarantee)
(i) On execution of the server role, key secrecy *and* session authentication *are guaranteed if the formulas in Table 4.4 hold. Formally,*

$$\Gamma_{tls,1} \wedge \Gamma_{tls,2} \vdash [\mathit{TLS{:}Server}]_X \phi_{tls,auth} \wedge \phi_{tls,sec}$$

(ii) The formulas in Table 4.4 are invariants of TLS. Formally, $\mathit{TLS} \vdash \Gamma_{tls,1} \wedge \Gamma_{tls,2}$

4.3.3 Operating Environment

We now characterize the class of protocols that compose safely with TLS. As in Section 4.2.3, we relate *invariants* in Table 4.4 to deployment considerations.

$\Gamma_{tls,1}$ states that messages of a certain format should not be sent out by any protocol that executes in the same environment as TLS. One set of terms represent keyed hashes of the handshake, where the key is the shared secret established by a TLS session; another set refers to signatures on the handshake messages. A client running a protocol that signs messages indiscriminately could cause the loss of the authentication property. Such an attack would only be possible if the client certificate used by TLS were shared with other protocols and $\Gamma_{tls,1}$ were violated by them.

$\Gamma_{tls,2}$ rules out an undesirable sequence of actions that may allow an intruder to learn the shared secret. Intuitively, if an honest principal is tricked into decrypting a

$$
\begin{array}{rcl}
\Gamma_{tls,1} & := & \neg\exists m.\mathsf{Send}(X,m) \wedge (\mathsf{Contains}(m, HASH_{secret}(handShake1, \text{“server”}))\vee \\
& & \quad \mathsf{Contains}(m, HASH_{secret}(handShake2, \text{“client”}))\vee \\
& & \quad \mathsf{Contains}(m, SIG_{Vx}(handShake1))) \\
\Gamma_{tls,2} & := & \mathsf{Honest}(\hat{Y}) \wedge \mathsf{Send}(Y,m) \wedge \mathsf{ContainsOut}(m, secret, ENC_{Ky}(secret)) \supset \\
& & \quad (\neg\mathsf{Decrypts}(Y,m') \wedge \mathsf{Contains}(m', secret))\vee \\
& & \quad (\mathsf{Receives}(Y,m'') < \mathsf{FirstSend}(Y, secret)\wedge \\
& & \quad \mathsf{ContainsOut}(m'', secret, ENC_{Ky}(secret)))
\end{array}
$$

Table 4.4: TLS Invariants

term containing the secret using its private key, after which it sends out the contents of the encryption, the *secrecy property* of TLS is lost. Clearly, if principals use an exclusive public/private key pair for TLS , such an attack is not possible. However, since another protocol (or another stage of 802.11i) may uses the same public/private key pair as TLS, it is important to check that these formulas are invariants of any other protocol.

4.4 Group Key Handshake

In multicast applications, the authenticator may distributes a Group Temporary Key (GTK) to supplicants. *Messages 3, 4* of the 4-Way Handshake may optionally set up this key distribution. The authenticator then runs the Group Key Handshake protocol periodically to update the GTK. In this section we prove the correctness of the Group Key Handshake.

4.4.1 Modelling Group Key Handshake

The programs for the Group Key Handshake are listed in Table 4.5. The authenticator sends *GrpMessage1* containing the GTK, and the supplicant confirms receipt of the GTK by sending *GrpMessage2*. The authenticator encrypts the GTK under the Key Encryption Key (KEK) (represented by term *ptk*) and sends it to the supplicant. The authenticator monotonically increases the sequence number for every key exchange

GK : AUTH	=	$(X, \hat{Y}, CurrSeqNo, ptk, gtk)[$ `match` $Succ(CurrSeqNo)/NewSeqNo$; `send` $\hat{X}, \hat{Y}, NewSeqNo$, *"grp1"*, $ENC_{ptk}(gtk)$, $HASH_{ptk}(NewSeqNo$, *"grp1"*, $ENC_{ptk}(gtk))$; `receive` $\hat{Y}, \hat{X}, z$; `match` $z/NewSeqNo$, *"grp2"*, $HASH_{ptk}(NewSeqNo$, *"grp2"*$)]_X$
GK : SUPP	=	$(Y, \hat{X}, OldSeqNo, OldGTK, ptk)[$ `receive` $\hat{X}, \hat{Y}, z$; `match` $z/NewSeqNo$, *"grp1"*, $ENC_{ptk}(gtk)$, $HASH_{ptk}(NewSeqNo$, *"grp1"*, $ENC_{ptk}(gtk))$; `isLess` $OldSeqNo, NewSeqNo$; `send` $\hat{Y}, \hat{X}, NewSeqNo$, *"grp2"*, $HASH_{ptk}(NewSeqNo$, *"grp2"*$)]_Y$

Table 4.5: Group Key Handshake Programs

message sent to prevent replay attacks. MICs are used to provide authentication and message integrity. The sequence number comparison `isLess` (a, b) is used by the supplicant to check that $a < b$; the expression $Succ(a)$ represents a number greater than a.

4.4.2 Security Properties

The properties we prove for the Group Key Handshake are listed below.

1. The Supplicant is assured that the GTK received in the current Group Key Handshake was sent by the Authenticator, and was generated by the Authenticator after the GTK that the supplicant holds from a previous Group Key Handshake or 4-Way Handshake. This is called the *key ordering* property, and is formalized in Definition 4.4.1.

2. The Authenticator is assured that the principals with knowledge of the GTK must have executed a 4-Way Handshake with the Authenticator. This is called the *key secrecy* property, and formalized in Definition 4.4.2.

Definition 4.4.1 (group key ordering)
For a supplicant $\hat{Y}$, the Group Key Handshake is said to provide key ordering if $\phi_{gk,ord}$ holds, where

$$\begin{aligned}
&\phi_{gk,ord} ::= \mathsf{Honest}(\hat{X}) \supset \\
&(\mathsf{Send}(X, \hat{X}, \hat{Y}, SeqNo1, ENC_{ptk}(gtk1)) \wedge \\
&\mathsf{Send}(X, \hat{X}, \hat{Y}, SeqNo2, ENC_{ptk}(gtk2)) \wedge \\
&\mathsf{isLess}(SeqNo1, SeqNo2) \supset \\
&\mathsf{FirstSend}(X, \hat{X}, \hat{Y}, SeqNo1, ENC_{ptk}(gtk1)) < \\
&\mathsf{FirstSend}(X, \hat{X}, \hat{Y}, SeqNo2, ENC_{ptk}(gtk2)))
\end{aligned}$$

Definition 4.4.2 (group key secrecy)
For an authenticator $\hat{X}$, the Group Key Handshake is said to provide key secrecy if $\phi_{gk,sec}$ holds, where

$$\begin{aligned}
&\phi_{gk,sec} ::= \mathsf{Honest}(\hat{Z}_1) \wedge \mathsf{Honest}(\hat{Z}_2) \ldots \mathsf{Honest}(\hat{Z}_n) \supset \\
&\quad ((\mathsf{Has}(\hat{Z}, gtk) \wedge \hat{Z} \neq \hat{X}) \supset \\
&\quad \hat{Z} = \hat{Z}_1 \vee \hat{Z} = \hat{Z}_2 \ldots \vee \hat{Z} = \hat{Z}_n)
\end{aligned}$$

Note that $\hat{Z}_1,\ \hat{Z}_2,\ \ldots\ \hat{Z}_n$ lists the set of supplicants that the authenticator intends to send the GTK. Each member of this set shares a PTK with the authenticator, established by a 4-Way Handshake. Though the 802.11i standard is not very clear with regards to the supplicant guarantee, two types of properties are conceivable - key freshness and key secrecy. However, key freshness cannot be achieved for the supplicant since a message could be lost; hence, we consider key ordering instead, which is a weaker requirement. Obviously the authenticator knows the key ordering since it generates the keys. Key secrecy can be guaranteed only for the authenticator because the supplicants do not have knowledge of the other supplicants in the group. Theorem 4 states the guarantee for the supplicant and the authenticator.

Theorem 4 (Group Key Guarantee)
(i) After execution of the supplicant role, key ordering *is guaranteed if the formulas in Table 4.6 hold. Formally,*

$$\Gamma_{gk,1} \wedge \Gamma_{gk,2} \wedge \Gamma_{gk,3} \vdash \theta_{gk}[GK{:}SUPP]_Y \phi_{gk,ord}$$

(ii) After execution of the authenticator role, key secrecy *is guaranteed if the formulas in Table 4.6 hold. Formally,*

$$\Gamma_{gk,1} \wedge \Gamma_{gk,2} \wedge \Gamma_{gk,3} \vdash \theta_{gk}[GK{:}AUTH]_X \phi_{gk,sec}$$

(iii) $\Gamma_{gk,1}$ *and* $\Gamma_{gk,2}$ *are invariants of the Group Key Handshake;* $\Gamma_{gk,3}$ *is an assumption on the environment. Formally,* $GK \vdash \Gamma_{gk,1} \wedge \Gamma_{gk,2}$

4.4.3 Operating Environment

In this section, we identify the class of operating conditions under which the Group Key Handshake provides security. Table 4.6 lists formulas required for the Group Key Handshake to function correctly. As in Sections 4.2.3 and 4.3.3, we regard these formulas as specifications for a safe operating environment.

$\Gamma_{gk,1}$ states that the authenticator should increase the sequence counter monotonically to guarantee the key ordering property for the supplicant. As pointed out in Section 3.2.3, the sequence numbers play an important role in the Group Key Handshake, but are redundant in the 4-Way Handshake. Our proof here indicates that sequence numbers in the 4-Way Handshake are redundant with respect to the key ordering properties of the Group Key Handshake as well. This is intuitively true because the Group Key stage uses key material established by the 4-Way stage, thus guaranteeing that actions of the Group Key protocol are done after actions of the 4-Way.

As in the case of $\Gamma_{tls,2}$, $\Gamma_{gk,2}$ states that the supplicants and the authenticator should not decrypt a message with the PTK, and send out the result of the decryption. This could conceivably happen if the data protection protocol uses the same

$$
\begin{array}{lcl}
\theta_{gk} & := & \mathsf{Has}(\hat{X}, ptk) \wedge \mathsf{Has}(\hat{Y}, ptk) \wedge (\mathsf{Has}(\hat{Z}, ptk) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{Y}) \wedge \\
 & & (\mathsf{Send}(X, m) \wedge \mathsf{Contains}(m, SeqNo) \supset \\
 & & \quad \mathsf{isLess}(SeqNo, Succ(CurrSeqNo))) \\
\Gamma_{gk,1} & := & \mathsf{Honest}(\hat{X}) \wedge (\mathsf{Send}(X, \hat{X}, \hat{Y}, m') < \mathsf{Send}(X, \hat{X}, \hat{Y}, m'')) \wedge \\
 & & \mathsf{Contains}(m', SeqNo1) \wedge \mathsf{Contains}(m'', SeqNo2) \supset \\
 & & \quad \mathsf{isLess}(SeqNo1, SeqNo2) \\
\Gamma_{gk,2} & \equiv & \Gamma_{tls,2} \\
\Gamma_{gk,3} & := & \mathsf{Honest}(\hat{X}) \wedge \mathsf{Send}(X, GrpMessage\ 1) \supset \neg\mathsf{Send}(X, GrpMessage\ 2)
\end{array}
$$

Table 4.6: Group-Key Protocol Precondition and Invariants

encryption key as the Group Key Handshake. Fortunately in 802.11i the PTK is divided into several parts, where the GTK is encrypted by KEK and the data confidentiality protocol uses a different part TK (Temporal Key). Our analysis highlights the need for the key hierarchy in 802.11i. Obviously, the *key secrecy* property fails if any member of the group reveals the PMK, PTK, or GTK.

Like the 4-Way Handshake, there is a restriction on a principal to not play both the authenticator and the supplicant roles. This is represented by the formula $\Gamma_{gk,3}$. This is an assumption about the operating environment which should be carefully considered in implementations.

4.5 Composition

The components of 802.11i are intended to be used in a specific order: TLS, 4-Way Handshake, Group Key Handshake, and the data confidentiality protocols. Since this is an example of sequential composition of protocols, previously proved composition theorems [28, 27], allow us to prove the correctness of the composite protocol by combining the independent proofs. However, 802.11i provides a failure-recovery strategy that reacts to failures and timeouts by restarting execution at the beginning. Furthermore, more efficient error-handling strategies have also been proposed (see Figure 4.1). Finally, 802.11i allows modes of operation that use Pre-Shared Keys instead of the TLS stage or reuse previously cached PMK's. The previously published

sequential composition theorems do not cover these more complicated situations.

In this section, we prove that the various components of 802.11i compose safely, for a general class of failure-recovery mechanisms, and for the modes of operation mentioned above. Section 4.5.1 is a technical section that introduces a new composition theorem and may be safely skipped by readers primarily interested in 802.11i. Section 4.5.2 applies this theorem to assert the correctness of 802.11i.

4.5.1 Staged Composition

The sequential composition theorem [28, 27] allows us to combine the proofs of sub-protocols into a proof of a composed protocol. However, sequential composition only works for control flows where the components execute one after the other in sequence. An examination of the control flow graphs in Figure 4.1 indicates that the intended sequence of execution of the sub-protocols forms a chain and the error-handling strategy introduces a set of backward arcs. This makes the sequential composition theorem stated in [28, 27] inadequate for our purpose. We introduce the notion of Staged Composition that extends sequential composition to handle control flow graphs with an arbitrary set of backward edges. This is formally stated as Theorem 5.

Recall that a protocol is a set of roles. For instance, the 4-Way Handshake consists of two roles - the authenticator and the supplicant. Each role is a sequence of protocol steps (`send` , `receive` , `new` and `match` actions), possibly depending on an input parameter list and providing an output parameter list. In a correct execution, we expect the role to run to completion. However, in reality the execution might be interrupted due to unexpected failures. Therefore, we partition a role into atomic steps, during each of which the execution does not block and cannot be interrupted. Since only `receive` actions require one thread to wait for another, roles are broken into atomic steps at receive points. The following definition captures the intuition that a protocol role may terminate at the end of any of its atomic protocol steps.

Definition 4.5.1 (Role-Prefix)
The set of role-prefixes of a role R is defined as $RolePref(R) :=$
$\bigcup_{0\leq i\leq k}\{(\overrightarrow{x})[b_0\ldots b_i]_X\langle\overrightarrow{x}\rangle\}\bigcup\{(\overrightarrow{x})[]_X\langle\overrightarrow{x}\rangle\}\bigcup\{R\}$, *where role* $R = (\overrightarrow{x})[b_0\ldots b_k]_X\langle\overrightarrow{y}\rangle$.

A role in the composed protocol corresponds to a possible execution path in the control flow graph. A backward edge in a control flow graph from role R_i to role R_j causes control to proceed from the end of an atomic protocol step of R_i to the beginning of R_j. The following definition captures the set of possible executions of a role in a composed protocol.

Definition 4.5.2 (Staged Role)
The set of staged roles $RComp(\langle R_1, R_2 \dots R_n \rangle)$ *is defined as the sequential composition* $r_{a_1}; r_{a_2}; \dots; r_{a_l}$, *where* $a_k \in \{1, 2, \dots, n\}$, $r_{a_k} \in RolePref(R_{a_k})$, *with* $a_1 = 1$. *Furthermore, if* $r_{a_j} = R_{a_j}$ *then* $a_{j+1} = a_j + 1$; *otherwise* $a_{j+1} \leq a_j$.

Given a chain of roles $\langle R_1, R_2 \dots R_n \rangle$, execution always starts at R_1; progress down the chain happens on successful completion of a role, with error-handling causing control to flow to the beginning of an earlier block in the chain. Failure of a role R_{ak} can happen at multiple points as characterized by the set $RolePref(R_{a_k})$ defined above. Finally, a protocol is simply a set of roles.

Definition 4.5.3 (Staged Composition)
A protocol Q *is in the set of staged compositions* $SComp(\langle Q_1, Q_2 ... Q_n \rangle)$ *of sub-protocols* $Q_1, Q_2 ... Q_n$, *if each role of* Q *is in* $RComp(\langle R_1, R_2 ... R_n \rangle)$, *where* R_i *is a role of protocol* Q_i.

Theorem 5 below states the conditions that need to hold for the composite protocol to be secure under staged composition. The first condition states that each protocol guarantees certain security properties assuming certain invariants are true. This follows from the correctness proofs of individual components. The second condition captures the intuition that the protocols running in the system do not cause vulnerabilities in each other. The third condition ensures that the precondition of a role R_i is discharged by the postcondition of R_{i-1}, which is required for sequential composition without error flows. Finally, the fourth condition requires that the precondition of a sub-protocol are preserved by the protocols steps of all the sub-protocols later in the chain. This ensures secure composition in the presence of arbitrary error handling mechanisms.

Theorem 5 (Staged Composition Theorem)
Given protocols $Q_1, Q_2 ... Q_n$, *if*

$$
\begin{aligned}
&(i) \quad \forall i,\ \Gamma_i \vdash \theta_i[P_i]_X \varphi_i \\
&(ii) \quad \forall i, j, \quad Q_i \vdash \Gamma_j \\
&(iii) \ \forall i,\ \varphi_i \supset \theta_{i+1} \\
&(iv) \quad \forall B \in \bigcup_{j \geq i} ProtocolSteps(Q_j),\ \theta_i[B]\theta_i
\end{aligned}
$$

then $SComp(\langle Q_1, Q_2 ... Q_n \rangle) \vdash \theta_1[P; P_i]_X \phi_i$, *where* $P; P_i \in SComp(\langle Q_1, Q_2, \ldots, Q_n \rangle)$ *and* $P_i \in Q_i$.

Intuitively, we treat branches introduced by the error handling flows as non-deterministic choices, which reduces staged composition to the previously studied notion of sequential composition [28, 27]. The set defined in Definition 4.5.2 represents all possible linearizations of the executions of a role in a composed protocol. We omit the proof of Theorem 5 due to space constraints.

4.5.2 Composition of 802.11i

We now apply the compositional proof method to the 802.11i protocol components by showing that the four conditions required by Theorem 5 are satisfied.

1. We start with the properties of the components proved independently in sections 4.2, 4.3, 4.4. This corresponds to asserting condition (i) from Theorem 5.

$$
\begin{aligned}
&TLS \vdash \Gamma_{tls,1} \wedge \Gamma_{tls,2} \\
&\Gamma_{tls,1} \wedge \Gamma_{tls,2} \vdash [TLS{:}Server]\phi_{4way,auth} \wedge \phi_{4way,sec} \\
&4WAY \vdash \Gamma_{4way,1} \\
&\Gamma_{4way,1} \wedge \Gamma_{4way,2} \vdash \theta_{4way}[4WAY{:}AUTH]\phi_{4way,sec} \wedge \phi_{4way,auth} \\
&GK \vdash \Gamma_{gk,1} \wedge \Gamma_{gk,2} \\
&\Gamma_{gk,1} \wedge \Gamma_{gk,2} \wedge \Gamma_{gk,3} \vdash \theta_{gk}[GK{:}AUTH]\phi_{gk,sec} \wedge \phi_{gk,ord}
\end{aligned}
$$

2. Next we prove that each component respects the invariants of the other components. This is done by induction over initial segments of the roles of the components. These conditions hold because components use different keys, and use MICs (Message Integrity Codes) that contain sufficient identity to be distinguished from each other.

$$\begin{aligned} TLS &\vdash \Gamma_{4way} \wedge \Gamma_{gk} \\ 4WAY &\vdash \Gamma_{tls} \wedge \Gamma_{gk} \\ GK &\vdash \Gamma_{tls} \wedge \Gamma_{4way} \end{aligned}$$

3. We prove that the precondition of the 4-Way authenticator role is implied by the postcondition of the TLS server role and the precondition of the Group Key authenticator role is implied by the postcondition of 4-Way authenticator role. At this point, we have the safe sequential composition of the components of 802.11i.

$$\begin{aligned} \phi_{tls} &\supset \theta_{4way} \\ \phi_{4way} &\supset \theta_{gk} \end{aligned}$$

4. Finally, the precondition of the 4-Way Handshake, which involves not sending out the PMK, is preserved by all protocol steps of the 4-Way and the Group Key Handshake, allowing restarts of the 4-Way Handshake to be secure. Similarly, the protocol steps of the Group Key Handshake do not send out the PTK ensuring restarts of the Group Key Handshake are secure.

The steps explained above allow us to apply Theorem 5, and prove the 802.11i authenticator guarantee. A similar process can be applied to the supplicant. Theorem 6 states the result of our analysis.

Theorem 6 (802.11i Guarantee)
(i) The security properties of the components listed in sections 4.2.2, 4.3.2, 4.4.2 are guaranteed in all modes of IEEE 802.11i, if the assumptions in Table 4.2, 4.4, 4.6 are satisfied.
(ii) Suppose no principals play both the authenticator and supplicant role, 802.11i satisfies assumptions in Table 4.2, 4.4, 4.6.

Note that most of our effort was on proving individual components correct. Steps 2, 3, and 4 took considerably less effort. This shows that it is convenient to apply the compositional method to the 802.11i protocol suite. Furthermore, our proof implies the correctness of other deployment modes in 802.11i, which are different from that in section 4.1.1. First, 802.11i may be run without the TLS stage, using Pre-Shared Keys (PSK) instead. When a PSK is shared by an authenticator-supplicant pair, the precondition requirement of the 4-Way Handshake is satisfied, which implies the safety of this mode. Second, supplicants may also run the 4-Way Handshake using a PMK cached from an earlier execution of the composite protocol. This corresponds to restarting execution from the 4-Way Handshake stage. Since our proof holds for an arbitrary set of such backward arcs, using a cached PMK is safe.

4.6 Conclusions

We present a formal correctness proof for the IEEE 802.11i and TLS protocols using Protocol Composition Logic (PCL). The proof of IEEE 802.11i and TLS supports many design decisions of the IEEE 802.11i TGi committee and reinforces conclusions of our previous studies [40, 41]. For example, the PCL proof demonstrates the need for separate keys for supplicant and authenticator to prevent a reflection attack, supports our previous intuition that various sequence numbers were unnecessary, shows that the protocol remains secure when a nonce-reuse mechanism is adopted in the 4-Way Handshake to reduce a vulnerability of Denial of Service [40, 41], and supports the adoption of optimized error-recovery strategies. Developing a single correctness proof for a class of failure-recovery strategies requires a new composition principle for PCL, which is formulated and proved semantically sound in this chapter.

The compositional nature of our protocol logic distinguishes our effort from other methods with similar foundations, such as Paulson's Inductive method [84] and Meadows' NRL protocol analyzer [66], which have been applied to protocols of similar scale and structure. In fact, Meadows previously identified composability of cryptographic protocols as a significant concern in establishing correctness [67]. A compositional approach is useful both for managing scale when working with a large protocol and in understanding how a single protocol interacts with its environment. In order to achieve compositionality, each individual proof involves a set of protocol invariants that must be satisfied in any environment where the protocol runs. These conditions must not only be satisfied by other subprotocols of 802.11i, but also by other protocols using the same certificates or other critical data which are run in parallel with 802.11i. Our proof therefore provides useful deployment information beyond the correctness of the protocols.

Our high-level logic with provable soundness over arbitrary symbolic runs is intended to combine the relative readability and ease of use of BAN-style logics [18] while being based on the semantic protocol execution model of Paulson's method [84]. Although we constructed all proofs manually, the proof system is completely rigorous and amenable to future automation. For more details about the PCL proof system and the actual proofs described in this chapter, we refer the reader to our Protocol Composition Logic web site. Finally, the axioms and rules used in the current proof have been proved sound for a symbolic model of protocol execution and attack. We hope that in the future a computational semantics of PCL, such as suggested in [24], can be developed for the entire proof system used here, providing a correctness proof of 802.11i under standard cryptographic assumptions.

Chapter 5

Using 802.11i in Ad Hoc Routing

As shown in Chapter 4, 802.11i can provide a confidential link between two neighboring nodes. Because of its ratification as an IEEE standard, it may be widely deployed in wireless devices in the near future. However, the security analysis of Chapter 4 is specific to a WLAN system: in infrastructure mode, typically a laptop can communicate with an Access Point through wireless channel directly; in ad hoc mode, two wireless devices must be in the transmitting range of each other. In a wireless ad hoc network, two nodes might be multiple hops apart from each other; therefore, a secure routing protocol is required to provide a valid path between them.

In this chapter, we will study the routing security problem in wireless ad hoc networks with 802.11i deployed in each node. The chapter is organized as follows. Section 5.1 summaries the secure routing problem and proposes some basic ideas; Section 5.2 describes the original and our improved DSR protocol; Section 5.3 analyzes the security of our improved DSR protocol under different situations; Section 5.4 discusses Denial of Service attacks; Section 5.5 concludes our work.

5.1 Secure Routing

Secure Routing has been extensively explored in recent years [46, 49, 44, 80, 91, 106, 81, 12, 104, 105, 43, 45, 20, 17, 102, 60, 63, 59]. Most of the designs have the following characteristics. First, they assume a strong key management scheme in which each

node has a shared key with every other node, or a Public Key Infrastructure is implemented. In either case, an individual node has global knowledge about the whole network and is able to validate the messages generated by all the other nodes. However, this is not the natural scenario for wireless ad hoc networks since it is difficult to have a global key management and it is even impossible to know the exact nodes present in the system prior to implementation. There have been a few studies on wireless ad hoc networks without global key management [20]; however, prior work focuses on searching for a trust path from an incomplete set of trust relationships. In our work, we reduce the requirement for a key management scheme from global to local; that is, each node has a shared key with only its neighbors, plus the source and target node share a key before communication. It is possible that this is the minimum requirement for key management in order to build a secure wireless ad hoc network.

Current secure routing protocols assume a global attacker; i.e., an attacker that can insert, delete, and intercept messages everywhere in the network simultaneously. This assumption is reasonable if the routing protocol runs on top of a completely plain network without any other secure mechanisms implemented. Since all messages in the system are unprotected, they are open to the attackers everywhere. Preventing attacks by this kind of global attacker is very difficult, which makes a routing protocol very complicated. However, in reality this attacker model might be too strong to be practical. Sometimes it is hard for an attacker to have a global control on the network due to physical or geographic limitations. Most of the potential attacks happen locally, when one or several of the legitimate nodes are compromised, or an attacker introduces its own malicious nodes, or a legitimate node acts in a bad way. Furthermore, when security is required in the system, it is very possible that other kinds of mechanisms are already implemented before routing; for example, 802.11i has been deployed for link layer security. Therefore, in this chapter, we restrict the attacker to a local model, which makes the secure routing problem much easier.

Based on these considerations, we investigate secure routing in a different way. We first build a secure link between neighbors, then build a secure routing protocol on top of the confidential links. This idea adheres to the basic divide-and-conquer paradigm in computer science and satisfies the natural property of a bottom-up construction for

wireless ad hoc networks. Instead of global key management, every node only needs key material to maintain the security association between itself and the neighbors, which results in scalable key management. In other words, the network can be built from a small group of nodes, and grow gradually by adding more and more neighbors. Furthermore, the establishment of confidential links between neighbors automatically restricts an attacker to be local even if it compromises a legitimate node since the messages appearing in one link are not globally understandable.

In this chapter, we show that our methodology significantly reduces the complexity of designing a secure routing protocol, using Dynamic Source Routing (DSR) as an example. First, there are numerous approaches to provide a security association between neighbors, from which we can take great advantage. For example, as we have proved in Chapter 4, IEEE 802.11i can provide a secure link between two neighboring nodes. The wide deployment of 802.11i means that we already have a security association between neighbors. Second, running on top of the confidential links, the routing protocol only needs to defend against local attackers, which apparently reduces the design complexity.

5.2 Dynamic Source Routing with 802.11i

5.2.1 The Original DSR

Dynamic Source Routing (DSR) [56] is a well-known routing protocol in wireless ad hoc networks. The original DSR consists of *Route Discovery* and *Route Maintenance* mechanisms. In this study we only consider the *Route Discovery* process, which is shown in Figure 5.1. Whenever a source node S wants to communicate with a target node T, S will initialize a Route Discovery by sending out a Route Request (RREQ) packet. The RREQ packet contains a tuple of ⟨SID,TID,REQ-ID⟩ as a unique indicator, representing the source and target addresses, and a request id, respectively. The intermediate node receiving this request will check whether it has received the same request before; if it has, the request is dropped silently, otherwise the node will add itself to the route list and re-broadcast the request to its neighbors.

Upon receiving the request, the target node sends out a Route Reply (RREP), which contains the complete routing information, to the source along the reversed path.

Note that this is only the basic operation in DSR; there are many existing variants for optimizing performance and handling general scenarios. For example, there might exist uni-directional links in the system, therefore, the target will send out the reply along the path it already knows or initialize a Route Request to find a path to the source; an intermediate node could send out a reply to the source directly if it knows a route to the target in its cache; any node hearing a route could cache this overheard routing information for performance improvements; furthermore, when one link fails, a Route Error message is generated and propagated in the network to update the routing information, which might be frequent and necessary in a highly mobile environment.

In the following sections, we restricted our attention to fixed wireless ad hoc networks with the simplified DSR protocol. We assume that all links are bi-directional, only the target will send out the reply to the source, the Route Reply is traveling along the reversed route, and there is no link failures. As shown in Figure 5.1, node S initializes a Route Discovery by sending out a Route Request message. Then, the request floods the whole network and two possible routes arrive at the target. Finally, the target will send the routes back to the source along the reversed paths.

5.2.2 The Improved DSR

Obviously the original DSR protocol is vulnerable since there are no security mechanisms implemented; previous studies [49, 80, 43, 45, 48, 47] have disclosed many attacks, like rushing, wormhole, flooding, and so on. In this section we improve the protocol to eliminate these attacks, under the assumption of a key management scheme as follows. Each node who wants to join the network must share a secure association with each neighbor. Here a secure association means that the nodes either share a symmetric key or are able to verify each other's public key, in order to build a confidential channel between them. Furthermore, the source and target nodes of any transmission must also share a secure association; otherwise there is no way to

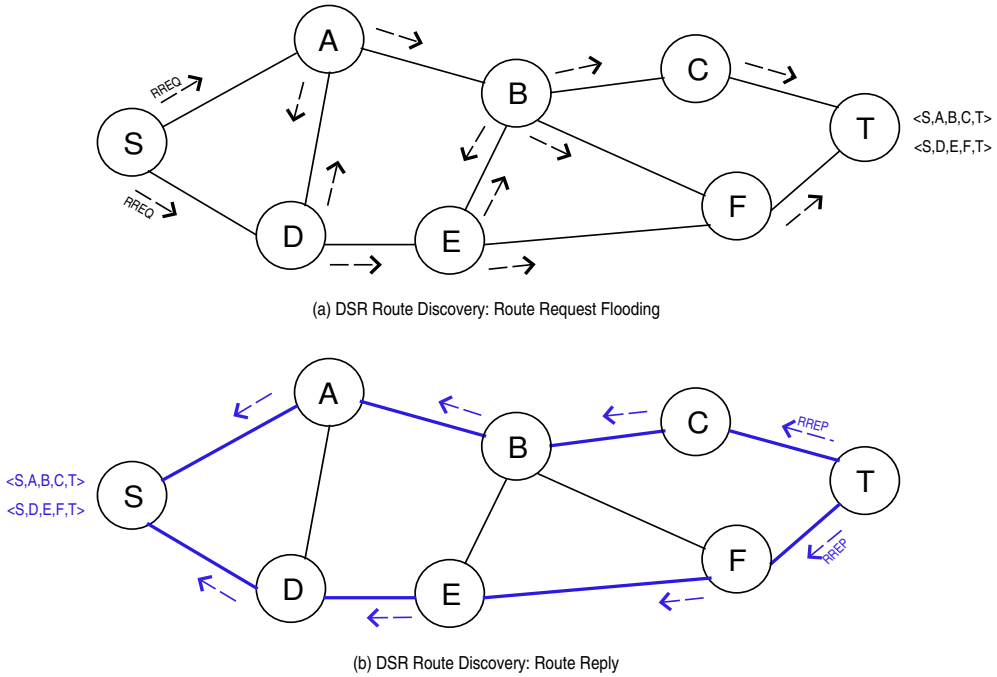

Figure 5.1: DSR route discovery

provide end-to-end security for them. Based on this key management assumption, we decompose the secure routing problem into two components: a hop-by-hop link security protocol and a routing protocol on top of the secure links. The hop-by-hop link security protocol is not our focus here, since there are already proposals to build a secure channel between nodes given a secure association (i.e., 802.11i). Instead, we focus on building a secure routing protocol on top of the network with confidential links. Note that in a routing protocol, every node (re)broadcasts a message to all its neighbors simultaneously. For simplicity we assume that we are able to distinguish the sender and recipients during broadcasting; for example, the broadcast messages can be sent through multiple point-to-point links, or there are some cryptographic mechanisms implemented. Based on these assumptions, we show that we can achieve secure routing by simply improving DSR in several aspects, which are summarized as follows.

(i) When composing the Route Reply message, the target attaches a Message Integrity Check (MIC) to the Route Reply, which is computed from the shared secure association and the message content; when the source gets the reply, it checks the authenticity of the message before accepting it.

(ii) When rebroadcasting the Route Request message, every intermediate node checks that the last hop in the route list matches the identity of its neighbor, from which the request is transmitted.

(iii) The intermediate nodes remember the route associated with the first Route Request received from the source; when forwarding the Route Reply back to the source, it checks that the route list between the source and itself matches the saved entry.

(iv) The source uses a cryptographical random number generator, instead of a monotonically increasing sequence counter, to generate the Route Request ID.

(v) The target sends back Route Reply messages, resulting in multiple routes, for all the Route Request messages received; the source chooses one from these multiple routes for subsequent data communications.

Note that the last improvement has been explored in [17, 60] for performance considerations. Here we introduce it to mitigate the possible Denial of Service (DoS) vulnerabilities. In the following sections, we will verify that the improved DSR protocol is secure on top of the confidential links.

5.3 Security Analysis

Since each node in the system authenticates its neighbors and establishes confidential channels with them, any malicious node without a valid secure association cannot enter the system. Without considering selfish behavior, if every legitimate node acts honestly, the original DSR routing protocol running on top of the confidential links can prevents malicious nodes from entering the system, thus, provides intended routing logic appropriately. However, the security is completely broken if one or several nodes are compromised, or the legitimate nodes act dishonestly. In other words, the routing is vulnerable to insiders, which requires the improvements as proposed in section 5.2.2. In this section, we evaluate the security of the improved DSR protocol under inside attackers.

5.3.1 DSR Property

We limit our interest to wireless ad hoc networks with fixed nodes, in which every node shares a confidential channel with its neighbor, but one legitimate node is compromised (all key materials are disclosed). The objective of the routing protocol is to find a *Route* from S to T, denoted as $R_{S,T}$, which is an ordered list of nodes starting from S and ending with T. A *Subroute* represents any subset of the ordered lists. We define the *Route Length* $|R_{S,T}|$ as the elapsed time when the route request message travels from the starting node S to the ending node T. In our analysis, we assume that every node processes the Route Request in a constant time, therefore, the*Route Length* can be simply represented by the number of hops in a *Route*, following the term of *hop* which represents the link from one node to another. However, it is worth

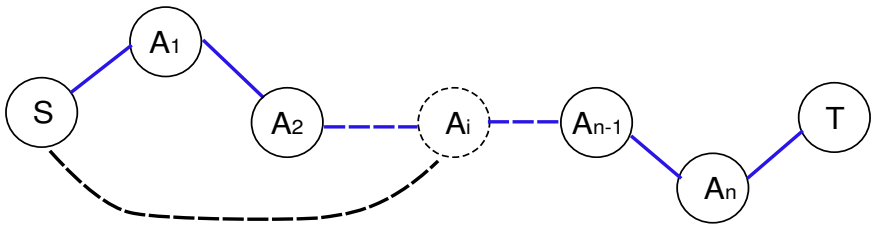

Figure 5.2: DSR Shortest Path Property

noting that in complicated scenarios, different nodes in the network might have different traffic load and introduce various processing time, in which case the *Route Length* is a weighted sum of the hops indicating the total processing time. Furthermore, *Route Length* is time-variant according to the traffic variations in practice, but our definition is meaningful in a specific time slot or on an average sense.

Since the target node responses to every Route Request it receives, there might be multiple routes returned to the source. However, since every intermediate node drops the Route Request that has been processed by itself before, there must be only one subroute from the source to any intermediate node; furthermore, all subroutes must be the shortest ones in term of the *Route Length* we defined earlier. These observations are formulated in the following theorem.

Theorem 7 (DSR Shortest Path Property)
Let $R_{S,T}$ denote the set of routes returned by the target, R_{S,A_i}, $A_i \in R_{S,T}$ denote the set of possible subroutes from S to A_i in $R_{S,T}$, n_{S,A_i} denote the number of elements in R_{S,A_i}, l_{S,A_i} denote the length of the shortest path from S to A_i, $|R_{S,A_i}|$ denote the length of the route, then the following properties hold for each $A_i \in R_{S,T}$ if all nodes are honest: (i) $|R_{S,A_i}| = l_{S,A_i}$; (ii) $n_{S,A_i} = 1$.

We can prove the theorem easily by contradiction. First, assume $|R_{S,A_i}| > l_{S,A_i}$, which means R_{S,A_i} is not the shortest path. Denote the shortest path as the dashed line shown in Figure 5.2. A_i will get the first Route Request from this shortest path, and drop the same requests from other paths, therefore, R_{S,A_i} cannot be returned.

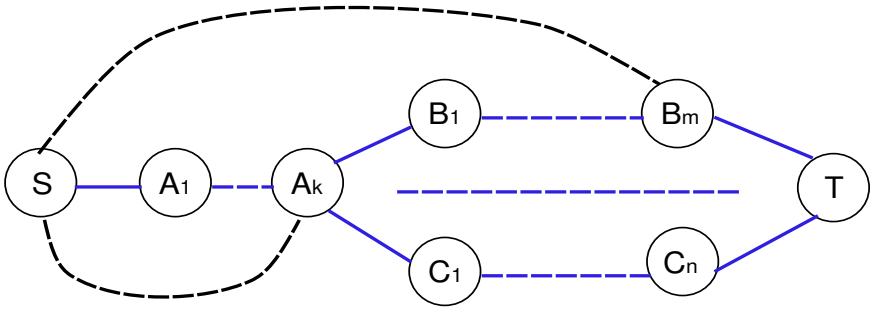

Figure 5.3: The set of routes DSR can return

Second, suppose that there are more than one subroutes returned for some intermediate node A_i. Without loss of generality, we can list two of them as $(S, A_1, A_2, ..., A_m, A_i)$ and $(S, B_1, B_2, ..., B_n, A_i)$. If m and n are different, A_i will drop the request from the longer path, so $m = n$. Since these two routes are different, there must exist at least one different node between all A's and B's. Assume A_m and B_m are different, then A_i will drop one of the message; therefore, A_m and B_m must be the same. Similarly, if A_{m-1} and B_{m-1} are different, then A_m will drop one of the message, and so on. Hence, these two routes are completely the same.

The reasoning above holds if all nodes are honest in the system, which means that the set of returned routes can be described as in Figure 5.3. Any other routes shown as dashed lines are impossible to present in the returned routes.

5.3.2 One Malicious Node

We have shown that the DSR protocol satisfies the shortest path property if all nodes are honest. However, when one or more nodes are compromised, they might not obey the routing protocol. For example, they can maliciously drop the routing request/reply packet, alter the packet content or construct the forged packet. Therefore, our routing protocol aims to defend against these kinds of attacks.

Since the target node uses the shared key to protect the Route Reply, it is impossible for any intermediate node to modify the route after the Route Request is received

and the Route Reply is returned by the target node. Furthermore, if a compromised node is simply dropping the routing messages, the only result is that it will exclude itself from serving as a relay. Since multiple paths are returned, the source and target can still communicate to each other by switching to other possible routes. We do not consider this as a severe vulnerability. If the malicious node is in a significant position and all routes from S to T must pass the node, this node can construct a Denial of Service attack by simply drop all the messages. Therefore, in any case the communication between S and T can be interrupted if such an intermediate node is compromised. Hence, we focus on the situations that a compromised node can maliciously return a valid route to the source and attract the traffic passing through itself, based on which it gains access to the traffic and might launch other severe attacks.

Furthermore, when a confidential link is established between neighbors, the compromised nodes are restricted to the local attacker model, which means it can only touch the links with its neighbors. Consider that the Route Request id is generated by the source randomly, the malicious node can only know the Route Request after the request has arrived at those links. Then the malicious node can modify the packet and forward to neighbors. Since other nodes are not controlled by the malicious node, it is possible that the modified route from the malicious node does not arrive at the target node, which means it is impossible for the malicious node to affect the returned route. However, it is very difficult to explore whether these route requests can arrive at the target or not; instead, we reason about the properties of the returned route similarly as in section 5.3.1.

Theorem 8 (One Malicious Node)
Suppose a route containing a compromised node E is returned, denoted as $(S, A_1, ..., A_m, E, B_1, ..., B_n, T)$, the following statements hold if other nodes in the system are honest.
(i) This route must exist in the system;
(ii) For each B_i, $|R_{E,B_i}| + l_{S,E} = l_{S,B_i}$;
(iii) For each A_i, $|R_{S,A_i}| = l_{S,A_i}$.

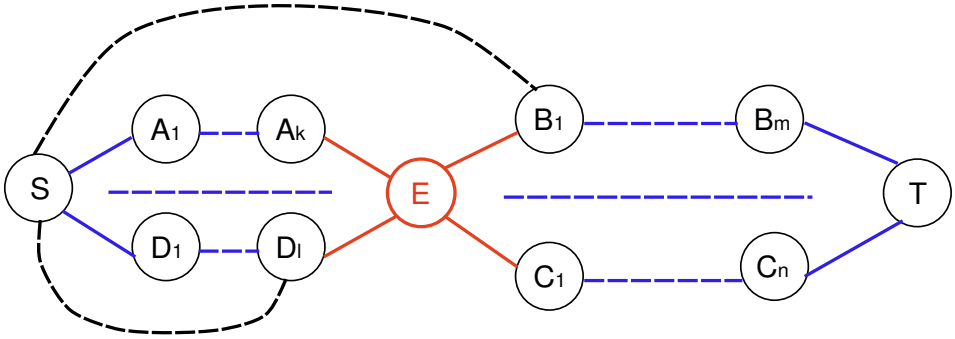

Figure 5.4: The set of returned routes with one malicious node

We can prove above theorem similarly by contradiction; the details are omitted here. This theorem states that a malicious node cannot introduce more vulnerabilities into the system, the only thing it can do is to introduce a set of paths from S to E, instead of only one, as shown in Figure 5.4.

Note that a rushing attack [48] has been identified on the ad hoc networks, in which a malicious node can forward the Route Request faster than other nodes, thus, cause the neighboring nodes to suppress the Route Requests from other nodes. In our study, since we assume that each node will process the request in a constant time, there is no rushing attack possible. If the attacker can cause the compromised node to forward packets sooner than other nodes, it is still able to launch a rushing attack. However, our results still hold, with a slight modification that the malicious node can reduce the length of its route by 1.

5.3.3 Multiple Compromised Nodes

In the above section, there is only one compromised node in the system; however, in practice, it is possible for an attacker to compromise more nodes and control them all together through some covert channels. Similarly we formulate this situation in Theorem 9, which states that a set of malicious nodes can only introduce a set of routes as shown in Figure 5.5.

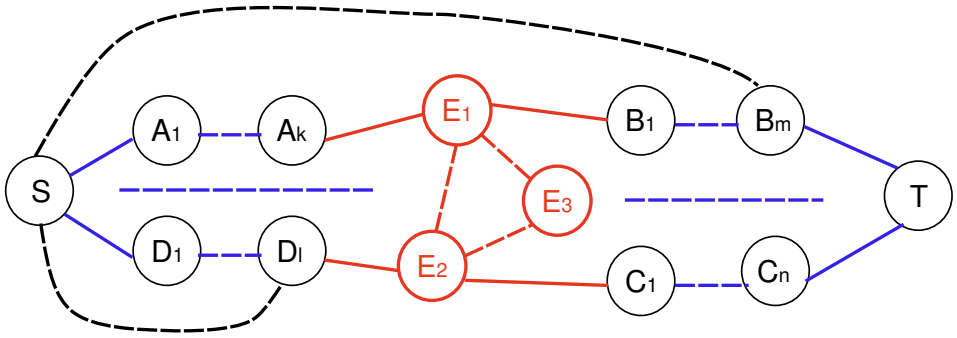

Figure 5.5: The set of returned routes with multiple malicious nodes

Theorem 9 (multiple malicious nodes)
Suppose the returned routes containing a set of compromised nodes $E_1, E_2, ..., E_n$*, denoted as* $(S, A_1, ..., A_k, E_1, ..., E_n, B_1, ..., B_m, T)$*, the following statements hold if other nodes in the system are honest.*
(i) R_{S,A_i} *and* $R_{B_i,T}$ *must exist in the system;*
(ii) For each B_i, $|R_{E_n,B_i}| + l_{S,E_1} \leq l_{S,B_i}$*;*
(iii) For each A_i, $|R_{S,A_i}| = l_{S,A_i}$.

Note that the worm hole attacks [47] are still possible if more than one nodes are compromised; furthermore, the colluded nodes can modify the route between them to an arbitrary list. However, Theorem 9 states that it does not affect the routes which do not pass through the compromised nodes, since they do not share any common nodes after the last malicious node. On the other hand, if a route has to pass through the compromised nodes, there are no differences between the converted channel and the real existing channel. In order to eliminate this attack, it is necessary to adopt other mechanisms [47] beyond entity authentication only.

5.4 Availability

In the original DSR protocol, in order to reduce the Route Request flooding in the system, a duplicate suppression mechanism is adopted. However, this causes an easy

DoS attack on any source node. The malicious node can simply flood the network by pretending to forward a Route Request for a source with a predicted Request ID, then all nodes in the system will drop the subsequent Route Requests from the real source. Therefore, in our improved DSR, we use a random number as the request id. Since nobody else can guess the id correctly until it receives the route request, this attack is avoided.

But for each node to avoid forwarding duplicated Route Request again and again, they need a cache to store the received request id's from each source. Then they can check whether they have processed the same request upon receiving another one. If they keep all received id's in cache, obviously there is a memory exhaustion attack. On the other hand, if they only remember the last received request id, the malicious node can forge a forwarded message with a random id to refresh this one-entry cache and disable the duplicate suppression mechanism; If the attacker periodically does this, the request message from the real source will be re-broadcast forever. This is a tradeoff. Fortunately, since the route discovery stage will complete in a relatively short period, every node only needs to store the received request id's from the same source for a short period of time. Whenever it receives a request from the source, it will check the cache to see whether the request has arrived before. If no, the node add itself to the route and re-broadcast it; otherwise the request is dropped silently.

The compromised node can also periodically send out route requests as a legitimate source to flood the whole network and cause traffic congestions. Since we assume that link layer security provides strong authentication, it is impossible for the malicious user to forge the identity of other nodes; however, it can still send out route requests using its real identity. In order to mitigate this attack, each node should rate limit the requests because the route discovery re-initialization is not expected so frequently in the network. For example, every node will only allow r requests per period from each link, and drop the subsequent requests after r trials. However, there are still problems since the rate will add up when a node has more than one neighbors, as shown in Figure 5.6. In this case the node should cache the requests, and select the requests to forward in a round-robin way.

Based on these discussions, we adopt the following mechanism. Every node limits

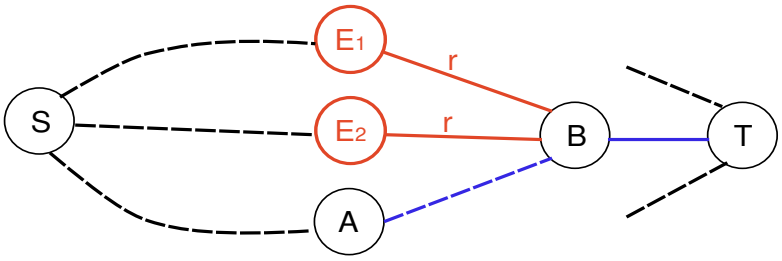

Figure 5.6: The rate limit problem

the rate of requests from neighboring links if the requests are originated from that node, i.e., the rate to generate new requests should be less than r. Furthermore, every node should cache the received request id's in order to enable the duplicate suppression mechanism. Assume there are c nodes compromised in the system, the cache size should be greater than $c * r$. The reasons are described as follows. Since there are c nodes in the system compromised, they can generate at most $c * r$ new request id's pretend to start from the same source, so every node have enough cache to store the true request id coming from the real source and suppress the duplicated messages later. Note that in a common system, r could be very small since the source is only expected to perform several tries to find a route, i.e., $r = 3$. So the cache size is affordable for the node even if it stores entries for every node in the system, totally $3 * c * n^2$.

5.5 Conclusions

In this chapter, we revisit the secure routing problem for wireless ad hoc networks. Secure routing is decomposed into two parts - a hop-by-hop authentication and a routing on top of the confidential links, which reduces the complexity of designing secure routing protocols. Hop-by-hop authentication can be achieved when IEEE 802.11i Standard is widely deployed in wireless devices. Furthermore, we have shown that the common DSR with slight improvements can provide secure routing on top of confidential links.

The intrisintic reason that link layer security helps secure routing is that the capacity of an attacker is restricted from a global control of the whole network to the local control of its neighboring links. In practice, even if the secure neighboring links are not guaranteed, the local attacker model can be enforced by geographic limitations, in which case the secure routing can be greatly simplified too. Furthermore, through this kind of decomposition, we reduce the key management requirements. In traditional security routing protocols, it is always assumed that every two nodes share a secure association; however, in our study, we only need the neighboring nodes share a secure association to build secure links between them, plus that the source and target share a secure association to build a secure end-to-end channel between them. Obviously this is a weaker requirement. Finally, although our work deals with the secure routing in wireless ad hoc networks, the similar methodology can be adopted to develop routing algorithms for other networking scenarios (e.g., BGP).

Chapter 6

Conclusions

6.1 Deployment Considerations

In proceeding through the analysis of 802.11i, we described several deployment considerations and suggested ways to avoid a range of security problems. Here is a summarized list of our recommendations for deploying 802.11i.

In order to provide **data confidentiality and integrity**:

1. The 802.11 data session should be encrypted under a shared key derived from the authentication session. Preferably CCMP should be chosen whenever possible, because WEP is completely insecure and TKIP has some weaknesses in the key scheduling algorithm. Furthermore, in case that CCMP is not available, WEP or TKIP should still be enabled to prevent occasional intruders.

In order to provide **mutual authentication**:

2. When Pre-RSNA and RSNA connections are allowed simultaneously in a hybrid WLAN configuration, the client interface should provide a user the chance to manually decide which to use prior to opening a connection; and the Access Points (APs) should impose different privilege policies for each security level. Otherwise, an adversary may degrade the security of the entire system to the lowest level through a Security Level Rollback Attack.

3. The EAP method should provide strong mutual authentication, defending against the dictionary attacks and Man-in-the-Middle attacks. EAP-TLS is the de facto standard and should be implemented whenever possible.

4. When EAP-TLS is implemented, it is very possible that the public/private key pair used in the TLS session is shared with other protocols running in the server side. In this case the server should be careful about decrypting a message using the private key and sending out the result during all concurrent protocols, since this sequence of actions might disclose the common secret achieved in the EAP-TLS session. This consideration is necessary for TLS protocol independently implemented in any systems.

5. In order to prevent a reflection attack on the 4-Way Handshake and Group Key Handshake, in the WLAN infrastructure mode, any participant should be restricted to only one role, either authenticator or supplicant; in the ad hoc mode, any participant have to play both roles, but different key materials should be used for different roles. Furthermore, if a Pre-Shared Key (PSK) is used for the 4-Way Handshake, the passphrase for PSK generation should be able to defend against dictionary attacks.

6. When the authenticator and the authentication server are implemented separately, the RADIUS protocol is responsible to provide a secure link between them. Therefore, the shared secret for RADIUS should be chosen carefully to defend against dictionary attacks.

In order to provide **availability**:

7. In the 4-Way Handshake, a supplicant should re-use the same nonce for all the *Message 1*s received from the authenticator until one instance of the handshake is completed successfully. With this improvement the supplicant does not need to store all possible nonces newly generated for every incoming *Message 1*, thus, eliminate the potential 4-Way Blocking Attack.

8. In the RSN IE confirmation mechanism during the 4-Way Handshake, the participants should only verify the authentication suite, then negotiate other security parameters using the authenticated RSN IE exchanged in the 4-Way Handshake. This relaxed verification condition can mitigate the potential DoS attacks.

9. When TKIP is adopted in a legacy device, the Michael MIC Failure countermeasure should be implemented with ceasing the communications instead of re-keying; furthermore, the TKIP Sequence Counter (TSC) should be updated before the message MIC is validated. These improvements can make the DoS attacks more difficult.

10. When the complete 802.11i handshakes are implemented, a number of EAP messages should be ignored to prevent the possible DoS attacks, i.e., EAPOL-Start, EAPOL-Success, EAPOL-Logoff, and EAPOL-Failure packets. Furthermore, the EAP identifier should be implemented separately for each association to adopt more than 255 association requests.

11. Since the management frames and control frames are unprotected, 802.11 networks are still vulnerable to the DoS attacks exploiting the virtual carrier sense mechanism and forging the Deauthentication/Disassociation frame. Deploying suitable validation policies for these frames can effectively reduce the vulnerabilities.

12. A set of failure recovery mechanisms could be used in an 802.11i deployment, each of which can guarantee the same security properties. The implementer should choose one according to the practical requirements. A good failure recovery can improve the efficiency and mitigate the DoS vulnerabilities.

6.2 Conclusions

In this dissertation, we analyzed the IEEE 802.11i Standard and explored its extension to wireless ad hoc networks. We identified several vulnerabilities in 802.11i and our

proposed repairs were adopted by the IEEE 802.11 TGi committee. Furthermore, we conducted a finite-state verification for the 4-Way Handshake, which showed the methodology for model checking security protocols, and illustrated the procedure to find vulnerabilities in the 4-Way Handshake. Moreover, we proved the correctness of the 802.11i protocols using a Protocol Composition Logic (PCL). Finally we revisited the secure routing problem in wireless ad hoc networks, under the assumption of 802.11i being implemented for Link Layer security. The following sections conclude our work in these aspects respectively.

6.2.1 IEEE 802.11i Security Analysis

In order to analyze the IEEE 802.11i protocols, the intuitive way is to manually examine all the possible attacks based on a thorough understanding of the protocols. Obviously this requires previous experience and expertise in security. We reviewed the details of 802.11i, and checked the protocols for data confidentiality, integrity, mutual authentication, and availability. Under the threats we consider, 802.11i is able to provide effective data confidentiality and integrity when CCMP is used. Furthermore, 802.11i is satisfactorily secure since a RSNA establishment procedure is adopted for mutual authentication and key management, although there might be several vulnerabilities in a practical implementation. However, since availability is not the primary design goal, 802.11i is still vulnerable to DoS attacks even if RSNA is implemented.

For mutual authentication, we found several potential problems in a WLAN system without careful considerations. First, if Pre-RSNA and RSNA algorithms are implemented simultaneously, an adversary is able to perform a Security Level Rollback Attack to force the communicating peers to use WEP, which is completely insecure. Second, if a wireless device is implemented to play the role of both the authenticator and the supplicant, an adversary can construct a reflection attack on the 4-Way Handshake, which could naturally appear in ad hoc networks. Furthermore, if the mutual authentication mechanism is not implemented appropriately, there might be a Man-in-the-Middle attack that reveals the shared secret; if a passphrase is used

to generate a 256-bit PSK, an adversary might be able to find the passphrase through dictionary attacks; an adversary is also able to discover the shared RADIUS secret through dictionary attacks.

For availability, we found more problems arising with 802.11i and proposed repairs accordingly. First, it is practical to launch a DoS attack easily on the Michael algorithm countermeasures. A significant improvement could be achieved by eliminating re-keying and updating the TSC carefully. Second, we described a new DoS attack through RSN IE poisoning and discussed several repairs. Relaxing the condition for RSN IE verification seems the preferred approach because it only requires minor modifications to the algorithm. Third, we found a DoS attack on the unprotected *Message 1* of the 4-Way Handshake and proposed the corresponding defense, which was adopted by the IEEE TGi committee. Furthermore, we discussed the tradeoffs in the failure-recovery strategy and suggested an efficient failure recovery, based on the characteristics of wireless networks. Finally, we integrated all the improvements to construct a DoS resistant variant of the 802.11i protocols.

6.2.2 Finite-State Verification

Beyond the intuitive approach, another powerful methodology is finite-state verification (or model checking), since it can help people to understand the protocol better and find some subtle problems which might be ignored by people. Using the 4-Way Handshake as an example, we showed the procedure for model checking protocols with a finite-state verification tool, called Murϕ. We identified the functionality of each field in the 4-Way Handshake messages, and proposed a simplified protocol that has the same authentication properties as the original one under our Murϕ model. Most significantly, we found an effective DoS attack on *Message 1* in the protocol, and proposed several simple solutions.

This attack exploits the vulnerability of unprotected *Message 1* in the 4-Way Handshake. In fact, the designers and reviewers have noticed that *Message 1* is vulnerable; however, the practical attack, which allows an attacker to block the handshake by simply inserting one forged message, is not apparent before our security

analysis. By intuition 802.11i is supposed to allow only one active 4-Way Handshake at any time, and generate a shared PTK between a corresponding supplicant and authenticator. However, we showed that the supplicant must allow multiple handshakes to execute in parallel, in order to ensure protocol completion in the presence of packet loss, which enables the attack. When 802.1X authentication is implemented, the attack could be more difficult due to the PMKID and link layer encryption, but the vulnerability still exists in the protocol.

Although it is not difficult to fix the problem, it is not obvious to find a solution with minor modifications on the protocol. We discussed three repairs - a random-drop queue, *Message 1* authentication, and reusing the nonce. A random-drop queue of a certain size can be implemented to avoid memory exhaustion attacks without any modifications to the protocol; however, it is not so effective even with reasonable queue sizes. *Message 1* authentication can resolve the vulnerability completely; however, it requires significant modification on the packet format. Our final solution, simply reusing the nonces in the supplicant until one legitimate 4-Way Handshake is completed, has no modifications to the protocol itself and eliminates the vulnerability inherently. It is even more efficient to re-use the nonces and store the received nonces and derived PTKs in a combination.

6.2.3 A Modular Correctness Proof

Although several vulnerabilities are identified, our analysis is incomplete because people could neglect some potential attacks, and finite-state verification is only capable of checking bounded number of participants and sessions due to physical limitations. In order to analyze the IEEE 802.11i protocols under unbounded number of participants and sessions, we adopted the Protocol Composition Logic (PCL) to conduct a correctness proof. The divide-and-conquer strategy is used to take advantage of the modularity of 802.11i.

For the security properties, we proved mutual authentication and key secrecy for individual components - TLS session, 4-Way Handshake and Group Key Handshake. We also analyzed the composibility of these components and concluded that all these

components compose securely. Since 802.11i has very complicated control flows when different failure recovery mechanisms are adopted, we developed a new composition theorem for staged composition. It follows that the components compose securely for a range of failure recovery control flows, including the improvements proposed in Chapter 2 and [41]. The general result also proves security for other configurations presented in the 802.11i specifications, including the use of a Pre-Shared Key (PSK) or cached Pair-wise Master Key (PMK). We also extended PCL to tackle memory associated with nonce reuse in the modified 4-Way Handshake and the global sequence counter in the Group Key Handshake.

For each component of 802.11i, we identified the required operating environments that other concurrent protocols in the system must satisfy to avoid destructive interaction. These insights are useful for making 802.11i implementation and deployment decisions, which demonstrate that PCL is suitable for compositional analysis of large protocols. Since TLS is widely deployed for a variety of purposes apart from 802.11i, these results for TLS also have independent interest.

6.2.4 Using 802.11i in Ad Hoc Routing

Since 802.11i is supposed to be widely implemented in wireless devices, we analyzed the influence of 802.11i on wireless ad hoc networks. In the current wireless ad hoc networks, most proposals for secure routing are very complicated, which might cause potential vulnerabilities due to inappropriate implementations. However, we showed that under the protection of 802.11i, it is very easy to achieve a secure routing mechanism by some simple improvements on the common DSR protocol.

Due to our modular correctness proof, 802.11i can authenticate the peers and provide a secure link between neighboring nodes. Therefore, if all nodes are honest, a common routing protocol, e.g. DSR, will work well to discover routing information. However, it is very possible that one or several nodes are compromised or dishonest, which destroys the whole routing security. We proposed a secure routing protocol with only slight modifications on the original DSR, and analyzed the security of the proposed protocol. Compared with designing a secure routing protocol on top of a

plain network, it appears to be much easier to achieve a secure routing protocol on top of a network with confidential links among neighbors.

Furthermore, our methodology is useful if any other link layer security mechanism is deployed instead of 802.11i. Actually this corresponds to the natural construction paradigm of wireless ad hoc networks, because the network can expand in a scalable way when the nodes in the system accept more and more neighbors.

Appendix A

Protocol Composition Logic

A.1 Programming Language

(names)	N	::=	$\hat{X}$	name
(session id)	S	::=	η	session id
(thread)	P	::=	$\langle N, S \rangle$	thread
(keys)	K	::=	K	key
			name(N)	name
(nonces)	n	::=	x	nonce
(numbers)	i	::=	i	natural number
			$succ(i)$	succ(i) > i
(terms)	t	::=	x	variable term
			nonce(n)	nonce
			na(N)	name
			thread(P)	thread
			key(K)	key
			$t, \ldots, t$	tuple of terms
			$ENC_K\{\|t\|\}$	encrypted term
			$HASH_K\{\|t\|\}$	keyed hash
			$SIG_K\{\|t\|\}$	signed term

(actions)	a	::=	`noop`	null action
			`send` t	send a term t
			`receive` x	receive into var x
			`new` x	generate new term
			`isLess` a, b	check that a $<$ b
			`match` t/t	match a term
(strands)	S	::=	$a; S \mid a$	
(cords)	C	::=	$N[S]$	

A.2 Protocol Logic

Action formulas

$$\begin{aligned} \mathsf{a} \quad ::= \quad & \mathsf{Send}(P,t) \mid \mathsf{Receive}(P,t) \mid \mathsf{New}(P,t) \mid \\ & \mathsf{Decrypt}(P,t) \mid \mathsf{Verify}(P,t) \end{aligned}$$

Formulas

$$\begin{aligned} \phi \quad ::= \quad & \mathsf{a} \mid \mathsf{a} < \mathsf{a} \mid \mathsf{Has}(P,t) \mid \mathsf{Fresh}(P,t) \mid \\ & \mathsf{FirstSend}(P,t,t') \mid \mathsf{Honest}(N) \mid \\ & \mathsf{Contains}(t_1,t_2) \mid \\ & \phi \wedge \phi \mid \neg\phi \mid \exists x.\phi \mid \mathsf{Start}(P) \end{aligned}$$

Modal formulas

$$\Psi \quad ::= \quad \{\phi,\ \rho,\ \phi\}$$

Here are the informal interpretations of the predicates:

$\mathsf{Has}(X, x)$ means principal $\hat{X}$ possesses information x in the thread X. This is "possess" in the limited sense of having either generated the data or received it in the clear or received it under encryption where the decryption key is known.

$\mathsf{Send}(X, m)$ means principal $\hat{X}$ sends message m in the thread X.

$\mathsf{Receive}(X, m)$, $\mathsf{New}(X, t)$, $\mathsf{Decrypt}(X, t)$, $\mathsf{Verify}(X, t)$ similarly mean that receive, new,

decrypt and signature verification actions occur.

$\mathsf{Fresh}(X, t)$ the term t generated in X is "fresh" in the sense that no one else has seen any term containing t as a subterm. Typically, a fresh term will be a nonce and freshness will be used to reason about the temporal ordering of actions in runs of a protocol.

$\mathsf{Honest}(\hat{X})$ the actions of principal $\hat{X}$ in the current run are precisely an interleaving of initial segments of traces of a set of roles of the protocol. In other words, $\hat{X}$ assumes some set of roles and does exactly the actions prescribed by them.

$\mathsf{Computes}(X, m)$ means that a principal $\hat{X}$ possesses enough information in thread X to build term m of some type. For example, a principal can possess an encrypted message if he received it in the past as a part of some message or if he possess the plaintext and the encryption key. Computes is used to describe the latter case.

$\mathsf{Start}(X)$ means that the thread X did not execute any actions in the past.

$\mathsf{isLess}(a, b)$ means that the number a is less than the number b.

$\mathsf{a_1} \leq \mathsf{a_2}$ means that both actions a_1 and a_2 happened in the run and moreover, that the action a_2 happened after the action a_1.

A.3 Proof System

A.3.1 Axioms for Protocol Actions

Axioms for protocol actions state properties that hold in the state as a result of (not) executing certain actions. Note that the a in axioms is any one of the 4 actions (send, receive, match, new) and a is the corresponding predicate in the logic.

AA1 $\phi[a]_X\ \mathsf{a}$

AA2 $\mathsf{Start}(X)[\,]_X\ \neg\mathsf{a}(\mathsf{X})$

AA3 $\neg\mathsf{Send}(X,t)[b]_X\neg\mathsf{Send}(X,t)$

if $\sigma\mathsf{Send}(X,t) \neq \sigma\mathsf{b}$ for all substitutions σ

(side condition for AA3 states that b)

(cannot be unified with $\mathsf{Send}(\mathsf{X},\mathsf{t})$)

AA4 $\phi[a_1; a_2; \ldots; a_k]_X \mathsf{a}_1 < \mathsf{a}_2 \wedge \ldots \wedge \mathsf{a}_{k-1} < \mathsf{a}_k$

AA5 $\neg(\mathsf{Send}(X,m) \wedge \mathsf{Contains}(m,t))[b]_X$

$\neg(\mathsf{Send}(X,m) \wedge \mathsf{Contains}(m,t))$

if $\sigma\mathsf{Send}(X,t) \neq \sigma\mathsf{b}$ for all substitutions σ

AN2 $\phi[\texttt{new}\ \ x]_X\ \mathsf{Has}(Y,x) \supset (Y = X)$

AN3 $\phi[\texttt{new}\ \ x]_X\ \mathsf{Fresh}(X,x)$

ARP $\mathsf{Receive}(X,p(x))[\texttt{match}\ \ q(x)/q(t)]_X\ \mathsf{Receive}(X,p(t))$

A.3.2 Axioms for PTK derivation

The principals executing the 4-Way Handshake use a shared secret established by TLS, along with nonces exchanged in the messages of the 4-Way Handshake, to derive a fresh shared secret called the Pairwise Temporary Key (PTK).

HASH1 $\mathsf{Computes}(X, HASH_K(x)) \supset$

$\mathsf{Has}(X,x) \wedge \mathsf{Has}(X,K)$

$$
\begin{array}{rl}
\textbf{HASH2} & \mathsf{Computes}(X, HASH_K(x)) \supset \\
& \mathsf{Has}(X, HASH_K(x)) \\
\textbf{HASH3} & \mathsf{Receive}(X, HASH_K(x)) \supset \\
& \exists Y.\mathsf{Computes}(Y, HASH_K(x)) \\
& \wedge \mathsf{Send}(Y, HASH_K(x)) \\
\textbf{HASH4} & \mathsf{Has}(X, HASH_K(x)) \supset \\
& \mathsf{Computes}(X, HASH_K(x)) \vee \\
& \exists Y, m. \quad \mathsf{Computes}(Y, HASH_K(x)) \\
& \wedge \mathsf{Send}(Y, m) \wedge \mathsf{Contains}(m, HASH_K(x)) \\
Define & \mathsf{Computes}(X, Hash_K(a)) \equiv \\
& \quad \mathsf{Has}(X, K) \wedge \mathsf{Has}(X, a)
\end{array}
$$

A.3.3 Possession Axioms

The possession axioms characterize the terms that a principal can derive if it possesses certain other terms. The last three axioms in this section relate to hashes.

$$
\begin{array}{rl}
\textbf{ORIG} & \mathsf{New}(X, x) \supset \mathsf{Has}(X, x) \\
\textbf{REC} & \mathsf{Receive}(X, x) \supset \mathsf{Has}(X, x) \\
\textbf{TUP} & \mathsf{Has}(X, x) \wedge \mathsf{Has}(X, y) \\
& \supset \mathsf{Has}(X, (x, y)) \\
\textbf{ENC} & \mathsf{Has}(X, x) \wedge \mathsf{Has}(X, K) \\
& \supset \mathsf{Has}(X, ENC_K\{|x|\}) \\
\textbf{PROJ} & \mathsf{Has}(X, (x, y)) \supset \mathsf{Has}(X, x) \wedge \mathsf{Has}(X, y) \\
\textbf{DEC} & \mathsf{Has}(X, ENC_K\{|x|\}) \wedge \mathsf{Has}(X, K) \\
& \supset \mathsf{Has}(X, x)
\end{array}
$$

A.3.4 Encryption and Signature

The next two axioms are aimed at capturing the black-box model of encryption and signature. **VER** refers to the unforgeability of signatures while **SEC** stipulates the need to possess the private key in order to decrypt a message encrypted with the corresponding public key.

$$\begin{array}{ll} \textbf{SEC} & \mathsf{Honest}(\hat{X}) \wedge \mathsf{Decrypt}(Y, ENC_{\hat{X}}\{\!|x|\!\}) \supset (\hat{Y} = \hat{X}) \\ \textbf{VER} & \mathsf{Honest}(\hat{X}) \wedge \mathsf{Verify}(Y, SIG_{\hat{X}}\{\!|x|\!\}) \wedge \hat{X} \neq \hat{Y} \supset \\ & \quad \exists m.\exists s.\mathsf{Send}(\langle \hat{X}, s\rangle, m) \wedge \mathsf{Contains}(m, SIG_{\hat{X}}\{\!|x|\!\}) \end{array}$$

A.3.5 Generic Rules

$$\frac{\theta[P]_X\phi \quad \theta[P]_X\psi}{\theta[P]_X\phi \wedge \psi}\ \mathbf{G1} \qquad \frac{\theta[P]_X\psi \quad \phi[P]_X\psi}{\theta \vee \phi[P]_X\psi}\ \mathbf{G2}$$

$$\frac{\theta[P]_X\phi \quad \theta' \supset \theta \quad \phi \supset \phi'}{\theta'[P]_X\phi'}\ \mathbf{G3} \qquad \frac{\phi}{\theta[P]_X\phi}\ \mathbf{G4}$$

A.3.6 Sequencing rule

Sequencing rule gives us a way of sequentially composing two cords P and P' when post-condition of P, matches the pre-condition of P'.

$$\frac{\phi_1[P]_A\phi_2 \qquad \phi_2[P']_A\phi_3}{\phi_1[PP']_A\phi_3}\ \mathbf{S1}$$

A.3.7 Preservation Axioms

For $\mathsf{Persist} \in \{\mathsf{Has}, \mathsf{FirstSend}, \mathsf{a} < \mathsf{b}, \mathsf{a}\}$:

P1 $\mathsf{Persist}(X,t)[a]_X\mathsf{Persist}(X,t)$

P2 $\mathsf{Fresh}(X,t)[a]_X\mathsf{Fresh}(X,t)$, where $t \not\sqsubseteq a$ or $a \neq \langle m \rangle$
(i.e., it is still fresh if you do not send it)

A.3.8 Axioms and Rules for Temporal Ordering

The next two axioms give us a way of deducing temporal ordering between actions of different threads. Informally, $\mathsf{FirstSend}(X,t,t')$ says that a thread X generated a fresh term t and sent it out first in message t'. This refers to such a first send event.

FS1 $\mathsf{Fresh}(X,t)[\mathtt{send}\ t']_X\mathsf{FirstSend}(X,t,t')$,
where $t \sqsubseteq t'$.

FS2 $\mathsf{FirstSend}(X,t,t') \wedge \mathsf{a}(Y,t") \supset \mathsf{Send}(X,t') < \mathsf{a}(Y,t")$,
where $X \neq Y$ and $t \sqsubseteq t"$.

A.3.9 Honesty Rule

$$\frac{\mathsf{Start}(X)[\,]_X\ \phi \qquad \forall \rho \in \mathcal{Q}.\forall P \in BS(\rho).\ \phi\ [P]_X\ \phi}{\mathsf{Honest}(\hat{X}) \supset \phi}\ \mathbf{HON_Q}$$

Note that there is no free variable in ϕ except X bound in $[P]_X$.

Appendix B

Proof of the 4-Way Handshake Guarantee

In Section 4.2, we state the security of the 4-Way Handshake in the authenticator side as Theorem 1, which claims both the security properties, including *key secrecy* and *session authentication*, and the *invariants* of the 4-Way Handshake. Here we describe the details of the proof.

The *invariants* are verified by induction over basice sequences. When proving *key secrecy* of the 4-Way Handshake, we need to show that the secrecy of the key established by TLS is preserved. This is accomplished by proving that the $\mathsf{NonceSource}$ predicate is preserved after each step, and that the $\Gamma_{tls,2}$ invariant is satisfied by the 4-Way Handshake. The details of the proof can be found in our website. In the proof of *key secrecy*, we proved that the following property holds after each step of the 4-Way Handshake.

$$\textbf{SPMK} \quad \mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset \mathsf{Has}(\hat{Z}, pmk) \supset \hat{Z} = \hat{X} \vee \hat{Y}$$

Using this formula, we will describe the proof details for *session authentication* as an example to show the general proof structures, stated as

$$\theta_{4way}[\mathbf{4WAY} : \mathbf{AUTH}]_X \phi_{4way,auth}$$

.

Recall that *session authentication* is represented as matching conversations; when the 4-Way Handshake is executed, the authenticator $\hat{X}$ can reason as follows:

1. Since the authenticator is honest, obviously it knows that actions of itself are in order; i.e., Send $Message$ 1, Receive $Message$ 2, Send $Message$ 3, Receive $Message$ 4 are matched, represented in line (1) of the proof;

2. Since the authenticator received and verified *Message 4*, there must be some entity $\hat{Z}$ who computes and sends out *Message 4* at previous stages, which implies $\hat{Z}$ must know the *ptk* used in the MIC to protect the message, indicated in line (2) and (3);

3. According to the invariants of the 4-Way Handshake and the properties of keyed hash, $\hat{Z}$ must be either the authenticator $\hat{X}$ itself or the supplicant $\hat{Y}$ since these are the only two parties who have the *ptk* in the system, described in line (4)-(6);

4. The authenticator knows it does not send out *Message 4* by itself; thus, it must be the supplicant who have computed and sent out *Message 4*. Furthermore, this occurs before the authenticator receives this *Message 4*, as shown in line (7)-(9);

5. The authenticator assumes that the honest supplicant acts honestly obeying the protocol. Therefore, the supplicant must has a sequence of Receive and Send actions in order, i.e., Receive $Message$ 1, Send $Message$ 2, Receive $Message$ 3, Send $Message$ 4, as in line (10);

6. The remaining task is to match the sequence of actions between the authenticator side and the supplicant side. Since the supplicant must have received and verified *Message 3* before sending out *Message 4*, the authenticator can similarly reason as in step 2, 3, and 4, and conclude that it must have sent out *Message 3* before the supplicant actually received it, as shown in line (11)-(15);

7. Due to the freshness of the nonce generated by the supplicant, the authenticator can only receive *Message 2* after the supplicant sends it. Similarly due to the

freshness of the nonce generated by the authenticator, the supplicant can only receive *Message 1* after the authenticator send it, shown in line (16)-(19).

Based on these arguments, all the actions are matched as in line (20). Hence, the authenticator can conclude that the security property of *session authentication* is guaranteed in the 4-Way Handshake.

$\mathbf{AA1}, \mathbf{ARP}, \mathbf{AA4}$ θ_{4way}
$[\mathbf{4WAY} : \mathbf{AUTH}]_X$
$\mathsf{Send}(X, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”}) <$
$\mathsf{Receive}(X, \hat{Y}, \hat{X}, y, \text{“}msg2\text{”}, HASH_{ptk}(y, \text{“}msg2\text{”})) <$
$\mathsf{Send}(X, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”})) <$
$\mathsf{Receive}(X, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”}))$ (B.1)

$\mathbf{ARP}, \mathbf{HASH3}$ θ_{4way}
$[\texttt{receive}\ \hat{Y}, \hat{X}, z;$
$\texttt{match}\ z/\text{“}msg4\text{”}, mic2;\ \texttt{match}\ mic2/HASH_{ptk}(\text{“}msg4\text{”})]_X$
$\mathsf{Receive}(X, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”})) \supset$
$\exists Z.\mathsf{Computes}(Z, HASH_{ptk}(\text{“}msg4\text{”})) \wedge$
$\mathsf{Send}(Z, HASH_{ptk}(\text{“}msg4\text{”})) \wedge$
$(\mathsf{Send}(Z, HASH_{ptk}(\text{“}msg4\text{”})) <$
$\mathsf{Receive}(X, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”})))$ (B.2)

$\mathbf{HASH1}$ $\mathsf{Computes}(Z, HASH_{ptk}(\text{“}msg4\text{”})) \equiv$
$\mathsf{Has}(\hat{Z}, ptk) \wedge \mathsf{Has}(\hat{Z}, \text{“}msg4\text{”})$ (B.3)

$\mathbf{HASH4}$ $\mathsf{Has}(\hat{Z}, ptk) \equiv \mathsf{Has}(\hat{Z}, HASH_{pmk}(x, y)) \supset$
$\mathsf{Computes}(Z, HASH_{pmk}(x, y)) \vee$
$(\exists Y, m.\mathsf{Computes}(Y, HASH_{pmk}(x, y)) \wedge$
$\mathsf{Send}(Y, m) \wedge \mathsf{Contains}(m, HASH_{pmk}(x, y)))$ (B.4)

$(4), \Gamma_{4way,2}$ θ_{4way}
$[\texttt{receive}\ \hat{Y}, \hat{X}, z;$
$\texttt{match}\ z/\text{“}msg4\text{”}, mic2;\ \texttt{match}\ mic2/HASH_{ptk}(\text{“}msg4\text{”})]_X$
$\mathsf{Has}(\hat{Z}, ptk) \equiv \mathsf{Has}(Z, HASH_{pmk}(x, y)) \supset$
$\mathsf{Computes}(Z, HASH_{pmk}(x, y))$ (B.5)

$(5), \mathbf{HASH1}, \mathbf{SPMK}$ θ_{4way}
$[\texttt{receive}\ \hat{Y}, \hat{X}, z;$
$\texttt{match}\ z/\text{“}msg4\text{”}, mic2;\ \texttt{match}\ mic2/HASH_{ptk}(\text{“}msg4\text{”})]_X$
$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset \mathsf{Computes}(Z, HASH_{ptk}(\text{“}msg4\text{”})) \supset$
$\mathsf{Has}(\hat{Z}, ptk) \supset \mathsf{Has}(\hat{Z}, pmk) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{Y}$ (B.6)

$\mathbf{AA1}, \Gamma_{4way,3}$ θ_{4way}
$[\texttt{new}\ x;\ \texttt{send}\ \hat{X}, \hat{Y}, x, \text{“}msg1\text{”};]_X$
$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Send}(X, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”}) \supset \hat{Z} \neq \hat{X}$ (B.7)

$(2), (6), (7)$ θ_{4way}
$[\mathbf{4WAY} : \mathbf{AUTH}]_X$
$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset$
$\exists Z.\mathsf{Computes}(Z, HASH_{ptk}(\text{“}msg4\text{”})) \wedge$
$\mathsf{Send}(Z, HASH_{ptk}(\text{“}msg4\text{”})) \wedge \hat{Z} = \hat{Y}$ (B.8)

$(2), (8)$ θ_{4way}
$[\mathbf{4WAY} : \mathbf{AUTH}]_X$
$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset$
$\mathsf{Send}(Y, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”})) <$
$\mathsf{Receive}(X, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”}))$ (B.9)

$(9), \phi_{\mathbf{HONESTY}}$ θ_{4way}
$[\mathbf{4WAY} : \mathbf{AUTH}]_X$
$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset$
$\mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”}) <$
$\mathsf{Send}(Y, \hat{Y}, \hat{X}, y, \text{“}msg2\text{”}, HASH_{ptk}(y, \text{“}msg2\text{”})) <$
$\mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”})) <$
$\mathsf{Send}(Y, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”}))$ (B.10)

$(10), \mathbf{HASH3}$ θ_{4way}
$[\mathbf{4WAY} : \mathbf{AUTH}]_X$
$\mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”})) \supset$
$\exists Z.\mathsf{Computes}(Z, HASH_{ptk}(x, \text{“}msg3\text{”})) \wedge$
$\mathsf{Send}(Z, HASH_{ptk}(x, \text{“}msg3\text{”})) \wedge$
$(\mathsf{Send}(Z, HASH_{ptk}(x, \text{“}msg3\text{”})) <$
$\mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”})))$ (B.11)

$\mathbf{HASH1}, (5), (6)$ θ_{4way}
$[\mathbf{4WAY} : \mathbf{AUTH}]_X$
$\mathsf{Computes}(Z, HASH_{ptk}(x, \text{“}msg3\text{”})) \supset$
$\mathsf{Has}(\hat{Z}, ptk) \supset \hat{Z} = \hat{X} \vee \hat{Z} = \hat{Y}$ (B.12)

$\Gamma_{4way,3}$ θ_{4way}
$[\mathbf{4WAY} : \mathbf{AUTH}]_X$
$\mathsf{Honest}(\hat{Y}) \wedge \mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”}) \supset \hat{Z} \neq \hat{Y}$ (B.13)

$(11), (12), (13)$ θ_{4way}
$[\mathbf{4WAY} : \mathbf{AUTH}]_X$
$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset$
$\exists Z.\mathsf{Computes}(Z, HASH_{ptk}(x, \text{“}msg3\text{”})) \wedge$
$\mathsf{Send}(Z, HASH_{ptk}(x, \text{“}msg3\text{”})) \wedge \hat{Z} = \hat{X}$ (B.14)

$(11), (14)$ θ_{4way}
$[\mathbf{4WAY} : \mathbf{AUTH}]_X$
$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset$
$\mathsf{Send}(X, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”})) <$
$\mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”}))$ (B.15)

$\mathbf{FS1}, \mathbf{AN3}$ θ_{4way}
$[\mathbf{4WAY} : \mathbf{AUTH}]_X$
$\mathsf{Honest}(\hat{Y}) \supset$
$\mathsf{FirstSend}(Y, y, \hat{Y}, \hat{X}, y, \text{“}msg2\text{”}, HASH_{ptk}(y, \text{“}msg2\text{”}))$ (B.16)

$(16), \mathbf{FS2}$ θ_{4way}
$[\mathbf{4WAY} : \mathbf{AUTH}]_X$
$\mathsf{Send}(Y, \hat{Y}, \hat{X}, y, \text{“}msg2\text{”}, HASH_{ptk}(y, \text{“}msg2\text{”})) <$
$\mathsf{Receive}(X, \hat{Y}, \hat{X}, y, \text{“}msg2\text{”}, HASH_{ptk}(y, \text{“}msg2\text{”}))$ (B.17)

$\mathbf{FS1}, \mathbf{AN3}$ θ_{4way}
$[\texttt{new } x; \texttt{send } \hat{X}, \hat{Y}, x, \text{“}msg1\text{”};]_X$
$\mathsf{FirstSend}(X, x, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”})$ (B.18)

$(18), \mathbf{FS2}$ θ_{4way}
$[\mathbf{4WAY} : \mathbf{AUTH}]_X$
$\mathsf{Send}(X, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”}) < \mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”})$ (B.19)

$(1, 9, 10, 15, 17, 19)$ θ_{4way}
$[\mathbf{4WAY} : \mathbf{AUTH}]_X$
$\mathsf{Honest}(\hat{X}) \wedge \mathsf{Honest}(\hat{Y}) \supset$
$\mathsf{Send}(X, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”}) < \mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg1\text{”}) <$
$\mathsf{Send}(Y, \hat{Y}, \hat{X}, y, \text{“}msg2\text{”}, HASH_{ptk}(y, \text{“}msg2\text{”})) <$
$\mathsf{Receive}(X, \hat{Y}, \hat{X}, y, \text{“}msg2\text{”}, HASH_{ptk}(y, \text{“}msg2\text{”})) <$
$\mathsf{Send}(X, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”})) <$
$\mathsf{Receive}(Y, \hat{X}, \hat{Y}, x, \text{“}msg3\text{”}, HASH_{ptk}(x, \text{“}msg3\text{”})) <$
$\mathsf{Send}(Y, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”})) <$
$\mathsf{Receive}(X, \hat{Y}, \hat{X}, \text{“}msg4\text{”}, HASH_{ptk}(\text{“}msg4\text{”}))$ (B.20)

Bibliography

[1] Verified By Visa. 2004. https://usa.visa.com/personal/security/vbv/.

[2] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The Spi Calculus. In *Proceedings of Fourth ACM Conference on Computer and Communications Security*, 1997.

[3] B. Aboba. Pros and Cons of upper layer network access. IEEE documents 802.11-00/382, November 2000.

[4] B. Aboba and A. Palekar. IEEE 802.1X and RADIUS security. Submissions to IEEE 802.11 TGi, November 2001.

[5] B. Aboba and D. Simon. PPP EAP TLS authentication protocol. RFC 2716, October 1999.

[6] W. Aiello, S. M. Bellovin, M. Blaze, J. Ioannidis, O. Reingold, R. Canetti, and A. D. Keromytis. Efficient, DoS-resistant, secure key exchange for Internet Protocols. In *the 9th ACM conference on Computer and communications security*, pages 48–58, 2002.

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

[8] W. A. Arbaugh, N. Shankar, and J. Wang. Your 802.11 network has no clothes. In *the first IEEE International Conference on Wireless LANs and Home Networks*, pages 131–144, December 2001.

[9] N. Asokan, V. Niemi, and K. Nyberg. Man-in-the-Middle in tunneled authentication protocols. Technical Report 2002/163, October 2002.

[10] T. Aura and P. Nikander. Stateless connections. In *International Conference on Information and Communications Security (ICICS'97)*, pages 87–97, November 1997.

[11] AusCERT AA-2004.02. Denial of Service vulnerability in IEEE 802.11 wireless devices. Technical Report 2002/163, May 13 2004.

[12] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens. An on-demand secure routing protocol resilient to Byzantine failures. In *Proceedings of ACM Workshop on Wireless Security (WiSe'02)*, pages 21–30, 2002.

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

[14] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryprtology - Crypto'93 Proceedings*. Springer-Verlag, 1994.

[15] L. Blunk, J. Vollbrecht, B. Aboba, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). Internet Draft draft-ietf-eap-rfc2284bis-06.txt, September 29 2003.

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

[17] S. Bouam and J. Ben-Othman. Data security in ad hoc networks using multipath routing. In *14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications (PIMRC'03)*, pages 1331 – 1335, 2003.

[18] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[19] N. Cam-Winget, R. Housley, D. Wagner, and J. Walker. Security flaws in 802.11 data link protocols. *SPECIAL ISSUE: Wireless networking security, Communications of the ACM*, 46(5):35–39, May 2003.

[20] S. Capkun and J.-P. Hubaux. Building secure routing out of an incomplete set of security associations. In *Proceedings of the 2003 ACM workshop on Wireless security*, 2003.

[21] D. Chen, J. Deng, and P. K. Varshney. Protecting wireless networks against a Denial of Service attack based on virtual jamming. In Poster Session of MobiCom2003, September 2003.

[22] H. Cheung. FBI Teaches Lesson in how to break into Wi-Fi networks, 2005. http://informationweek.networkingpipeline.com/160700382.

[23] Cisco Systems. Cisco Aironet response to University of Maryland's paper, An initial security analysis of the IEEE 802.1X standard, August 2002.

[24] A. Datta, A. Derek, J. Mitchell, V. Shmatikov, and M. Turuani. Probabilistic polynomial-time semantics for a protocol security logic. In *Int'l Colloquium on Automata, Languages, and Programming (ICALP)*, 2005.

[25] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system for security protocols and its logical formalization. In *Proceedings of 16th IEEE Computer Security Foundations Workshop*, pages 109–125. IEEE, 2003.

[26] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Abstraction and refinement in protocol derivation. In *Proceedings of 17th IEEE Computer Security Foundations Workshop*, pages 30–45. IEEE, 2004.

[27] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security*, 2004.

[28] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Secure protocol composition. In *Proceedings of 19th Annual Conference on Mathematical Foundations*

of Programming Semantics, volume 83 of *Electronic Notes in Theoretical Computer Science*, 2004.

[29] T. Dierks and C. Allen. The TLS Protocol - Version 1.0. IETF RFC 2246, January 1999.

[30] D. L. Dill. The Murϕ verification system. In *the 8th International Conference on Computer Aided Verification*, pages 390–393, July 1996.

[31] D. L. Dill, S. Park, and A. G. Nowatzyk. Formal specification of abstract memory models. In *Symposium on Research on Integrated Systems*, pages 38–52, 1993.

[32] P. Ding, J. Holliday, and A. Celik. Improving the security of Wireless LANs by managing 802.1X Disassociation. In *the IEEE Consumer Communications and Networking Conference (CCNC'04)*, January 2004.

[33] J. Duntemann. Wardriving FAQ, April 2003.

[34] N. Durgin, J. C. Mitchell, and D. Pavlovic. A compositional logic for protocol correctness. In *Proceedings of 14th IEEE Computer Security Foundations Workshop*, pages 241–255. IEEE, 2001.

[35] N. Durgin, J. C. Mitchell, and D. Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11:677–721, 2003.

[36] D. B. Faria and D. R. Cheriton. DoS and authentication in wireless public access networks. In *the first ACM Workshop on Wireless Security (WiSe'02)*, September 2002.

[37] D. B. Faria and D. R. Cheriton. No long-term secrets: Location-based security in overprovisioned wireless LANs. In *Proceedings of the Third ACM Workshop on Hot Topics in Networks (HotNets-III)*, 2004.

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

[39] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of RC4. In *Lecture Notes In Computer Science, Revised Paper from the 8th Annual International Workshop on Selected Areas in Cryptography*, pages 1–24, 2001.

[40] 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 (WiSe'04)*, 2004.

[41] 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 (NDSS'05)*, 2005.

[42] C. He, M. Sundararajan, A. Datta, A. Derek, and J. C. Mitchell. A modular correctness proof of IEEE 802.11i and TLS. In *Proceedings of 12th ACM Conference on Computer and Communications Security (CCS'05)*, 2005.

[43] Y. Hu, A. Perrig, and D. Johnson. Efficient security mechanisms for routing protocols. In *Proceedings of Network and Distributed Systems Security (NDSS'03)*, 2003.

[44] Y.-C. Hu, D. B. Johnson, and A. Perrig. SEAD: Secure efficient distance vector routing in mobile wireless ad hoc networks. In *Proceedings of 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'02)*, pages 3–13, 2002.

[45] Y.-C. Hu and A. Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy Magazine*, 2:28–39, 2004.

[46] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne - a secure on-demand routing protocol for ad hoc networks. In *Proceedings of 8th Annual International*

Conference on Mobile Computing and Networking (MobiCom'02), pages 12–23, 2002.

[47] Y.-C. Hu, A. Perrig, and D. B. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *Proceedings of Proc. of 22nd Annual Joint Conference of IEEE Computer and Communications Societies (InfoCom03)*, pages 1976–1986. IEEE, 2003.

[48] Y.-C. Hu, A. Perrig, and D. B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of 2003 ACM Workshop on Wireless Security (WiSe'03)*, pages 30–40. ACM, 2003.

[49] J.-P. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proceedings of 2nd Symposium Mobile Ad Hoc Networking and Computing (MobiHoc'01)*, pages 146–155, 2001.

[50] 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, April 2004.

[51] IEEE Standard 802.11-1999. Information technology - Telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - Part 11: Wireless LAN Medium Access Control and Physical Layer Specifications, 1999.

[52] IEEE Standard 802.11a-1999. Higher-speed Physical Layer in the 5 Ghz band, supplement to IEEE standard for information technology - Telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, 1999.

[53] IEEE Standard 802.11b-1999. Higher-speed Physical Layer extension in the 2.4 Ghz band, supplement to IEEE standard for information technology - Telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, September 1999.

[54] IEEE Standard 802.11g-2003. Amendment 4: Further higher data rate extension in the 2.4 Ghz band, IEEE standard for information technology - Telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, 2003.

[55] IEEE Standard 802.1X-2001. IEEE standard for local and metropolitan area networks - port-based network access control, June 2001.

[56] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[57] J. Jonsson. On the security of CTR + CBC-MAC. In *Lecture Notes In Computer Science, Revised Paper from the 9th Annual International Workshop on Selected Areas in Cryptography*, pages 76–93, 2002.

[58] P. Kyasanur and N. H. Vaidya. Detection and handling of MAC layer misbehavior in wireless networks. In *the International Conference on Dependable Systems and Networks (DSN'03)*, June 2003.

[59] C.-L. Lee, X.-H. Lin, and Y.-K. Kwok. A multipath ad hoc routing approach to combat wireless link insecurity. In *2003 IEEE International Conference on Communications (ICC'03)*, pages 448 – 452, 2003.

[60] W. Lou, W. Liu, and Y. Fang. A simulation study of security performance using multipath routing in ad hoc networks. In *2003 IEEE 58th Vehicular Technology Conference (VTC 2003-Fall)*, pages 2142 – 2146, 2003.

[61] M. Lynn and R. Baird. Advanced 802.11 attack. Black Hat Briefings, July 2002.

[62] J. Malinen. HostAP. http://hostap.epitest.fi/.

[63] J. Marshall, V. Thakur, and A. Yasinsac. Identifying flaws in the secure routing protocol. In *Proceedings of the 2003 IEEE International Performance, Computing, and Communications Conference*, pages 167 – 174, 2003.

[64] T. Marshall. Antennas enhance WLAN security, October 2001.

[65] K. Matsuura and H. Imai. Modified aggressive mode of Internet key exchange resistant against Denial-of-Service attacks. *IEICE Transactions on Information and Systems*, E83-D(5):972–979, May 2000.

[66] C. Meadows. A model of computation for the NRL protocol analyzer. In *Proceedings of 7th IEEE Computer Security Foundations Workshop*, pages 84–89. IEEE, 1994.

[67] C. Meadows. Open issues in formal methods for cryptographic protocol analysis. *Lecture Notes in Computer Science*, 2052:21–36, 2001.

[68] C. Meadows and D. Pavlovic. Deriving, attacking and defending the GDOI protocol. In *ESORICS*, pages 53–72, 2004.

[69] J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of ACM Conference on Computer and Communications Security*, pages 166–175, 2001.

[70] A. Mishra and W. A. Arbaugh. An initial security analysis of the IEEE 802.1X standard. Technical Report CS-TR-4328, UMIACS-TR-2002-10, University of Maryland, February 2002.

[71] J. C. Mitchell. Finite-State analysis of security protocols. In *Computer Aided Verification*, pages 71–76, 1998.

[72] J. C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murϕ. In *IEEE Symposium on Security and Privacy*, pages 141–151, 1997.

[73] J. C. Mitchell, V. Shmatikov, and U. Stern. Finite-State analysis of SSL 3.0. In *Proceedings of Seventh USENIX Security Symposium*, pages 201–216, 1998.

[74] J. C. Mitchell, V. Shmatikov, and U. Stern. Finite-state analysis of SSL 3.0. In *Proceedings of Seventh USENIX Security Symposium*, pages 201–216, 1998.

[75] 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.

[76] T. Moore. Validating 802.11 disassociation and deauthentication messages. Submission to IEEE P802.11 TGi, September 2002.

[77] R. Moskowitz. Weakness in passphrase choice in WPA interface, November 2003.

[78] National Institute of Standards and Technology. FIPS Pub 197: Advanced Encryption Standard (AES), November 26 2001.

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

[80] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS02)*, 2002.

[81] P. Papadimitratos and Z. J. Haas. Secure link state routing for mobile ad hoc networks. In *Proceedings of IEEE Workshop on Security and Assurance in ad hoc networks*, pages 27–31, 2003.

[82] J. S. Park and D. Dicoi. WLAN security: current and future. *IEEE Internet Computing*, 8(2):76–83, April 2004.

[83] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[84] L. C. Paulson. Inductive analysis of the Internet protocol TLS. *ACM Transactions on Computer and System Security*, 2(3):332–351, 1999.

[85] L. C. Paulson. Verifying the SET protocol. In *Proceedings of International Joint Conference on Automated Reasoning*, 2001.

[86] L. Ricciulli, P. Lincoln, and P. Kakkar. TCP SYN flooding defense. In *In Communication Networks and Distributed Systems Modeling and Simulation (CNDS'99)*, 1999.

[87] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS). RFC 2865, June 2000.

[88] P. Rogaway and D. Wagner. A critique of CCM. Unpublished manuscript, February 2 2003.

[89] A. Roy and H. Chang. Physical Layer security of WLANs.

[90] P. Y. A. Ryan, S. A. Schneider, M. H. Goldsmith, G. Lowe, and A. W. Roscoe. *Modelling and Analysis of Security Protocols*. Addison-Wesley Publishing Co., 2000.

[91] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of 10th IEEE International Conference of Network Protocols (ICNP02)*, pages 78–87, 2002.

[92] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley and Sons, New York, second edition, 1996.

[93] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundram, and D. Zamboni. Analysis of a Denial of Service attack on TCP. In *1997 IEEE Symposium on Security and Privacy*, 1997.

[94] SETCo. SET Secure Electronic Transaction specification: Business description, 1997.

[95] P. Shipley. Open WLANs: the early result of Wardriving, 2001.

[96] U. Stern and D. L. Dill. Automatic verification of the SCI cache coherence protocol. In *Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, pages 21–34, 1995.

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

[98] F. J. Thayer-Fábrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Why is a security protocol correct? In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, pages 160–171, Oakland, CA, May 1998. IEEE Computer Society Press.

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

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

[101] C. Wullems, K. Tham, J. Smith, and M. Looi. Technical summary of Denial of Service attack against IEEE 802.11 DSSS based Wireless LAN's.

[102] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang. Security in mobile ad hoc networks: challenges and solutions. *IEEE Wireless Communications [see also IEEE Personal Communications]*, 11:38 – 47, 2004.

[103] L. Yang, D. Gao, J. Mostoufi, R. Joshi, and P. Loewenstein. System design methodology of UltraSPARCtm-i. In *32nd Design Automation Conference*, pages 7–12, 1995.

[104] S. Yi, P. Naldurg, and R. Kravets. Security-aware ad-hoc routing for wireless networks, 2001.

[105] M. G. Zapata. Secure ad hoc on-demand distance vector (SAODV) routing, 2002.

[106] M. G. Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proceedings of ACM Workshop on Wireless Security (WiSe02)*, pages 1–10, 2002.