KUOPION YLIOPISTON JULKAISUJA H. INFORMAATIOTEKNOLOGIA JA KAUPPATIETEET 13
KUOPIO UNIVERSITY PUBLICATIONS H. BUSINESS AND INFORMATION TECHNOLOGY 13

KEIJO HAATAJA

Security Threats and Countermeasures in Bluetooth-Enabled Systems

Doctoral dissertation

To be presented by permission of the Faculty of Business and Information Technology of the University of Kuopio for public examination in Auditorium MET, Mediteknia building, University of Kuopio, on Friday 6th February 2009, at 12 noon

Department of Computer Science
University of Kuopio

KUOPION YLIOPISTO

KUOPIO 2009

Distributor: Kuopio University Library
P.O. Box 1627
FI-70211 KUOPIO
FINLAND
Tel. +358 40 355 3430
Fax +358 17 163 410
http://www.uku.fi/kirjasto/julkaisutoiminta/julkmyyn.shtml

Series Editors: Professor Markku Nihtilä, D.Sc.
Department of Mathematics and Statistics

Assistant Professor Mika Pasanen, Ph.D.
Department of Business and Management

Author's address: Department of Computer Science
University of Kuopio
P.O. Box 1627
FI-70211 KUOPIO
FINLAND
Tel. +358 40 355 2563
Fax +358 17 162 595
E-mail: Keijo.Haataja@uku.fi

Supervisors: Docent Elena Trichina, Ph.D.
Department of Computer Science
University of Kuopio

Professor Martti Penttonen, Ph.D.
Department of Computer Science
University of Kuopio

Reviewers: Professor Ahmad-Reza Sadeghi, Ph.D.
Chair for System Security
Ruhr-University, Bochum, Germany

Guido Bertoni, PhD.
AST, ST Microelectronics
VIA C. Olivetti 2, 20041 Agrate B.za, Italy

Professor Jari Veijalainen
Department of Computer Science and Information Systems
University of Jyväskylä

Opponent: Associate Professor Susanne Wetzel
Department of Computer Science
Stevens Institute of Technology, Hoboken, USA

ISBN 978-951-781-992-3
ISBN 978-951-27-0111-7 (PDF)
ISSN 1459-7586

Kopijyvä
Kuopio 2009
Finland

Haataja, Keijo. Security Threats and Countermeasures in Bluetooth-Enabled Systems. Kuopio University Publications H. Business and Information Technology 13. 2009. 187 p.
ISBN 978-951-781-992-3
ISBN 978-951-27-0111-7 (PDF)
ISSN 1459-7586

ABSTRACT

Bluetooth is a technology for short range wireless data and realtime two-way voice transfer providing data rates up to 3 Mb/s. It can be used to connect almost any device to another device. Bluetooth-enabled devices, such as mobile phones, headsets, PCs, laptops, printers, mice, and keyboards, are widely used all over the world. Already in 2006, the one billionth Bluetooth device was shipped, and the volume is expected to increase rapidly in the near future. The target volume for 2010 is as high as two billion Bluetooth devices. Therefore, it is very important to keep Bluetooth security issues up-to-date.

As an interconnection technology, Bluetooth has to address all traditional security problems, well known from distributed networks. In addition, security issues in wireless ad-hoc networks are much more complex than those of more traditional wired or centralized wireless networks. Moreover, Bluetooth networks are formed by radio links, which means that there are additional security aspects whose impact is not yet well understood.

The aim of our work is to evaluate security threats in Bluetooth-enabled systems. Our research work concentrates on practical aspects of Bluetooth security. It can be roughly divided into four parts.

First, weaknesses of Bluetooth security are studied based on a literature review, and a Bluetooth security laboratory environment for implementing Bluetooth security attacks in practice has been built.

Secondly, different types of attacks against Bluetooth security are investigated and the feasibility of some of them are demonstrated in our research laboratory. Countermeasures against each type of attack are also proposed.

Thirdly, some of the existing Bluetooth security attacks are enhanced and new attacks are proposed. To carry out these attacks in practice, Bluetooth security analysis tools are implemented. Countermeasures that render these attacks impractical are also proposed.

Finally, a comparative analysis of the existing Man-In-The-Middle attacks on Bluetooth is presented, a novel system for detecting and preventing intrusions in Bluetooth networks is proposed, and a further classification of Bluetooth-enabled ad-hoc networks is provided.

Universal Decimal Classification: 004.056, 004.725.5, 004.732
Inspec Thesaurus: ad hoc networks; personal area networks; Bluetooth; security; security of data; telecommunication security

Acknowledgements

I wish to thank Elena Trichina, Martti Penttonen, and Tapio Grönfors for their guidance and supervision during the work presented in this thesis. I also want to thank the staff of the Department of Computer Science at the University of Kuopio, especially Konstantin Hyppönen for his cooperation on Bluetooth Man-In-The-Middle related research. Finally, I want to thank my parents Kari and Anneli, and also my dear girlfriend Riikka for her understanding and patience with my late working hours during my work on this thesis.

Kuopio, October 2008

_____

Keijo Haataja

Abbreviations and notations

3DES	Triple Data Encryption Standard; TDES
ACL	Asynchronous Connection-Less
ACO	Authenticated Ciphering Offset
AES	Advanced Encryption Standard
AFH	Adaptive Frequency Hopping
ASCII	American Standard Code for Information Interchange
ATM	Automated Teller Machine
BD_ADDR	Bluetooth Device Address
BER	Bit-Error-Rate
BNEP	Bluetooth Network Encapsulation Protocol
C/I	Carrier-to-Interference ratio; CIR
CA	Certification Authority
CF	Compact Flash
CL	Connection-Less
CO	Connection-Oriented
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSR	Cambridge Silicon Radio
dBi	Decibels relative to an isotropic source
dBm	Decibels relative to one milliwatt
DES	Data Encryption Standard
DID	Disclosure, Integrity and DoS
DoS	Denial-of-Service
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
DVD	Digital Versatile Disc; Digital Video Disc
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
EDR	Enhanced Data Rate
EFS	Encrypting File System
EncFS	Encrypted Filesystem

eSCO	Extended Synchronous Connection-Oriented
FHS	Frequency Hop Synchronization
FHSS	Frequency Hopping Spread Spectrum
FINEID	Finnish Electronic Identification
gdc	Greatest common divisor
GF	Galois Field
GSM	Global System for Mobile communications
HCI	Host Controller Interface
HEC	Header Error Check
HID	Human Interface Devices
HMAC	A keyed-Hash Message Authentication Code
HV1	High-quality Voice 1
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
iff	If and only if
IMEI	International Mobile Equipment Identity
IO	Input/Output
IP	Internet Protocol
IPSec	Internet Protocol Security
IRC	Internet Relay Chat
IrDA	Infrared Data Association
ISM	Industrial, Scientific, and Medical
IV	Initialization Vector
J2ME	Java 2 Micro Edition
K_A	Unit key
K_{AB}	Combination key
K_C	Encryption key
K_{init}	Initialization key
KSA	Key Scheduling Algorithm
L2CAP	Logical Link Control and Adaptation Protocol
L2TP	Layer 2 Tunneling Protocol
LAP	Lower Address Part
LC	Link Controller
LM	Link Manager

LMP	Link Manager Protocol
MAC	Message Authentication Code
MD5	Message-Digest 5
MITM	Man-In-The-Middle
mod	modulo
NAK	Negative Acknowledgement
NAP	Nonsignificant Address Part
NFC	Near Field Communication
NIST	National Institute of Standards and Technology
NSA	National Security Agency
NTFS	New Technology File System
OBEX	Object Exchange Protocol
OOB	Out-Of-Band
PCMCIA	Personal Computer Memory Card International Association
PDA	Personal Digital Assistant
PDF	Portable Document Format
PDU	Protocol Data Unit
PHY	Physical layer
PKI	Public Key Infrastructure
PIN	Personal Identification Number
PL	Path Loss
PPP	Point-to-Point Protocol
QoS	Quality-of-Service
RAND	Pseudorandom number
RC4	Rivest Cipher 4; Ron's Code 4; ARCFOUR
RC5	Rivest Cipher 5; Ron's Code 5
RC6	Rivest Cipher 6; Ron's Code 6
ReiserFS	Reiser Filesystem
RF	Radio Frequency
RFCOMM	Radio Frequency Communication
RSA	Rivest-Shamir-Adleman
RSSI	Received Signal Strength Indicator
RX	Receiver
SAFER+	Secure And Fast Encryption Routine +

SCO	Synchronous Connection-Oriented
SDP	Service Discovery Protocol
Seattle	Bluetooth 3.0
SHA	Secure Hash Algorithm
SIG	Special Interest Group
SIS	Symbian Installation System
SRES	Signed Response
SSH	Secure Shell
SSH1	Secure Shell 1; SSH-1
SSH2	Secure Shell 2; SSH-2
SSL	Secure Sockets Layer
SSP	Secure Simple Pairing
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
TCS	Telephony Control protocol Specification
TI	Texas Instruments
TLS	Transport Layer Security
TX	Transmitter
UAP	Upper Address Part
UDP	User Datagram Protocol
USB	Universal Serial Bus
USRP	Universal Software Radio Peripheral
UWB	Ultra-Wideband
WAV	Waveform
WEP	Wired Equivalent Privacy
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network
XML	Extensible Markup Language
XOR	Exclusive OR

List of the original publications

Although this dissertation is written in monograph form, it covers mainly the results found in our ten published articles in unified form, including some direct citations to some parts of the articles. The articles are referred to in the text as normal literature references:

I. Haataja K.: *Bluetooth Security Threats and Possible Countermeasures*. Proceedings of the Annual Finnish Data Processing Week at the University of Petrozavodsk (FDPW'2004), Advances in Methods of Modern Information Technology, Vol. 6, Petrozavodsk, 2005, pp. 116-150.

II. Haataja K.: *Two Practical Attacks Against Bluetooth Security Using New Enhanced Implementations of Security Analysis Tools*. Proceedings of the IASTED International Conference on Communication, Network and Information Security (CNIS'2005), Phoenix, Arizona, USA, November 14-16, 2005, pp. 13-18.

III. Haataja K.: *Bluetooth Network Vulnerability to Disclosure, Integrity and Denial-of-Service Attacks*. Proceedings of the Annual Finnish Data Processing Week at the University of Petrozavodsk (FDPW'2005), Advances in Methods of Modern Information Technology, Vol. 7, Petrozavodsk, 2006, pp. 63-103.

IV. Hassinen M., Hyppönen K., and Haataja K.: *An Open, PKI-Based Mobile Payment System*. Proceedings of the ACM/IEEE/Springer International Conference on Emerging Trends in Information and Communication Security (ETRICS'2006), LNCS, Vol. 3995, Springer-Verlag, June 6-9, 2006, pp. 86-100.

V. Haataja K.: *Three Practical Bluetooth Security Attacks Using New Efficient Implementations of Security Analysis Tools*. Proceedings of the IASTED International Conference on Communication, Network and Information Security (CNIS'2007), Berkeley, California, USA, September 24-26, 2007, pp. 101-108.

VI. Haataja K.: *New Practical Attack Against Bluetooth Security Using Efficient Implementations of Security Analysis Tools*. Proceedings of the IASTED International Conference on Communication, Network and Information Security (CNIS'2007), Berkeley, California, USA, September 24-26, 2007, pp. 134-142.

VII. Hyppönen K. and Haataja K.: *"Niño" Man-In-The-Middle Attack on Bluetooth Secure Simple Pairing*. Proceedings of the IEEE Third International Conference in Central Asia on Internet, The Next Generation of Mobile, Wireless and Optical Communications Networks (ICI'2007), Tashkent, Uzbekistan, September 26-28, 2007.

VIII. Haataja K.: *New Efficient Intrusion Detection and Prevention System for Bluetooth Networks*. Proceedings of the ACM International Conference on Mobile, Wireless MiddleWare, Operating Systems, and Applications (Mobilware'2008), Innsbruck, Austria, February 12-15, 2008.

IX. Haataja K. and Hyppönen K.: *Man-In-The-Middle Attacks on Bluetooth − a Comparative Analysis, a Novel Attack, and Countermeasures*. Proceedings of the IEEE Third International Symposium on Communications, Control and Signal Processing (ISCCSP'2008), St. Julians, Malta, March 12-14, 2008, pp. 1096-1102.

X. Haataja K.: *Further Classification of Bluetooth-enabled Ad-hoc Networks Depending on a Risk Analysis Within Each Classified Group*. Proceedings of the IEEE Seventh International Conference on Networking (ICN'2008), Cancun, Mexico, April 13-18, 2008, pp. 232-237.

Author's Contribution

This dissertation is the result of research carried out at the Department of Computer Science at the University of Kuopio.

Publication I presents the most serious Bluetooth security threats and proposes countermeasures against these attacks. Moreover, our Bluetooth security laboratory environment and an experimental Bluetooth security attack are described. The article and the research work were done solely by the author.

Publication II introduces new enhanced implementations of two existing Bluetooth security analysis tools and two new attacks against Bluetooth security as well as the most feasible countermeasures. Moreover, a laboratory environment in which these new attacks against Bluetooth security have been performed is described. The article and the research work were done solely by the author.

Publication III discusses the real vulnerability of Bluetooth networks in respect of Disclosure, Integrity and Denial-of-Service attacks. In addition, very well-known and less well-known attacks against Bluetooth security are described, and the real impact of each attack together with the most feasible countermeasures are evaluated. Moreover, some experimental attacks against Bluetooth security are demonstrated. The article and the research work were done solely by the author.

Publication IV proposes a novel mobile payment system and refines the protocols it is based on. The system is open to all merchants, financial institutions and mobile users. The proposed system can be seen as an example of a real-world Bluetooth-enabled system that uses application layer key exchange and encryption methods to secure communication on top of the existing Bluetooth security measures. Hassinen wrote the implementation details of the paper. The protocols were jointly refined by Hassinen and Hyppönen. The Bluetooth and NFC parts were written by Haataja. The general planning and structure of the article were jointly done by Hassinen, Hyppönen and Haataja.

Publication V describes three practical Bluetooth security attacks using new efficient implementations of security analysis tools. In addition, the article demonstrates with experimental figures that by using these security analysis tools, attacks against Bluetooth devices become practical. Moreover, countermeasures that render these attacks impractical,

although without totally eliminating their potential danger, are proposed. The article and the research work were done solely by the author.

Publication VI presents a new practical attack against Bluetooth security using efficient implementations of security analysis tools. In addition, the article demonstrates with experimental figures that by using our security analysis tools, attacks against Bluetooth-enabled printers become practical. Moreover, countermeasures that render these attacks impractical, although without totally eliminating their potential danger, are proposed. The article and the research work were done solely by the author.

Publication VII proposes a novel Man-In-The-Middle attack on Bluetooth Secure Simple Pairing. The attack is based on the falsification of information sent during the input/output capabilities exchange. Moreover, the article proposes countermeasures that render the attack impractical, although without totally eliminating its potential danger. The article and the research work were jointly done by Hyppönen and Haataja.

Publication VIII investigates how various Bluetooth security attacks in progress can be prevented and stopped by monitoring communication to discover such attacks. Moreover, a new efficient Intrusion Detection and Prevention System for Bluetooth networks to prevent attacks in progress is proposed. The proposed system is based on the set of rules that are used to identify strange communication behaviour in Bluetooth devices. The article and the research work were done solely by the author.

Publication IX provides a comparative analysis of the existing Man-In-The-Middle attacks on Bluetooth. In addition, a novel Bluetooth Man-In-The-Middle attack against Bluetooth-enabled printers that support Secure Simple Pairing is proposed. The attack is based on the fact that the security of the protocol is likely to be limited by the capabilities of the least powerful or the least secure device type. Moreover, improvements to the existing Bluetooth Secure Simple Pairing are proposed in order to make it more secure. The article and the research work were jointly done by Haataja and Hyppönen.

Publication X presents further classification of Bluetooth-enabled ad-hoc networks and their security procedures/requirements depending on a risk analysis within each classified group. In addition, the breaches and damage that can be inflicted by various attacks in such scenarios are described. Moreover, security procedures to prevent malicious Bluetooth devices from stealing information from other Bluetooth devices are devised. The article and the research work were done solely by the author.

Contents

1 INTRODUCTION

The use of wireless communication systems and their interconnections via networks have grown rapidly in recent years. Because RF (Radio Frequency) waves can penetrate obstacles, wireless devices can communicate with no direct line-of-sight between them. This makes RF communication easier to use than wired or infrared communication, but it also makes eavesdropping easier. Moreover, it is easier to disrupt and jam wireless RF communication than wired communication. Because wireless RF communication can suffer from these new threats, additional countermeasures are needed to protect against them.

Excluding mobile phone-related data transfer, there are three popular wireless data transfer technologies widely used all over the world: Bluetooth [Blu07a], WLAN (Wireless Local Area Network) [IEE07] and IrDA (Infrared Data Association) [Inf05]. Bluetooth and WLAN are wireless RF communication systems, while IrDA is a wireless infrared communication system. Our work focuses on the security of Bluetooth technology.

Bluetooth is a technology for short range wireless data and realtime two-way voice transfer providing data rates up to 3 Mb/s. Almost any device can be connected to another device by using Bluetooth. Many kinds of Bluetooth devices, such as mobile phones, headsets, PCs, laptops, printers, mice and keyboards, are widely used all over the world. Already in 2006, the one billionth Bluetooth device was shipped [Blu06a], and the volume is expected to increase rapidly in the near future. According to the Bluetooth SIG (Special Interest Group), the target volume for 2010 is as high as two billion Bluetooth devices. Therefore, it is very important to keep Bluetooth security issues up-to-date.

As an interconnection technology, Bluetooth has to address all traditional security problems, well known from distributed networks [And01]. In addition, security issues in wireless ad-hoc networks are much more complex than those of more traditional wired or centralized wireless networks. Moreover, Bluetooth networks are formed by radio links, which means that there are additional security aspects whose impact is not yet well understood.

The aim of our work is to evaluate security threats in Bluetooth-enabled systems. Our research work can be roughly divided into four parts. First, weaknesses of Bluetooth security are studied and a Bluetooth security laboratory environment for implementing Bluetooth security attacks in practice has been built. Secondly, different types of attacks against Bluetooth security are investigated and the feasibility of some of them are demonstrated in our research laboratory. Countermeasures against each type of attack are also proposed. Thirdly, new Bluetooth security analysis tools are implemented, new attacks against Bluetooth security are presented, and countermeasures that render these attacks impractical are proposed. Finally, a comparative analysis of the existing MITM (Man-In-The-Middle) attacks on Bluetooth is provided, a novel system for detecting and preventing intrusions in Bluetooth networks is described, and a further classification of Bluetooth-enabled ad-hoc networks is provided.

The rest of the thesis is organized as follows. Chapter 2 gives an overview of Bluetooth technology. Different Bluetooth versions are explained and a brief survey of the future of Bluetooth is given. Bluetooth communication, special characteristics of Bluetooth, and Bluetooth protocols are also explained.

Chapter 3 is a digest of various topics in cryptography and network security which are relevant in the context of this thesis. An overview of symmetric cryptography is given and some widely used symmetric encryption methods are explained. Basic notions of public-key cryptography and some widely used public-key encryption methods are also presented. Moreover, an overview of message authentication and hash functions is given. Digital signatures, MITM attacks, and public key certificates are also explained. To set the scene for the main content of the thesis, network security issues are discussed and some guidelines for successfully implementing data security policy are given.

Chapter 4 gives an overview of Bluetooth security, and the existing Bluetooth security architecture is explained. Moreover, vulnerabilities of Bluetooth networks are discussed, and reasons for these vulnerabilities are explained.

Chapter 5 gives a detailed description of our Bluetooth security laboratory in which several Bluetooth security attacks were demonstrated. Bluetooth equipment used in several security attacks are also introduced.

Chapter 6 is dedicated to our practical experiments. It describes our new Bluetooth security analysis tools and introduces new attacks against Bluetooth security, which we were able to devise and carry out in practice by using these tools. On the positive side, the chapter proposes countermeasures against these attacks, and it also provides Bluetooth vulnerability evaluation. In addition, a comparative analysis of the existing MITM attacks on Bluetooth is provided, including our two MITM attacks on Bluetooth SSP (Secure Simple Pairing). Moreover, our novel Intrusion Detection and Prevention System for Bluetooth Networks is described, and a further classification of Bluetooth-enabled ad-hoc networks depending on a risk analysis within each classified group is provided.

Finally, Chapter 7 concludes the thesis and sketches future work.

2 INTRODUCTION TO BLUETOOTH TECHNOLOGY

Bluetooth operates at 2.4 GHz frequency in the free ISM-band (Industrial, Scientific, and Medical) by using frequency hopping. Bluetooth frequency hopping uses a maximum of 79 different Baseband frequencies to avoid channels that suffer from interference. It also enables a large number of Bluetooth devices to operate in the same 2.4 GHz ISM-band.

Section 2.1 gives a brief overview of different Bluetooth versions. Bluetooth communication is described in Section 2.2. Section 2.3 explains the special characteristics of Bluetooth medium. Bluetooth protocols are outlined in Section 2.4.

2.1 Bluetooth versions

The preliminary work for developing Bluetooth technology started in 1994, when Ericsson began researching the possible ways of replacing cables between accessories and mobile phones with wireless links. Ericsson quickly realized the potential market for Bluetooth products, but worldwide cooperation was needed for the products to succeed. Therefore, the *Bluetooth SIG* [Blu07b] was founded in February 1998 by Ericsson, Nokia, IBM, Intel and Toshiba. 3Com, Lucent, Microsoft and Motorola joined the Bluetooth SIG in December 1999. These nine members of the Bluetooth SIG are known as the *Bluetooth SIG Promoters*. They are responsible for upper-level SIG administration, and for providing manpower to run the marketing, qualification and legal processes. Currently, the Bluetooth SIG has over 10000 member companies. [Blu07b, Mor02]

The first public version of Bluetooth specification, *Bluetooth 1.0A* [Blu99a], was released in July 1999. Many device manufacturers had difficulties in making their Bluetooth 1.0A compatible products interoperable. Therefore, the *Bluetooth 1.0B* specification [Blu99b] was released later in the same year (December 1999) to fix the interoperability problems. The *Bluetooth 1.1* specification [Blu01] was released in February 2001. It fixed many errors that were found in the Bluetooth 1.0B specification and added support for unencrypted communication as well as support for RSSI (Received Signal Strength Indicator). RSSI is a

measurement of the received radio signal strength that is used for controlling power in Bluetooth devices. It can also be used for Bluetooth positioning purposes, for example.

The *Bluetooth 1.2* specification [Blu03] was released in November 2003. It included major improvements such as: [Blu03]

- *AFH (Adaptive Frequency Hopping)*: AFH further improves the original Bluetooth frequency hopping method FHSS (Frequency Hopping Spread Spectrum) by avoiding the use of channels that suffer from interference. A maximum of 59 "bad" channels can be switched off during the communication session, i.e. only 20 different "good" channels are required. AFH also gives higher transmission speeds in practice by decreasing the need for retransmissions.

- *eSCO (extended Synchronous Connection-Oriented)*: eSCO improves the voice quality of Bluetooth audio links by allowing retransmissions of corrupted packets.

- *Optional QoS (Quality-of-Service) improvements*: QoS improvements further enhance the capabilities for error detection, flow controlling and synchronization.

The *Bluetooth 2.0+EDR (Enhanced Data Rate)* specification [Blu04a] was released in November 2004. The main improvement was the introduction of *EDR,* which provides data rates up to 3 Mb/s. The original Bluetooth data rate before EDR was 1 Mb/s. According to the Bluetooth SIG, EDR has the following effects on Bluetooth communication: [Blu04a, Blu04b]

- Three times faster transmission speed (up to 10 times in certain cases).

- Lower power consumption through a reduced duty cycle.

- Simplification of multilink scenarios due to more available bandwidth.

- Further improved BER (Bit-Error-Rate) performance.

New Bluetooth versions are backward-compatible with the older versions. The latest public version of Bluetooth specification, *Bluetooth 2.1+EDR* [Blu07a], was released in July 2007. It provides many improvements such as: [Blu07a]

- *Encryption Pause Resume*: Encryption Pause Resume will further enhance security by allowing encrypted links to change their encryption keys periodically. Master-slave role switches (Section 2.2 explains the master-slave relationship) will also be possible on an encrypted link.

- *Extended Inquiry Response*: Extended Inquiry Response will provide more information, such as the name of the device and a list of supported services, during the inquiry procedure, allowing better device filtering before the connection is established.

- *SSP*: SSP (see Section 4.1) radically improves the Bluetooth pairing experience by simplifying the pairing process from the user's point of view. It will also increase the strength of security by providing the protection against both passive eavesdropping attacks and MITM attacks (active eavesdropping attacks). The Bluetooth SIG expects that this feature will significantly increase the use of Bluetooth technology.

- *NFC (Near Field Communication) [NFC08] as an OOB (Out-Of-Band) channel*: In order to provide protection against MITM attacks, SSP either uses NFC as an OOB channel or asks the user to compare two six-digit numbers. Such a comparison can also be thought as an OOB channel which is not controlled by the MITM. However, when NFC radio interface is available, SSP supports the automatic creation of secure Bluetooth connections.

- *Sniff Subrating*: Sniff Subrating will further reduce the power consumption of Bluetooth devices. For example, it will increase the battery life of HID (Human Interface Devices) devices, such as mice and keyboards, by 3 to 10 times compared with the battery life times of older Bluetooth HID devices.

- *QoS improvements*: QoS improvements will further enhance the quality of audio and video transmissions.

The next version of Bluetooth specification, currently codenamed *Seattle* [Hol06, Wal05] (also referred to as *Bluetooth 3.0*), is expected to enable very fast data transfer rates (up to 480 Mb/s) by adopting UWB (Ultra-Wideband) radio technology [Blu06b]. In addition, Seattle will provide low-power idle modes of Bluetooth to save batteries. Moreover, Seattle will provide ultra low power Bluetooth by including Wibree [Nok08] as a part of the specification. Expected use cases for ultra low power Bluetooth include sports sensors, wrist watches, wireless keyboards, toys and medical devices [Nok08]. Seattle is also expected to further enhance the security of Bluetooth by including AES (Advanced Encryption Standard; see Section 3.1). Seattle is expected to be released by the Bluetooth SIG in 2009. [Blu06b, Hol06, Wal05]

The combination of a radio using little power in idle mode and a high data rate radio for transmitting bulk data could be the start of software radios. Therefore, Bluetooth versions after Seattle will provide an excellent signalling channel for enabling the software radio concept. *Software radio* is the technique of getting code as close to the antenna as possible, i.e. radio hardware issues are turned into software issues. The main idea in software radio is that software defines the transmitted waveforms and it also demodulates the received waveforms. In traditional radios, the processing is done with analog circuitry or with analog circuitry combined with digital chips.

Figure 1 illustrates Bluetooth device shipments so far (years 2000-2007) and the Bluetooth SIG's near-future predictions (years 2008-2010) [Blu06a, Blu08a]. It is worth noting that Bluetooth 2.1+EDR devices have only been available in the mass markets since the first half of 2008. It means that at the end of 2007, there were approximately 1.9 billion (1.9×10^9) Bluetooth devices in use without SSP's improved security features. Moreover, it is expected that consumers will be able to buy these older Bluetooth devices for many years to come. In fact, it can take even a decade before the existing Bluetooth devices will be replaced by new and more secure ones.

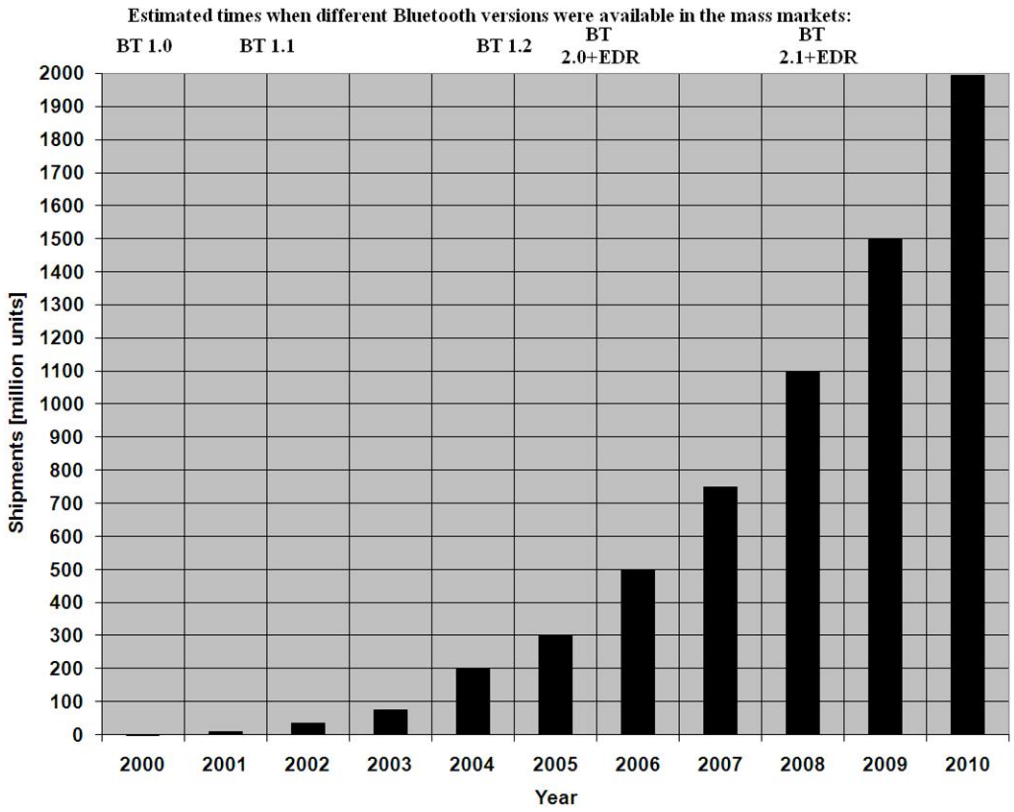

Figure 1. Bluetooth device shipments so far and near future predictions. [Blu06a, Blu08a]

2.2 Bluetooth communication

Connection types define the ways Bluetooth devices can exchange data. Bluetooth has three connection types: ACL (Asynchronous Connection-Less), SCO (Synchronous Connection-Oriented) and eSCO.

ACL links are for symmetric (maximum of 1306.9 kb/s for both directions) or asymmetric (maximum of 2178.1 kb/s for send and 177.1 kb/s for receive) data transfer. Retransmission of packets is used to ensure the integrity of data.

SCO links are symmetric (maximum of 64 kb/s for both directions) and are used for transferring realtime two-way voice. Retransmission of voice packets is not used. Therefore, when the channel BER is high, voice can be distorted.

eSCO links are also symmetric (maximum of 864 kb/s for both directions) and are used for transferring realtime two-way voice. Retransmission of packets is used to ensure the integrity of data (voice). Because retransmission of packets is used, eSCO links can also carry data packets. However, they are mainly used for transferring realtime two-way voice. Bluetooth 1.2 (or later) devices can use eSCO links, but they must also support SCO links to provide backward-compatibility.

Bluetooth devices that communicate with each other form a *piconet*. The device that initiates a connection is the piconet *master*. One piconet can have a maximum of seven active *slave* devices and one master device. All communication within a piconet goes through the piconet master. The clock of the piconet master and frequency hopping information are used to synchronize the piconet slaves with the master. Two or more piconets together form a *scatternet*, which can be used to eliminate Bluetooth range restrictions. A scatternet environment requires that different piconets must have a common device, called a *scatternet member*, to relay data between the piconets. Figure 2 illustrates Bluetooth communication network topology when either ACL or SCO/eSCO links are used. [Blu07a]

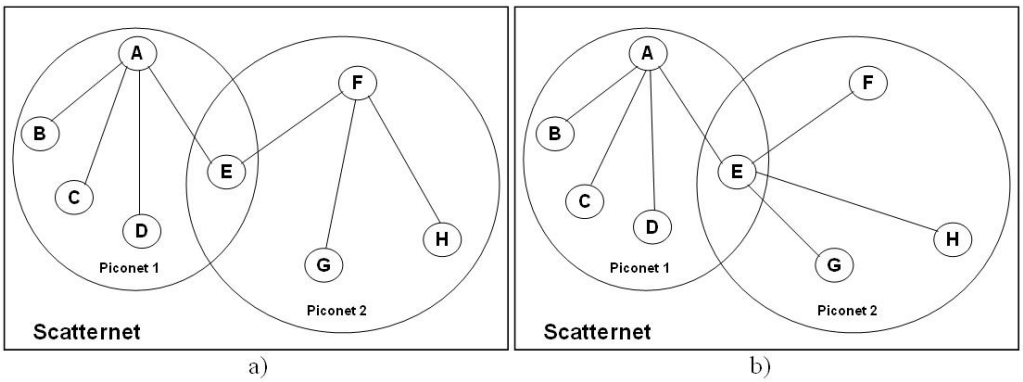

Figure 2. a) Bluetooth topology when ACL links are used. b) Bluetooth topology when SCO or eSCO links are used. [Blu07a]

When ACL links are used (see Figure 2a), a scatternet member can be a slave in many piconets. Device A is the master for piconet 1, and devices B, C, D and E are equal slaves for that piconet. Device F is the master for piconet 2, and devices E, G and H are equal slaves for that piconet. Piconets 1 and 2 together form a scatternet. Piconets 1 and 2 are not synchronized with each other and the scatternet member must multiplex between these two piconets.

When SCO or eSCO links are used (see Figure 2b), the scatternet member must be a slave for piconet 1 and master for piconet 2. If, for example, master A's clock runs at slightly slower rate than the clock of the common device E, master A's timeslots are drifting slowly to the right. To avoid an eventual overlap of timeslots, the common device E must periodically delay the exchange of voice packets by a pair of timeslots. Only the master device is allowed to delay the exchange of voice packets for SCO or eSCO links. Device A is the master for piconet 1, and devices B, C, D and E are equal slaves for that piconet. Device E is the master for piconet 2, and devices F, G and H are equal slaves for that piconet. Piconets 1 and 2 together form a scatternet.

2.3 Special characteristics of the Bluetooth medium

Bluetooth is a wireless RF communication system using mainly omnidirectional antennas. Communication with other Bluetooth devices is possible within the range, and no direct line-of-sight between the communicating Bluetooth devices is required. This capability makes Bluetooth communication much easier to use than the traditional cable-based communication or very short range direct line-of-sight infrared communication, but on the other hand it also makes eavesdropping much easier.

The design goals for Bluetooth technology have been simplicity, compatibility, inexpensive and compact microchips, fast data transfer, globality, secure communication, and low power consumption. Simplicity means that a Bluetooth device must be as easy as possible to use. Compatibility means manufacturer-independent interoperability between different Bluetooth devices, and it also means backward-compatibility with older Bluetooth versions (see Section 2.1). Bluetooth microchips are also very compact (roughly 5 mm × 5 mm of size) and cheap

(roughly $2.50 per microchip [Tan06]). The latest public version of the Bluetooth specification, Bluetooth 2.1+EDR, supports data rates up to 3 Mb/s. The next version of the Bluetooth specification (Seattle) is expected to provide data rates up to 480 Mb/s. Globality means that Bluetooth can be used all over the world using the same free ISM-band. Bluetooth has built-in security measures at the link level to provide the secure communication for the piconet. Bluetooth microchips also have low power consumption, which is why they are widely used in many different kinds of mobile devices. Moreover, Seattle will provide ultra low power Bluetooth by including Wibree as a part of the specification. This will allow future Bluetooth devices to connect to an entire new range of tiny battery-powered devices.

There are three Bluetooth device classes: class 1, class 2 and class 3. The maximum transmit powers for *class 1*, *class 2* and *class 3* devices are 100 mW (20 dBm, i.e. 20 decibels relative to one milliwatt), 2.5 mW (4 dBm), and 1 mW (0 dBm) respectively. According to the Bluetooth specification [Blu07a], the reference sensitivity level of a Bluetooth device has to be -70 dBm or better.

The range of Bluetooth devices depends on the class of devices at both ends, the sensitivity levels at both ends, and the level of obstacles. The quantity n is the so-called *PL (Path Loss) exponent* that can be adjusted to account for the amount of clutter in the path between the transmitter and the receiver. The level of obstacles can be roughly divided into four categories: *none* (a free space without clutter in the transmit-receive path; $n=2.0$), *light* (a lightly cluttered path such as an office environment with moveable walls; $n=2.5$), *moderate* (a moderately cluttered path such as an office environment with fixed walls; $n=3.0$), or *heavy* (a heavily cluttered path in which the density of the materials used in the building's construction is very high; $n=4.0$). [Mor02]

The most common case is a moderate level of obstacles, which is why the Bluetooth specification promises that the range of a Bluetooth device is from 10 meters (class 3 device) to 100 meters (class 1 device) indoors when the level of obstacles is moderate. For application, as well as from the security analysis point of view, it is interesting to know how long is the range of Bluetooth devices.

Several assumptions must be made before proceeding with the range calculations. These assumptions reflect the typical characteristics of Bluetooth devices and can be summarized as follows. First, let us assume that a Bluetooth transmitter's (TX) power is either 0 dBm (class 3 device) or 20 dBm (class 1 device). The most common TX power for Bluetooth devices is 0 dBm (1 mW). Secondly, let us also assume that a Bluetooth receiver's (RX) sensitivity level is either -70 dBm (standard sensitivity level) or -80 dBm (enhanced sensitivity level). Finally, let us assume that Bluetooth transmit and receive antennas each have a gain of 0 dBi (decibels relative to an isotropic source). Table 1 presents *the range of Bluetooth devices* with these prerequisites by using the formula $d=10^{(PL-40)/(10n)}$, where d is the range of Bluetooth devices, PL is the Path Loss value, and n is the PL exponent. A more precise definition of the range calculations can be found in [Mor02].

Level of obstacles:	n:	TX power (dBm):	RX sensitivity (dBm):	PL:	Range (m):
None	2.0	0	-70	70	32
None	2.0	0	-80	80	100
None	2.0	20	-70	90	316
None	2.0	20	-80	100	1000
Light	2.5	0	-70	70	16
Light	2.5	0	-80	80	40
Light	2.5	20	-70	90	100
Light	2.5	20	-80	100	251
Moderate	3.0	0	-70	70	*10*
Moderate	3.0	0	-80	80	*22*
Moderate	3.0	20	-70	90	*46*
Moderate	3.0	20	-80	100	*100*
Heavy	4.0	0	-70	70	6
Heavy	4.0	0	-80	80	10
Heavy	4.0	20	-70	90	18
Heavy	4.0	20	-80	100	32

Table 1. The range of Bluetooth devices. [Mor02]

Bluetooth devices can form ad-hoc networks of several devices in which no fixed infrastructure is needed. This can be very useful in, for example, meetings where all participants have Bluetooth-compatible laptops which can share files with each other without

a traditional cable-based interconnection network. On the other hand, security issues in ad-hoc networks are much more complex than those in more traditional wired or centralized wireless networks.

2.4 Protocols

A Bluetooth protocol stack is illustrated in Figure 3. Protocols below the *HCI (Host Controller Interface)* are built-in to the Bluetooth microchip, and protocols above the HCI are located as a part of the host device's software package. A HCI is needed between the hardware and software protocols. The purpose of the HCI is to enable the manufacturer-independent combining of Bluetooth chips (Host Controller) and the actual host device. The HCI takes care of security communication between the host and the Bluetooth module.

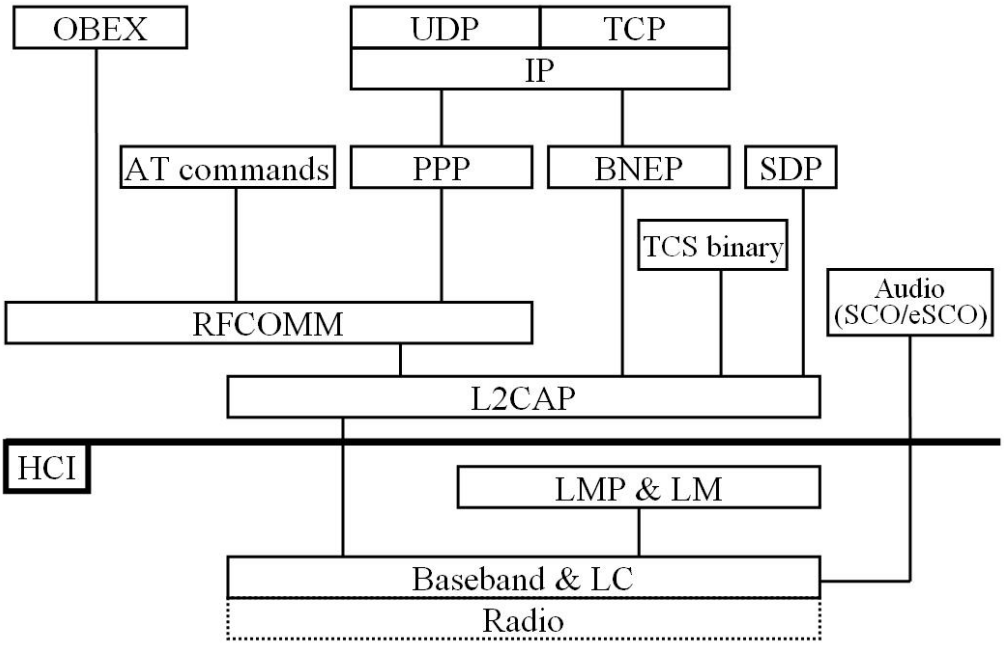

Figure 3. Bluetooth protocol stack. [Blu07a]

Baseband and *LMP (Link Manager Protocol)* together enable the physical RF connection. The *LC (Link Controller)* is a state machine that defines the current state of the Bluetooth

device. A Bluetooth device can be in low-power mode for saving batteries, in the connected state for normal piconet operation, or in the paging state for the master to bring new slaves to the piconet, for example. The LC has a pseudorandom number generation capability, methods for managing security keys, and the capability for providing the mathematical operations needed for authentication and encryption.

The *LM (Link Manager)* acts as a liaison between the application and the LC on the local device, and it also communicates with the remote LM via PDUs (Protocol Data Units) using the LMP, i.e. the LM communicates with three different entities during a Bluetooth session: the local host through the HCI, the local LC (local operations), and the remote LM (link configuration, link information, and link management operations). The PDU is acknowledged at the Baseband level, but it is acted upon by the LM. The local LM usually resides on the Bluetooth module as a complete host-module implementation. The remote LM can be defined as the LM at the other end of the Bluetooth link. The LM also has several commands for handling security issues. [Blu07a, Mor02]

SCO and eSCO links are used for transferring realtime two-way voice (see Section 2.2). They are established directly from the Baseband level, so the overhead of upper layer protocols does not cause any delays for realtime two-way voice connections. Four packet types have been defined for SCO links, whereas eSCO links support seven packet types [Blu07a]. One of these 11 packet types, SCO link's HV1 (High-quality Voice 1), is really interesting from the security point of view, because one single HV1 SCO link reserves all Bluetooth piconet resources and therefore makes various DoS (Denial-of-Service) attacks possible (see Subsection 4.2.3 and Section 6.9).

The *L2CAP (Logical Link Control and Adaptation Protocol)* is a software module that normally resides on the host. It fits upper layer protocols to the Baseband, i.e. it acts as a conduit for data on the ACL link between the Baseband and host applications. The L2CAP also offers *CO* (*Connection-Oriented*; from master to one slave and from slave to master) and *CL* (*Connection-Less*; from master to multiple slaves) services, and it is defined only for ACL links. Lower layer protocols do not have to know how layers above the L2CAP work and vice

versa. The L2CAP can initiate security procedures when a CO or a CL channel connection attempt is made. [Mor02]

The *SDP (Service Discovery Protocol)* is used to find the services of Bluetooth devices in the range. *RFCOMM (Radio Frequency Communication)* emulates serial ports over the L2CAP, and therefore it is possible to use existing serial port applications via Bluetooth.

The *OBEX (Object Exchange Protocol)* [Inf03] is used to exchange objects, such as calendar notes, business cards and data files, between devices by using the client-server model. The OBEX supports six simple and self-explanatory operations: Connect (choose your partner, negotiate capabilities and establish connection), Disconnect (terminate connection), Put (push objects to the server), Get (pull objects from the server), Abort (abort an object exchange that is in progress), and SetPath (set server's directory path to a new value).

The *TCS (Telephony Control protocol Specification)* binary defines the call control signalling for the establishment/release of speech and data calls between Bluetooth devices. It also provides functionality for exchanging signalling information that is unrelated to ongoing calls. Many *AT commands* are also supported for transmitting control signals for telephony control.

The *BNEP (Bluetooth Network Encapsulation Protocol)* is used to provide networking capabilities for Bluetooth devices. It allows *IP (Internet Protocol)* packets to be carried in the payload of L2CAP packets. The IP is a network layer protocol in the *TCP/IP (Transmission Control Protocol/Internet Protocol)* protocol suite. *TCP* and *UDP (User Datagram Protocol)* are transport layer core protocols used in the TCP/IP protocol suite. *PPP (Point-to-Point Protocol)* can also be used to provide TCP/IP networking capabilities for Bluetooth devices, but it is slower, i.e. it works over RFCOMM whereas BNEP works directly over the L2CAP, and therefore PPP is rarely used now.

More information about Bluetooth technology and its protocols can be found in [Blu07a, GPS04, Haa00, Mor02].

3 CRYPTOGRAPHY AND NETWORK SECURITY BASICS

Information can be stored in many different forms: for example, in people's memories, printed on a paper, in digital form, or as a physical artifact. Our work focuses only on digital information (data) and its security in wireless Bluetooth networks. Dozens of Bluetooth security attacks (see Subsections 4.2.1-4.2.4) have been developed by various researchers and hackers in recent years, and Bluetooth security threats are finally (but slowly) being taken more seriously by some Bluetooth device manufacturers, by some Bluetooth device users, and by the Bluetooth SIG. To enable better understanding of the special characteristics of Bluetooth security, we first provide a general overview of cryptography and network security (see Sections 3.1-3.5).

The use of computer systems and their interconnections via networks have grown rapidly in recent decades. This has increased the dependence of both individuals and organizations on the information stored and communicated using these systems. Therefore, it is very important to protect data and resources from disclosure, to guarantee the authenticity of data and messages, and to protect systems from network-based attacks. Fortunately, the disciplines of cryptography and network security have matured, which has led to the development of practical and readily available applications to enforce network security. [Sta03]

Data security means that data, data processing and data transfer are secure. *Computer security* is the generic name for the collection of tools designed to protect data against hackers and attackers. *Network security* is needed to protect data during transmission. [Pfl03, Sta03]

The general aims of data security are confidentiality, integrity, nonrepudiation, and availability. *Confidentiality* means the protection of data from unauthorized disclosure. *Integrity* means the assurance that data received is exactly as sent by an authorized entity. *Nonrepudiation* provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication. *Availability* means that data can be used when needed, data can be obtained fast enough, and data can be used easy enough. [IET00, ITU91, Pfl03, Sta03]

An attacker can use many different methods against security: some examples are interruption of service, interception of data, modification of data, and fabrication of data. *Interruption of service* can be achieved by destroying hardware, for example. Such an attack breaks availability. *Interception of data* by some kind of eavesdropping is an attack against confidentiality. *Modification of data* compromises data integrity. *Fabrication of data* threatens authenticity. [Pfl03, Sta03]

Cryptology involves *cryptography* (also referred to as *encryption methodology*) and *cryptanalysis* (also referred to as *analysis of the encrypted messages*). Techniques used for decrypting a message without any knowledge of the encrypting details fall into the area of cryptanalysis. [Pfl03, Sch96, Sta03]

Already in antiquity, encryption was used in warfare and diplomacy. The ancient encryption methods are insufficient nowadays, but some of their ideas are utilized in the new encryption methods. Encryption schemes are used to keep messages or stored data secret. *Symmetric encryption* (also referred to as *conventional encryption*, *single-key encryption* or *secret key encryption*) was the only type of encryption in use prior to the development of *public-key encryption* (also referred to as *asymmetric encryption*). Symmetric encryption methods in general use a *secret encryption key* for both encryption and decryption. Public-key encryption methods use a *public key* for encryption and a *private key* for decryption. Even though public-key encryption methods have some advantages in many applications, symmetric encryption methods are not obsolete. Typically, public-key cryptography is used for authentication and session key exchange, while symmetric cryptography is efficiently used for the encryption of data streams. [Buc01, Pfl03, Sch96, Sta03]

Section 3.1 gives an overview of symmetric cryptography and explains some widely used symmetric encryption methods. An overview of public-key cryptography is given and some widely used public-key encryption methods are explained in Section 3.2. Section 3.3 gives an overview of message authentication and hash functions. Digital signatures, MITM attacks, and public key certificates are explained in Section 3.4. Section 3.5 discusses network security, and it also gives some guidelines for successfully implementing data security policy.

3.1 Symmetric cryptography

The cryptosystem is called *symmetric* if the encryption key is equal to the decryption key, or if the decryption key can be easily computed from the encryption key. If, for example, Alice and Bob use a symmetric cryptosystem, they must exchange the secret key before they start communicating. Therefore, secure key exchange is very important. The secret key must be kept secret, because anybody who knows the secret key also knows or can determine the corresponding decryption key. [Buc01, Sta03]

The main concepts of the symmetric encryption scheme are: [Buc01, Pfl03, Sta03]

1. *Plaintext*: The plaintext is the original intelligible message or data that is fed into the encryption algorithm as input.

2. *Encryption algorithm*: The encryption algorithm performs various substitutions and transformations on the plaintext. The process of disguising a message in such a way as to hide its substance is called *encryption* (also referred to as *enciphering*).

3. *Secret key*: The secret key is also input into the encryption algorithm that produces a different output depending on the specific key being used at the time.

4. *Ciphertext*: The ciphertext is a scrambled message produced as an output that depends on the plaintext and the secret key.

5. *Decryption algorithm*: The decryption algorithm is essentially the encryption algorithm run in reverse. It produces the original plaintext by using the ciphertext and the secret key. The process of restoring the plaintext from the ciphertext is called *decryption* (also referred to as *deciphering*).

The security of a symmetric cryptosystem depends on two things: the *strength of the algorithm* and the *length of the key*. There are two general approaches for attacking a symmetric encryption scheme: [Sch96, Sta03]

1. *Cryptanalysis*: Cryptanalytic attacks rely on the nature of the algorithm and perhaps some knowledge of the general characteristics of plaintext or even some sample plaintext-ciphertext pairs. The main aim is to exploit the characteristics of the algorithm in an attempt to deduce a specific plaintext or to deduce the key being used. If the key (or even a part of the key) is discovered, all future and past messages encrypted with that key are compromised.

2. *Brute-force attack*: An attacker tries every possible key value on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible key values must be tried to achieve success.

The amount of information in a message is measured by the *entropy* of that message. The entropy of a message measured in bits is $\log_2 n$, where n is the number of possible meanings in which each meaning is equally likely. The entropy of a message also measures its uncertainty. For example, the entropy of a message indicating the day of the week is $\log_2 7 \approx 2.8$ *bits*, i.e. *000*=Monday, *001*=Tuesday, *010*=Wednesday, *011*=Thursday, *100*=Friday, *101*=Saturday, *110*=Sunday, and *111* is unused. Similarly, a four-number long password that is widely used in various applications, such as in ATMs (Automated Teller Machines) for withdrawing money or in mobile phones for unlocking the phone, achieves only $4 \times \log_2 10 \approx 13.3$ *bits* of entropy. [Sch96]

Shannon postulated already in 1945 (his 1949 paper [Sha49] appeared originally as a classified report in 1945) that a good symmetric encryption method is based on two operations: [Sha49]

- *Diffusion*: Diffusion spreads the influence of a plaintext symbol as evenly as possible in the ciphertext block. Therefore, it is difficult to make statistical conclusions.

- *Confusion*: Confusion aims to make the relation between a key and a ciphertext as complicated as possible.

Based on the ideas of Shannon, Feistel proposed in 1973 that the following principles and parameters should be applied when designing good encryption algorithms: [Fei73]

- *Block size*: A larger block size increases security, but it also decreases speed.
- *Key size*: A larger key size increases security, but it also decreases speed.
- *Rounds*: Multiple rounds increase security.
- *Round function*: A more complex round function makes cryptanalysis more difficult.
- *Subkey generation*: Greater complexity increases security.

An encryption scheme is said to be *computationally secure* if the following two criteria are met: [Sta03]

1. *The cost of breaking the cipher exceeds the value of the encrypted data*: It is important to define the adequate level of security in different situations.
2. *The time required to break the cipher exceeds the useful lifetime of the data*: Data are useless for an attacker when the cipher has been broken.

The *DES (Data Encryption Standard)* is a symmetric encryption method developed by IBM. The standard was accepted in 1977 and the latest version of the standard [NIS99] is from 1999. Originally, DES should have used 128-bit keys, but on the recommendation of the NSA (National Security Agency) the key length of 56 bits was chosen. The DES is a *block cipher* method, because it encrypts data in 64-bit blocks. It is intended to be implemented by hardware. Encryption is done in three phases. First, the initial permutation rearranges the bits to produce the permuted input. Secondly, the 16 rounds of the same function involve both permutation and substitution operations. For each of the 16 rounds, a subkey K_i ($i=1,...,16$) is produced by the combination of a left circular shift and a permutation. The permutation operation is the same for each round, but a different subkey is produced because of the

repeated iteration of the key bits. The output of the last round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the preoutput. Finally, the preoutput is passed through the inverse initial permutation to produce the 64-bit ciphertext. For decryption, the same algorithm is used, but the application of subkeys is reversed. The 56-bit DES is no more secure. It was first broken in 1998 with a specially designed machine, the EFF DES Cracker [Ele98], by exhaustive search in 56 hours. Later in the same year, a distributed.net project [Dis08] managed to solve the task in 22 hours. The insufficient security of the DES has motivated researchers to develop alternatives to it. [Buc01, Pfl03, Sch96, Sta03]

The DES is efficient in hardware implementations and it can be made secure by using *3DES* (*Triple DES*; also referred to as *TDES*) coding [NIS04]. In 3DES, two keys K_1 and K_2 are needed, and they are applied in the following way. The plaintext block is coded by using K_1 and the result is decoded by using K_2. Because K_2 is a "wrong" key, the result is not the original plaintext. This result is coded again by using K_1, which finally produces the result of the 3DES coding. Because 3DES keys have 112 bits, the number of different 3DES keys is quadratic to the number of DES keys. Currently 3DES is the de-facto standard in banking. [NIS04, Sta03]

Blowfish [Sch94a, Sch94b] is a block cipher method developed by Bruce Schneier in 1993. It is freely distributed, unlike DES, which is patented. The main features of Blowfish are the following. Blowfish is fast, because it needs only 18 clock cycles per byte on a 32-bit microprocessor. It is also simple to implement and compact (it needs only 5 kB of memory). Blowfish has an adjustable key length from 32 to 448 bits, which allows a tradeoff between higher speed and higher security. From the original key, eighteen 32-bit subkeys and four 256-element S-boxes (an *S-box*, i.e. a *Substitution box*, substitutes input bits and produces output bits) of 32-bit words are formed. An S-box transforms an 8-bit number (position *0,...,255*) into a 32-bit word (in that position). Blowfish encryption consists of 17 phases. Each phase consists of XOR, addition modulo 2^{32}, and S-box operations. Decryption is identical to encryption, except that 18 subkeys are used in reverse order. Blowfish is considered secure. [Sch94a, Sch94b, Sch96, Sta03]

Ron Rivest has developed a sequence of encryption algorithms called RCi (*i=1,2,3,...*). Let us take a closer look at two of them: *RC4* (*Rivest Cipher 4*; also referred to as *Ron's Code 4* or *ARCFOUR*) [Riv92a] and *RC6* (*Rivest Cipher 6*; also referred to as *Ron's Code 6*) [RRS98].

RC4 encodes data byte-by-byte, i.e. it is a *stream cipher*, not a block cipher. It is a widely used algorithm, because RC4 is used in the WEP (Wired Equivalent Privacy) protocol that is part of the IEEE (Institute of Electrical and Electronics Engineers) 802.11 standard (i.e. the popular WLAN standard), and it is also used in the *SSL (Secure Sockets Layer)* standard that has been defined to protect Internet traffic. The RC4 algorithm was originally secret, but it was leaked to the Internet in 1994 [Cyb94]. RC4 works in the following way. It has an adjustable key length from 1 to 256 bytes (8 to 2048 bits), which is used to initialize a 256-byte state vector *S* with elements *S[i]* (*i=0,1,...,255*). At all times *S* contains a permutation of all 8-bit numbers from 0 through 255. For encryption and decryption, a byte *k* is generated from *S* by selecting one of the 255 entries in a systematic way. As each value of *k* is generated, the entries in *S* are permuted again. To encrypt data, the value *k* is XORed with the next byte of plaintext. To decrypt data, the value *k* is XORed with the next byte of ciphertext. RC4 is essentially a pseudorandom number generator that is initialized with the secret key. Many papers, such as [FlM00, KMP98, MaS01, MiT98], have been published about analyzing methods of attacking against RC4, but these approaches are not practical against RC4 with a reasonable key length such as 128 bits. A more serious problem was reported in 2000 when Jesse Walker showed in his paper [Wal00] that the WEP protocol is insecure. Another paper [FMS01] also demonstrated WEP's shortcomings in 2001, and it made clear that the original WEP protocol was no more secure. In essence, the problem is not with the RC4 itself, but with the way in which keys are generated in the WEP protocol for use as input to the RC4. [Cyb94, FMS01, Riv92a, RRS98, Sch96, Sta03, Wal00]

RC6, from 1998, is the latest of Ron Rivest's encryption algorithms, and it was also his candidate for the AES contest. RC6 is a block cipher derived from *RC5* (*Rivest Cipher 5*; also referred to as *Ron's Code 5)* [Riv94]. RC6 is adjusted by parameters *w* (word length in bits), *r* (number of rounds), and *b* (the length of encryption key in bits). Typical values are *w=32*, *r=20* and *b=128*. Basic operations are addition, subtraction, multiplication modulo 2^w, XOR,

rotation to the left by *log w* bits, and rotation to the right by *log w* bits. RC6 is easy to program and fast. [Riv94, RRS98]

SAFER+ (Secure And Fast Encryption Routine +) [MKK98] was developed by Massey et al. in 1998. It was also submitted as a candidate for the AES contest, but the cipher was not selected as a finalist. SAFER+ is a block cipher with the following main features. It has a block size of 128 bits and three different key lengths (128, 192 and 256 bits). Bluetooth uses SAFER+ with a 128-bit key as an algorithm for authentication and key generation (see Section 4.1). It consists of nine phases (*eight identical rounds* and the *output transformation*) and a *KSA (Key Scheduling Algorithm)* in the following way. KSA produces 17 different 128-bit subkeys. Each round uses two subkeys and a 128-bit input word from the previous round to calculate a 128-bit word that is a new input word for the next round. The last subkey is used in the output transformation, which is a simple bitwise XOR of the last round's output with the last subkey. Although some optimizations for faster breaking of SAFER+ exist (for example, in [ShW05]), it is considered secure. [Blu07a, MKK98, ShW05]

AES [NIS01] was published by the NIST (National Institute of Standards and Technology) in 2001 after the evaluation process of the AES contest. *Rijndael* was the winner of the contest and the NIST selected it as the algorithm for AES. AES is a symmetric block cipher that is intended to replace DES as the approved standard for a wide range of applications, but this process will take many years. The NIST anticipates that 3DES will remain an approved algorithm for the foreseeable future, at least for U.S. government use. The future version of Bluetooth, Seattle (see Section 2.1), is expected to improve the security of Bluetooth by including AES. [NIS01, Pfl03, Sta03, Wal05]

AES encryption consists of 10-14 rounds in which data blocks are processed step-by-step in the following way (except the final round; it is noteworthy that AES decryption is symmetric to AES encryption): [NIS01, Pfl03, Sta03]

1. *Byte substitution*: Byte substitution uses an *S-box* to perform a byte-by-byte substitution of the block.

2. *Row shifting*: Row shifting is a simple permutation.

3. *Column mixing*: Column mixing is a substitution, which makes use of arithmetic over GF(2^8). *Galois Field GF(2^8)* is a finite field of 256 elements, which can be denoted by strings of eight bits or by hexadecimal notation.

4. *Round key adding*: Round key adding is a simple bitwise XOR of the current block with a portion of the expanded key.

The final round of the AES encryption (and AES decryption) is slightly different: [NIS01, Pfl03, Sta03]

1. *Byte substitution*.

2. *Row shifting*.

3. *Round key adding*.

The main features of AES are the following. It is considered secure. AES is very fast and compact (about 1 kB of code). Its block size is multiples of 32 (typically 128 bits) and its key length is also multiples of 32 (typically 128, 192 or 256 bits). Moreover, AES has a very neat algebraic description. [NIS01, Pfl03, Sta03]

3.2 Public-key cryptography

The invention of public-key cryptography in 1976 by Diffie and Hellman [DiH76a, DiH76b] was a major milestone in the history of cryptography. The difficulty of symmetric encryption lies in the distribution of secret keys. In public-key cryptosystems, the encryption key and the decryption key are distinct, and the computation of a decryption key from the encryption key is infeasible. Therefore, the encryption key can be made public. If, for example, Bob wants to receive encrypted messages, he publishes an encryption key and keeps the corresponding decryption key secret. Anyone can use the *public key* (encryption key) to encrypt messages for Bob, but only Bob can decrypt the messages by using his *private key* (decryption key). [Buc01, Pfl03, Sta03]

The main concepts of the public-key encryption scheme are: [Buc01, Pfl03, Sch96, Sta03]

1. *Plaintext*: The plaintext is the original intelligible message or data fed into the encryption algorithm as input.

2. *Encryption algorithm*: The encryption algorithm performs various transformations on the plaintext.

3. *Public key*: The public key is used for encryption.

4. *Private key*: The private key is used for decryption.

5. *Ciphertext*: The ciphertext is a scrambled message produced as an output, which depends on the plaintext and the public key.

6. *Decryption algorithm*: The decryption algorithm accepts the ciphertext and the matching private key, and produces the original plaintext.

As with a symmetric encryption scheme (see Section 3.1), a public-key encryption scheme may be vulnerable to a brute-force attack. The countermeasure is also the same: the key size must be large enough to make brute-force attacks impractical, but small enough for practical encryption and decryption. Another way to attack a public-key encryption scheme is to find a way to compute the private key from the given public key. In this attack, any given algorithm is suspect. The history of cryptanalysis shows that a problem which seems almost impossible to solve from one perspective can have a (simple) solution if the problem is looked at in an entirely different way. [Pfl03, Sch96, Sta03]

Table 2 summarizes and compares the main aspects of symmetric and public-key encryption (see also the descriptions of symmetric and public-key encryption schemes in Sections 3.1 and 3.2). [Sta03]

Symmetric encryption:	Public-key encryption:
Needed to work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key.	Needed to work: 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one).
Needed for security: 1. The key must be kept secret. 2. It must be impossible or at least impractical to decrypt a message if no other information is available. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	Needed for security: 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decrypt a message if no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Table 2. The main aspects of symmetric and public-key encryption. [Sta03]

RSA (Rivest-Shamir-Adleman) was developed in 1977 by Rivest, Shamir and Adleman, and it was published in 1978 [RSA78]. The RSA algorithm works in the following way: [Buc01, Pfl03, RSA78, Sch96, Sta03]

- Choose two large (odd) primes p and q (typically more than 100 decimal digits), and compute the product $n=pq$ when $\phi(n)=(p-1)(q-1)$. The number n is called the *RSA modulus*. In number theory, $\phi(n)$ is used to denote the number m of those numbers less than n in which $gcd(m,n)=1$, i.e. the numbers m and n do not have common divisors. The notation $c=gcd(a,b)$ means that the number c is the *greatest common divisor* of the numbers a and b, iff (if and only if) $a \bmod c = 0$, $b \bmod c = 0$, and there is no greater number fulfilling these conditions.

- Choose an (odd) integer e with $1<e<\phi(n)$ and $gcd(e,\phi(n))=1$. The number e is called the *encryption exponent*. The public encryption key is (e,n).

- Compute an integer d with $1<d<\phi(n)$ and $de \equiv 1 \bmod \phi(n)$ (notation $a \equiv b \bmod m$ *iff* $(a-b) \bmod m = 0$ is used). The private decryption key is (d,n). The number d is called the *decryption exponent*.

- Number x (plaintext) is encrypted by computing $c = x^e \bmod n$, where $x < \phi(n)$.

- Ciphertext c is decrypted by computing $x = c^d \bmod n$.

Without knowing the factorization $n=pq$, there is no obvious way of solving the decryption key (d,n). Breaking an RSA encryption has been a popular sport among researchers. 512-bit encryption *RSA-155* (512 bits is 155 decimal digits) was first broken on 22.8.1999 by a group of researchers [RSA99]. This was accomplished by using 292 workstations for seven months, altogether *35.7 processor years*. Approximately seven million-fold time and 2650 times more memory would be needed for breaking the 1024-bit (309 decimal digits) RSA [RSA04, Sta03]. RSA numbers were originally spaced at intervals of 10 decimal digits between one and five hundred digits, but when computers and algorithms became faster, the naming convention was changed so that the trailing number indicates the number of bits (for example, 640-bit encryption *RSA-640* has 193 decimal digits). *RSA-640* was first broken on 2.11.2005 by Bahr, Boehm, Franke and Kleinjung [RSA05]. According to the submitters, breaking took approximately *thirty 2.2 GHz-Opteron-CPU (Central Processing Unit) years* (over five months of calendar time). RSA Laboratories recommends 1024-bit key for corporate use and 2048-bit key for applications that require extremely high security. In order to break RSA faster, a new faster algorithm for the factorization should be invented. Currently, the best breaking method has been a systematic search of factor candidates. It is also speculated that quantum computation may change the speed of factorization if it becomes available in a few decades. [RSA04, RSA05, RSA99, Sta03]

Diffie-Hellman key exchange [DiH76a, DiH76b] is based on the use of discrete logarithm. Let us assume that Alice and Bob want to use a symmetric encryption system to keep their communication over an insecure channel secret. In order to do that, they must first exchange a common secret key. The Diffe-Hellman key exchange system allows Alice and Bob to use their insecure communication channel for the key exchange. Everyone can listen to the key

exchange, but the information obtained cannot be used for the secret key construction. Bluetooth versions up to 2.0+EDR use a password authenticated multi-party Diffie-Hellman key exchange (see Section 4.1). [Blu07a, Buc01, DiH76a, DiH76b, Sta03]

The Diffie-Hellman key exchange works in the following way: [Buc01, DiH76a, DiH76b, Sta03]

- Choose a prime p and a number $g<p$, which is a primitive root of p. Numbers p and g are public.
- User A chooses the private key $S_A<p$ and computes the public key $J_A = g^{S_A} \bmod p$. Therefore, the whole public key of user A is (p,g,J_A).

The public key can be used for sharing a common secret key in the following way: [Buc01, DiH76a, DiH76b, Sta03]

- User A sends the public key (p,g,J_A) to user B and user B sends (p,g,J_B) to user A.
- User A computes $J_B^{S_A} = g^{S_B S_A}$, and user B computes $J_A^{S_B} = g^{S_A S_B}$.
- $g^{S_A S_B} = g^{S_B S_A}$ is their common secret key (to be used for symmetric encryption), which is difficult for others to invent, even if they know p, g, J_A and J_B.

The Diffie-Hellman problem is the following. A primitive root of p is needed. To find one, we need a method that is faster than trying all numbers $g<p$ and computing all powers of g. If this were possible, breaking the code would also be possible. As long as the Diffie-Hellman problem is difficult to solve, no eavesdropper can figure out the secret key from the publicly known information. [Buc01, Sta03]

As described earlier, the key length for secure RSA use has increased rapidly over recent years. Therefore, applications using RSA also have a heavier processing load than before. *ECC (Elliptic Curve Cryptography)* [Kob87, Mil85] is based on the mathematical properties of elliptic curves, and it appears to offer equal security to RSA for a much smaller key size [JuM97]. Bluetooth 2.1+EDR improves the security of pairing (see Section 4.1) by using

ECDH (Elliptic Curve Diffie-Hellman) public-key cryptography [Blu07a]. ECDH is a key agreement protocol for allowing two communicating parties to establish a common secret key over an unsecured channel. It is a variant of the Diffie-Hellman key exchange protocol using ECC. [Blu07a, Buc01, JuM97, Kob87, Mil85, Sta03]

An elliptic curve over real numbers $\mathbb{R}$ is a set of points (x,y) that satisfy an equation $y^2=x^3+ax+b$, where $x,y,a,b \in \mathbb{R}$. Every elliptic curve also contains an element O, which is called the *point at infinity* (also referred to as the *zero point*). Calculations over real numbers are slow and inaccurate due to rounding error. Therefore, elliptic curves with $x,y,a,b \in \mathbb{R}$ are not usable in practice. Instead, *elliptic groups modulo p* (where p is a prime) are defined in the following way. Two non-negative integers $a,b<p$, which satisfy $4a^3+27b^2 \neq 0 \bmod p$, are chosen. $E_p(a,b)$ denotes the *elliptic group modulo p*, where elements (x,y) are pairs of non-negative integers less than p satisfying both $y^2 \equiv x^3+ax+b \bmod p$ and the point at infinity O. It is worth noting that now the number of points on the elliptic curve is not infinite. Moreover, it is not clear even how to connect these discrete points to make their graph look like a curve. Therefore, the geometrical definition of operations on these points cannot be used. Instead, the algebraic rules can be used for *elliptic curve groups modulo p* for making computations precise. [Buc01, JuM97, Kob87, Mil85]

The ECC cryptosystem can be defined in the following way. The *domain parameters* (known to all participants) are: [JuM97, Kob87, Mil85]

- *A prime number p and parameters a and b defining the elliptic group of points* $E_p(a,b)$.
- *A generator point G on* $E_p(a,b)$: The important criterion for selecting G is that the smallest value of n (n is called the *order of the point G*), for which $nG=O$, be a very large number.

The *private key* is an integer k, where $2 \leq k \leq n-2$. The *public key* is the point $Q=kG$. A key exchange between, for example, Alice and Bob can be performed with ECC as an analogue to a Diffie-Hellman key exchange. It works in the following way. Alice's keys are (k_A,Q_A) and Bob's keys are (k_B,Q_B). Alice computes the secret key $K=k_AQ_B$ and Bob computes the secret

key $K=k_BQ_A$. Both calculations produce the same result, because $K=k_AQ_B=k_A\times(k_BG)=k_B\times(k_AG)=k_BQ_A=K$. Therefore, Alice and Bob can use K for the symmetric encryption of messages, for example, with 3DES, Blowfish or AES. [Buc01, JuM97, Kob87, Mil85]

ECC encryption and decryption can work in the following way, for example. The plaintext message m has to be represented as a point P_m on $E_p(a,b)$. Various straightforward techniques of transforming the message m into coordinates on the elliptic curves exist. When Alice wants to encrypt and send P_m to Bob, she chooses a positive integer k and produces the ciphertext $C_m=\{kG,P_m+kQ_B\}$. Bob can decrypt the ciphertext by calculating $P_m+kQ_B-k_B\times(kG)=P_m+k\times(k_BG)-k_B\times(kG)=P_m$. Finally, Bob decodes the plaintext message m from the point P_m. Although the theory of ECC has been around for some time, it is only recently that ECC products have begun to appear. Therefore, the confidence level in ECC is not yet as high as that in RSA. [Buc01, JuM97, Kob87, Mil85, Sta03]

3.3 Message authentication and hash functions

Authentication is used for assuring that the communicating entity is the one that it claims to be. The best-known example of identification and authentication is the login procedure, where the identity is authenticated by a password. Another example of authentication is the digital signature. Authentication can be used, for example, to protect against *masquerading/ impersonating* (messages are inserted into the network from a fraudulent source), *content modification* (for example, insertion, deletion or transposition), *sequence modification* (modification to a sequence of messages), and *timing modification* (delay or replay of messages). [Buc01, Pfl03, Sta03]

There are many ways to authenticate: [Buc01, Pfl03, Sta03]

- *Encryption*: The whole message encrypted with the secret key of the sender proves the authenticity of the sender, because only the real sender has that key. It also proves the originality of the message, because after encryption nobody else can change and encrypt it.

- *Hash function*: A public hash function forms a constant length code from the message. The hash value is appended to the message and it proves the originality of the message.

- *MAC (Message Authentication Code*; also referred to as *cryptographic checksum)*: MAC is a constant length code formed from the message and the secret key. It is appended to the message to prove the originality of the message.

MD5 (Message-Digest 5) [Riv92b] is a hash function developed by Ron Rivest in 1991. It processes the message in blocks of 512 bits and outputs a code of 128 bits long. MD5 works in the following way. At the beginning, 1 to 512 padding bits are appended so that the length l (greater than 512) is 448 (mod 512). After padding bits, 64 bits of the number $l \bmod 2^{64}$ are appended so that the length of the message is divisible by 512. Intermediate results (and ultimately also the final result) are stored in a buffer that consists of four 32-bit registers. The initial values of the registers are *A=67452301*, *B=EFCDAB89*, *C=98BADCFE* and *D=10325476* in hexadecimal. Now, let the 512 bits of the block be *X[1...16]*. Registers will be updated by the formula *A,B,C,D=f(A,B,C,D,X[1...16],T[1...64])*, where *T[i]* is the 32-bit representation of the integer part of $2^{32}|sin\ i|$ (where i is in radians). The value of f is computed in 16 phases of the form $a=b+((a+g(b,c,d)+X[\rho(i)]+T[i])<<<s)$, where a, b, c and d are the registers A, B, C and D in certain order, depending on the phase. Depending on the phase, g is one of the functions $F(b,c,d)=(b \wedge c) \vee (\neg b \wedge d)$, $G(b,c,d)=(b \wedge d) \vee (c \wedge \neg d)$, $H(b,c,d)= b \oplus c \oplus d$, or $I(b,c,d)=c \oplus (b \vee \neg d)$, where $\oplus$ is XOR, $\wedge$ is AND, $\vee$ is OR, and $\neg$ is NOT. The rotation of bits s positions to the left is denoted as $<<<s$. Addition modulo 2^{32} is denoted as +. Depending on the phase, ρ is one of the functions $\rho_1(i)=i$, $\rho_2(i)=(1+5i) \bmod 16$, $\rho_3(i)=(5+3i) \bmod 16$, or $\rho_4(i)=7i \bmod 16$. Finally, the output (128-bit code) is read in registers A, B, C and D. [Pfl03, Riv92b, Sch96, Sta03]

MD5 is no longer considered secure, because an attacker can generate hash collisions by modifying one or more messages. A *hash collision* is a situation that occurs when two distinct inputs into a hash function produce identical outputs. In August 2004, collisions for the full MD5 were announced [WFL04]. In March 2005, new results appeared in which MD5

collisions were constructed in a few hours on a typical notebook PC [Kli05, LWW05]. MD5 is still used in many applications. For many cases the security of MD5 is appropriate. However, it is highly recommendable to use better hash functions than MD5 when possible.

A 160-bit hash function *SHA-1 (Secure Hash Algorithm 1)* was published in 1995 by NIST [NIS95]. SHA-1 has some resemblance to MD5. It also processes message in 512-bit blocks, but it uses five 32-bit registers, i.e. 5×32 *bits=160 bits*. SHA-1 works in the following way. At the beginning, 1 to 512 padding bits are appended so that the length l (greater than 512) is 448 (mod 512). After padding bits, 64 bits of the number $l \bmod 2^{64}$ are appended so that the length of the message is divisible by 512. The initial values of the five registers are *A=67452301*, *B=EFCDAB89*, *C=98BADCFE*, *D=10325476* and *E=C3D2E1F0* in hexadecimal. The following 80 constants K_i will be used: the integer part of $2^{30}\sqrt{2}$ for $0 \le i \le 19$, the integer part of $2^{30}\sqrt{3}$ for $20 \le i \le 39$, the integer part of $2^{30}\sqrt{5}$ for $40 \le i \le 59$, and the integer part of $2^{30}\sqrt{10}$ for $60 \le i \le 79$. 512-bit block is split in sixteen 32-bit words. New values for the registers are computed in four rounds (round number $j=1,2,3,4$). The output of one round is $A,B,C,D,E=(A,B,C,D,E)+f_j(A,B,C,D,E,W[1...80])$, where *W[i]* is a 32-bit word derived from the 512-bit block. The first 16 *W* words are taken directly from the 512-bit message block. The remaining *W* words are defined in the following way: $W_t=W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}<<<1$ (for $16 \le t \le 79$). Each round consists of 20 phases of the form $A,B,C,D,E=(E+g_t(B,C,D)+(A<<<5)+W_t+K_t),A,(B<<<30),C,D$, where g_t is of the form $f_1=g_t(B,C,D)=(B \wedge C) \vee (\neg B \wedge D)$ for $0 \le t \le 19$, $f_2=g_t(B,C,D)=B \oplus C \oplus D$ for $20 \le t \le 39$, $f_3=g_t(B,C,D)=(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$ for $40 \le t \le 59$, or $f_4=g_t(B,C,D)=B \oplus C \oplus D$ for $60 \le t \le 79$. Finally, the output (160-bit code) is read in registers *A*, *B*, *C*, *D* and *E*. [NIS95, Pfl03, Sch96, Sta03]

Due to the greater number of bits (160 vs. 128 bits) and phases (20 vs. 16 phases), SHA-1 is more secure than MD5. However, collisions on SHA-1 can be found with 2^{63} operations (comparable with a brute-force search with a complexity of 2^{63}) as was demonstrated in August 2005 [WYY05]. The current situation calls for more reliable hash functions. The viable options could be, for example, SHA-224, SHA-256, SHA-384 and SHA-512 [NIS02].

Some experts propose that a contest for hash function proposals should be started, as was done with AES earlier.

HMAC (a keyed-Hash Message Authentication Code) is an authentication algorithm developed by Bellare, Canetti and Krawczyk in 1996 [BCK96a, BCK96b]. It defines how the secret key is used for authentication, but it does not define what hash function should be used (usually MD5 or SHA is chosen). HMAC (as well as any MAC) can be used to simultaneously verify both the *integrity* of data and the *authenticity* of the message. It works in the following way. A message is processed as *b*-bit blocks and the result of hashing is an *n*-bit number. The length of the secret key should be at least *n* bits. If the size of the secret key is smaller than *b* bits, zeros are added as a prefix until the length *b* is reached. The padding block called *ipad* is constructed by repeating *b/8* times the bit string *00110110*. Also, the padding block called *opod* is constructed by repeating *b/8* times the bit string *01011010*. The block value (*n*-bit code) is computed by $H((K \oplus opad) \circ H((K \oplus ipad) \circ m_i))$, where m_i is the block to be processed, $\circ$ is the concatenation operation, *H* is the hash function, and *K* is the secret key. The security of HMAC obviously depends on the hash function being used, on the size of the secret key, and on the quality of the secret key. [BCK96a, BCK96b, Sta03]

3.4 Digital signatures, MITM attacks and public key certificates

It is not always enough that a sender and a receiver protect themselves against third parties. Sometimes they also need to protect themselves against each other, which is the purpose of the *digital signature*. A good digital signature should have the following properties. A signature depends on the contents of the message and it is unique to the sender, i.e. forging and denial should be prevented. In addition, a signature is relatively easy to produce and recognize. Moreover, falsification of the signature should be unfeasible. A combination of some of the following techniques are used for digital signatures: *symmetric encryption*, *public-key encryption*, *hashing*, *timestamps* (for proving the freshness of the messages and the correct order of events), many different kinds of *serial numbers*, and *reliable third party* offering authentication and signature services. [Pfl03, Sch96, Sta03]

Digital signature can be implemented in the following way, for example (the so-called *direct digital signature*): [Sch96, Sta03]

1. The signature is the whole message or its hash code encrypted with the common secret key.

2. If the confidentiality of the message must also be guaranteed, it can be done by further encrypting the whole message plus the signature with the common secret key or with the public key of the receiver.

The reliability of a direct digital signature depends on how well the common secret key is kept secret. If the sender wants to deny the message later on, she can always say that the secret key has leaked to the wrong hands. Therefore, a digital signature can be made more reliable if a reliable arbiter is available. [Sch96, Sta03]

In August 1991, the NIST proposed a *DSA (Digital Signature Algorithm)* for use in their *DSS (Digital Signature Standard)* [NIS91]. The DSS was revised in 1993 [NIS94], and it was further updated in 1996 and 2000. The original DSS algorithm implements a digital signature by using SHA hashing. The latest version of the DSS [NIS00] also incorporates digital signature algorithms based on RSA and ECC. [NIS00, NIS91, NIS94, Sch96]

The original DSS algorithm works in the following way: [NIS00, NIS91, NIS94, Pfl03, Sch96, Sta03]

- Construct a public key *(p,q,g)* in the following way: let p be a prime, $2^{L-1}<p<2^{L}$, where $512 \leq L \leq 1024$ is a multiple of 64. Let q be a prime that is a factor of $(p-1)$ and $2^{159}<q<2^{160}$. Let $g=h^{(p-1)/q} \bmod p$, where $h^{(p-1)/q} \bmod p>1$.

- Construct a pair *(x,y)* of private and public keys for the user in the following way: let x be a prime ($0<x<q$) and $y=g^{x} \bmod p$.

- Construct a secret session key k, where $0<k<q$ is random or pseudorandom.

- Construct a signature in the following way: let $r=(g^k \bmod p) \bmod q$ and $s=(k^{-1}(h(m)+rx)) \bmod q$. Send $m \circ r \circ s$.

- At the receiving end, verification is performed in the following way: compute $w=s^{-1} \bmod q$, $t=h(m)w \bmod q$, $u=rw \bmod q$, and $v=(g^t y^u \bmod p) \bmod q$. If $r=v$, the verification is completed successfully, i.e. the signature is validated.

Let us assume that Alice and Bob are communicating with each other and they want to secure their communication by using some public-key encryption method. In a *MITM attack*, Mallory (an attacker) intrudes between Alice and Bob. Mallory can eavesdrop on messages, modify messages, delete messages and generate new messages between Alice and Bob in such a way that his presence is unrevealed, i.e. Alice and Bob do not know that the link between them is compromised by Mallory. Mallory is also able to imitate Bob when talking to Alice and vice versa. This simple example of a MITM attack works in the following way: [Pfl03, Sch96]

1. Alice sends her public key to Bob, but Mallory is able to intercept it. Mallory sends Bob his own public key for which he has the matching private key. Now Bob wrongly thinks that he has Alice's public key.

2. Bob sends his public key to Alice, but Mallory is able to intercept it. Mallory sends Alice his own public key for which he has the matching private key. Now Alice wrongly thinks that she has Bob's public key.

3. Alice sends Bob a message encrypted with Mallory's public key, but Mallory is able to intercept it. Mallory decrypts the message with his private key, keeps a copy of the message, re-encrypts the message with Bob's public key, and sends the message to Bob. Now Bob wrongly thinks that the message came directly from Alice.

4. Bob sends Alice a message encrypted with Mallory's public key, but Mallory is able to intercept it. Mallory decrypts the message with his private key, keeps a copy of the message, re-encrypts the message with Alice's public key, and sends the message to Alice. Now Alice wrongly thinks that the message came directly from Bob.

Even if the public keys of Alice and Bob are stored on a database, a MITM attack will work. Mallory can intercept Alice's (or Bob's) database inquiry and substitute his own public key for Bob's (or Alice's) public key. He can also somehow break into the database and substitute his key for both Alice's public key and Bob's public key. A MITM attack works, because Alice and Bob have no way to verify that they are truly using each other's correct public keys. If Mallory does not cause any noticeable delays to the communication, Alice and Bob have no idea that Mallory has intruded between them. [Pfl03, Sch96]

Without any verification of the public keys, MITM attacks are generally possible (in principle) against any message sent using public-key technology. One solution to this problem is to use *public key certificates* (also referred to as *digital identity certificates*) [Koh78], which use digital signatures to bind together public keys with the information of their respective users, i.e. information such as the name of the user, the address of the user, and so on. Each user is associated with a trusted authority, a *CA (Certification Authority)*, and each certificate is created by such a CA. A certificate establishes a verifiable connection between the user and his public keys. The users know their CA's public key and therefore they can verify the signatures of their CA. The certificate is stored in a directory. Only the CA is allowed to write in this directory, but all users of the CA can read the information in the directory. [Buc01, Koh78, Sch96, Sta03]

Defences against MITM attacks use authentication techniques which are based on *public key certificates*, *two-way authentication* (also referred to as *mutual authentication*), *secret keys*, *passwords*, and other methods (such as *voice recognition* and *other biometrics*). [Pfl03, Sch96]

3.5 Network security

As described earlier, network security is needed to protect data during transmission. A *network* in its simplest form consists of two devices connected across some medium (wired or wireless) by hardware and software that enable communication. Wireless radio networks (such as Bluetooth) have to address not only the traditional security problems found in wired networks, but also additional security aspects that wireless communication brings along.

Wireless radio transmissions are difficult to confine into a certain area. In addition, wireless radio signals are quite easy to intercept, disrupt and jam. Therefore, wireless radio communication requires additional countermeasures to those required by wired communication. Moreover, the complexity of wireless ad-hoc networks is much higher than that of traditional wired or centralized wireless networks. [Mor02, Pfl03, Sta03]

The creators of *TCP/IP* had no idea how popular it would be and for what it would be used. Originally, it was used between researchers for sending technical messages. Currently, it is used not only for all kinds of correspondence, but also for bank services, multimedia, running programs over a network, and so on. TCP/IP is used by all kinds of people in all countries. Therefore, network security is a very important issue. In layered architecture (such as TCP/IP), network security can be improved in the *physical layer (PHY)*, the *link layer*, the *network layer*, the *transport layer*, and the *application layer*. The advantage of having security in a lower layer is that all higher layers get security without any significant extra work. However, high quality individualized security can also be achieved by application layer solutions. In a physical layer, network security can be improved by confining wireless signals into a certain area with directional antennas, for example, and also by using some kind of transmit power controlling. In a link layer, network security can be improved by using the *L2TP (Layer 2 Tunneling Protocol),* for example. In a network layer, network security can be improved by using *IPSec (Internet Protocol Security),* for example. In a transport layer, network security can be improved by using *SSL,* for example. In an application layer, network security can be improved by using *SSH (Secure Shell),* for example. [Pfl03, Sta03]

L2TP [IET99] was published in 1999 and the latest version of the protocol was published in 2005 [IET05a]. It is a protocol for tunneling network traffic between two peers over an existing network (usually the Internet). L2TP does not provide confidentiality or strong authentication by itself. Therefore, *IPSec* [IET05b, IET07] is often used to secure L2TP packets by providing confidentiality, authentication and integrity. IPSec is a part of the *TCP/IP version 6* network architecture and its goal is to implement secure network communication in the Internet. IPSec offers security to all applications, which need not even be aware of the existence of IPSec. It also offers secure connection from device to device.

Secure "virtual" subnetworks within a network can be built using IPSec. On the other hand, IPSec does not secure connections from application to application. In addition, it authenticates machines, not users. Moreover, IPSec does not protect against availability attacks (*DoS* attacks). Because IPSec works on the network layer, it protects datagrams (header, contents, or both). Authentication data can be added in optional fields between the header and contents of the datagram to allow individual authentication for each datagram. The contents of a datagram are protected by symmetric encryption. It is also possible to encrypt the whole datagram and add a new header. In this way, a secure "virtual" network within a network can be built. [IET05a, IET05b, IET07, IET99, Pfl03, Sta03]

SSL [Net96] was originally implemented in 1994 by Netscape in order to improve the security of communication between a web browser and a server. Since then, the IETF (Internet Engineering Task Force) has taken SSL as a basis for a network protocol standard called *TLS (Transport Layer Security)* [IET06]. TLS is used by web browsers such as Netscape Navigator, Mozilla and Microsoft Internet Explorer. It is also used in web servers such as Apache, Stronghold and Roxen. SSL is used in web browsers for electronic commerce, for example. An SSL session starts with a negotiation for algorithm support between the client and the server. After the negotiation, a public-key encryption-based key exchange and certificate-based authentication is performed. The supported methods include RSA, Diffie-Hellman, MD5 and SHA. After authentication, symmetric cryptography (and possibly also compression) is used for data transfer. The supported methods include RC4, DES, 3DES, Blowfish and AES. [IET06, Net96, Pfl03]

SSH1 (also referred to as *SSH-1*) was developed in 1995 by Tatu Ylönen. In December 1995, he founded the SSH Communications Security company [SSH08] to market and further develop SSH. SSH works on the application layer and it replaces earlier remote terminal programs by a more secure alternative. SSH1 was replaced in 1996 by *SSH2* (also referred to as *SSH-2*), which is not compatible with SSH1. SSH uses public-key encryption for authentication (the supported methods include RSA and Diffie-Hellman) and symmetric encryption for encrypting data (the supported methods include DES, 3DES, Blowfish, RC4, and AES). [Pfl03, SSH08]

The purpose of a *firewall* is to isolate the local area network from the Internet so that there remains limited access to the Internet, but there is no free access from the Internet to the local area network. Therefore, a firewall has two purposes: [Pfl03, Sta03]

1. To restrict access from the Internet to a PC or to a local area network.

2. To restrict access from a PC or from a local area network to the Internet.

The purpose of an *antivirus software* is to protect a computer system, such as a PC, laptop, PDA (Personal Digital Assistant) or mobile phone, from malicious programs such as viruses and worms. Four different generations of antivirus software exist: [Sta03, Ste93]

1. *Simple scanners*: A simple scanner, a so-called *signature-specific scanner*, requires virus/worm signatures in order to identify them, and it is limited to the detection of previously known viruses/worms. Another form of a simple scanner can maintain a record of program lengths in order to find changes in lengths.

2. *Heuristic scanners*: A heuristic scanner does not need virus/worm signatures to identify them. It uses heuristic rules for searching probable virus/worm infections. Another form of a heuristic scanner uses integrity checking in which a checksum is appended to all programs. If a program is infected without changing the checksum by a virus/worm, an integrity check will reveal the change. An encrypted hash function can be used as a countermeasure for such viruses/worms that are able to change the checksum.

3. *Activity traps*: Third-generation antivirus programs are memory-resident programs for identifying viruses/worms by their actions rather than their structure in infected programs.

4. *Full-featured protection*: Fourth-generation antivirus products consist of many different antivirus techniques, such as components for scanning, components for activity traps, and capability for access controlling, used in conjunction.

Most access violations occur internally by insiders in an organization. Such abuses can be substantially decreased by using an *EFS* (*Encrypting File System)*. Windows 2000, Windows XP Professional and Windows Server 2003 include EFS as a component of the NTFS (New Technology File System) [Bra08]. In Linux environments, for example, EncFS (Encrypted Filesystem) [Gou08] or ReiserFS (Reiser Filesystem) [Shi08] can be used.

On the other hand, not all information is secret. In fact, most of the information may be publicly available, and an enterprise can publish information if it does not endanger its own business. Security should be based on a *data security policy*, which is not only a solution that concerns data and devices, but a process where risks are considered and protection is applied as needed. [And01]

A data security policy can be implemented in the following way, for example: [And01]

1. *Recognize objects that need protection*: Objects are priorized and secured by need, i.e. maximum level of security is not needed to all objects.

2. *Recognize the threats and estimate the probabilities of the threats*: Methods to protect against the threats should also be defined.

3. *Implement security cost effectively*: Because data security costs, it is not reasonable to secure data in vain.

4. *Update the process when objects and/or threats change*: It is important to keep a data security policy up-to-date.

In the next chapter we give an overview of Bluetooth security by first explaining Bluetooth security architecture. We also discuss Bluetooth network vulnerabilities and explain the reasons for such vulnerabilities.

4 OVERVIEW OF BLUETOOTH SECURITY

Because Bluetooth is a wireless communication system, there is always a possibility that the transmission could be deliberately jammed or intercepted, or that false or modified information could be passed to the piconet devices. To provide protection for the piconet, the system can establish security at several protocol levels. Bluetooth has built-in security measures at the link level.

Section 4.1 explains how Bluetooth security is organized. Section 4.2 discusses vulnerabilities of Bluetooth network. Section 4.3 explains the reasons for Bluetooth network vulnerabilities.

4.1 Bluetooth security architecture

This Section explains how security issues have been taken into consideration in current public Bluetooth specifications [Blu99a, Blu99b, Blu01, Blu03, Blu04a, Blu07a]. The basic Bluetooth security configuration is done by the user, who decides how a Bluetooth device will implement its connectability and discoverability options. The different combinations of connectability and discoverability capabilities can be divided into three categories, or *security levels*:

1. *Silent*: The device will never accept any connections. It simply monitors Bluetooth traffic.

2. *Private*: The device cannot be discovered, i.e. it is a so-called *non-discoverable device*. Connections will be accepted only if the *BD_ADDR (Bluetooth Device Address)* of the device is known to the prospective master. A 48-bit BD_ADDR is normally unique and refers globally to only one individual Bluetooth device.

3. *Public*: The device can be both discovered and connected to. It is therefore called a *discoverable device*.

There are also four different *security modes* that a device can implement. In Bluetooth technology, a device can be in only one of the following security modes at a time:

1. *Nonsecure*: The Bluetooth device does not initiate any security measures.
2. *Service-level enforced security mode*: Two Bluetooth devices can establish a nonsecure ACL link. Security procedures, namely authentication, authorization and optional encryption, are initiated when an L2CAP CO or an L2CAP CL channel request is made.
3. *Link-level enforced security mode*: Security procedures are initiated when an ACL link is established.
4. *Service-level enforced security mode*: This mode is similar to mode 2, except that only Bluetooth devices using SSP can use it, i.e. only Bluetooth 2.1+EDR (or later) devices can use this security mode.

Authentication is used for proving the identity of one piconet device to another. The results of authentication are used for determining the client's *authorization level*, which can be implemented in many different ways: for example, access can be granted to all services, only to a subset of services, or to some services while other services require additional authentication. *Encryption* is used for encoding the information being exchanged between Bluetooth devices in a way that eavesdroppers cannot read its contents.

Bluetooth security is based on building a chain of events, none of which provides meaningful information to an eavesdropper, and all events must occur in a specific sequence for security to be set up successfully. Two Bluetooth devices begin pairing with the same *PIN (Personal Identification Number) code* that is used for generating several 128-bit keys, as Figure 4 illustrates. PIN code selection, for example in a personal Bluetooth network environment, i.e. when a Bluetooth network consists of various Bluetooth devices such as a mobile phone, a printer, a DVD (Digital Versatile Disc; also referred to as Digital Video Disc) player, a mouse and a keyboard, can be done by using the same PIN code for all Bluetooth devices, because the user owns and therefore also trusts all Bluetooth devices that are used in his personal

Bluetooth network. However, each master-slave pair can have a different PIN code for providing trusted relationship between the devices. Therefore, PIN code selection, for example in a conference environment where two "friends" meet and want to create a Bluetooth network between their devices, should be done by using a different PIN code for each master-slave pair, because otherwise all other Bluetooth connections that are using the same PIN code may be compromised, i.e. it is possible that the "friend" will use the PIN code for eavesdropping or for attacking purposes. SAFER+ (see Section 3.1) with a 128-bit key is used as an algorithm for Bluetooth authentication and key generation.

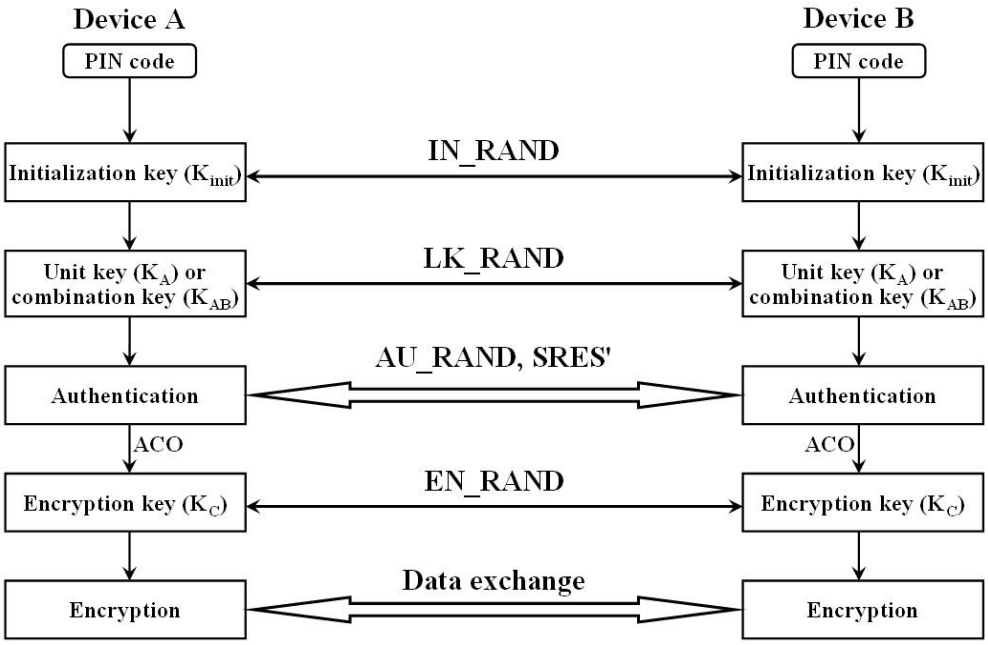

Figure 4. Summary of Bluetooth security operations. [Blu07a]

An *initialization key (K_{init})* is generated when Bluetooth devices meet for the first time and it is used for securing the generation of other more secure 128-bit keys, which are generated during the next phases of the security chain of events. The K_{init} is derived from a 128-bit pseudorandom number IN_RAND, an *L*-byte (*1≤L≤16*) PIN code, and a BD_ADDR. It is worth noting that the IN_RAND is sent via air in unencrypted form.

The output of a certain key generation function can be expressed in terms of the function itself and its inputs. The K_{init} is produced in both devices using the formula $K_{init}=E_{22}(PIN',L',IN\_RAND)$. The PIN code and its length L are modified into two different quantities called PIN' and L' before sending them to the E_{22} function. If the PIN is less than 16 bytes, it is augmented by appending bytes from the device's BD_ADDR until the PIN' either reaches a total length of 16 bytes or the entire BD_ADDR is appended, whichever comes first. If one device has a fixed PIN code, the BD_ADDR of the other device is used. If both devices can support a variable PIN code, the BD_ADDR of the device that received the IN_RAND is used.

The K_{init} is used to encrypt a 128-bit pseudorandom number (RAND or LK_RAND), i.e. $RAND \oplus K_{init}$ or $LK\_RAND \oplus K_{init}$, exchanged in the next phase of the security chain of events when a link key (a unit key or a combination key) is generated.

A *unit key* (K_A) is produced from the information of only one device (device A) using the formula $K_A=E_{21}(BD\_ADDR_A,RAND_A)$. Device A encrypts the K_A with the K_{init}, i.e. $K_A \oplus K_{init}$, and sends it to device B. Device B decrypts the K_A with the K_{init}, i.e. $(K_A \oplus K_{init}) \oplus K_{init}=K_A$, and now both devices have the same K_A as a link key. Only devices that have limited resources, i.e. no memory to store several keys, should use the K_A, because it provides only a low level of security. Therefore, Bluetooth specifications do not recommend using the K_A anymore.

A *combination key* (K_{AB}) is dependent on two devices and therefore it is derived from the information of both devices. The K_{AB} is produced in both devices using the formula $K_{AB}=E_{21}(BD\_ADDR_A,LK\_RAND_A) \oplus E_{21}(BD\_ADDR_B,LK\_RAND_B)$. It is worth noting that generating the K_{AB} is nothing more than a simple bitwise XOR between two unit keys, i.e. $K_{AB}=K_A \oplus K_B$. Each device can produce its own unit key and each device also has the BD_ADDR of the other device. Therefore, two devices have to exchange only their respective pseudorandom numbers in order to produce each other's unit key.

Device A encrypts the LK_RAND_A with the current key K, i.e. $LK\_RAND_A \oplus K$ where K can be the K_{init}, the K_A or the K_{AB} that was created earlier, and sends it to device B. The K is the

K_{init} if the devices create a link key for the first time together. The K is the K_A if the link key is being upgraded to a combination key, and it is the K_{AB} if the link key is being changed. Device *B* decrypts the LK_RAND_A with the K, i.e. $(LK\_RAND_A \oplus K) \oplus K = LK\_RAND_A$, and can now produce the K_A. Correspondingly, device *B* encrypts the LK_RAND_B with the K, i.e. $LK\_RAND_B \oplus K$, and sends it to device *A*. Device *A* decrypts the LK_RAND_B with the K, i.e. $(LK\_RAND_B \oplus K) \oplus K = LK\_RAND_B$, and produces the K_B. Finally, both devices can produce the K_{AB} by XORing the K_A with the K_B, i.e. $K_{AB} = K_A \oplus K_B$.

The next phase of the security chain of events is the challenge-response authentication in which a claimant's knowledge of a secret link key is checked, as Figure 5 illustrates. During each authentication, a new 128-bit pseudorandom number AU_RAND is exchanged via air in unencrypted form. Other inputs to the authentication function E_1 are the BD_ADDR of the claimant and the current link key (K_A or K_{AB}).

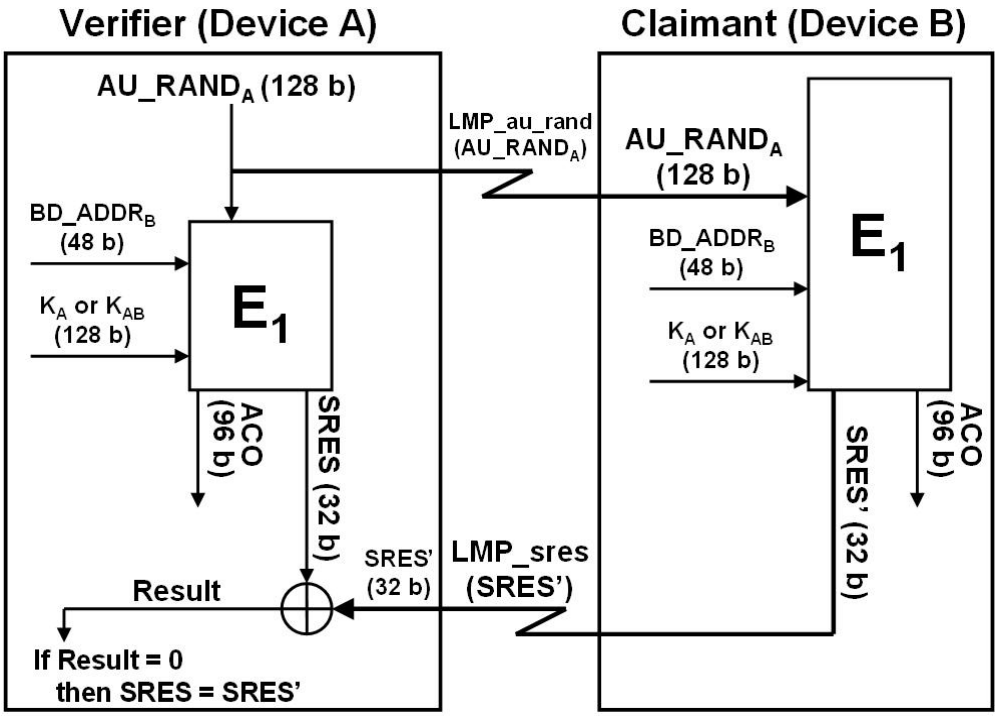

Figure 5. Bluetooth challenge-response authentication. [Blu07a]

A 32-bit result (SRES, Signed Response) and a 96-bit result (ACO, Authenticated Ciphering Offset) are produced in both devices by the $E_1(AU\_RAND_A,BD\_ADDR_B,Link\ key)$ function, where the *Link key* is the K_A or the K_{AB}. The claimant sends the SRES', i.e. the SRES value produced by the claimant, via air in unencrypted form to the verifier. The verifier compares the generated SRES value with the received SRES value, and if these values match, the authentication is completed successfully. The ACO is used in the next phase of the security chain of events when an encryption key is generated.

It is worth noting that the SRES and the SRES' are 32-bit values, not 128-bit values. The 32-bit SRES provides reasonable protection against an attacker who is trying to guess the value, while it also reduces the chance that the PIN code will be compromised by an attacker who has somehow determined the correct SRES value.

Figure 6 illustrates Bluetooth data encryption between two Bluetooth devices. The ACO, the current link key (K_A or K_{AB}) and a 128-bit pseudorandom number EN_RAND are inputs to the encryption key generation function E_3 that is used for generating an *encryption key* (K_C). The master (device *A*) generates the EN_RAND and sends it to the slave (device *B*) via air in unencrypted form. The K_C is produced in both devices using the formula $K_C=E_3(EN\_RAND_A,ACO,Link\ key)$, where the *Link key* is the K_A or the K_{AB}.

The keystream generator function E_0 (see Figure 6) makes symmetric encryption possible by generating the same cipher bit stream, or a *keystream* (also referred to as a *running key*), in both devices. The inputs to the E_0 function are the K_C, the BD_ADDR of the master (BD_ADDR_A), and the 26 bits of the master's real-time clock (CLK_{26-1}). The keystream is generated by the $E_0(K_C,CLK_{26-1},BD\_ADDR_A)$ function that is reinitialized for every new sent or received Baseband packet, i.e. the CLK_{26-1} is updated for every new Baseband packet. It means that inputs to the E_0 function are never identical longer than a lifetime of one Baseband packet, and therefore a new keystream is generated for every new Baseband packet.

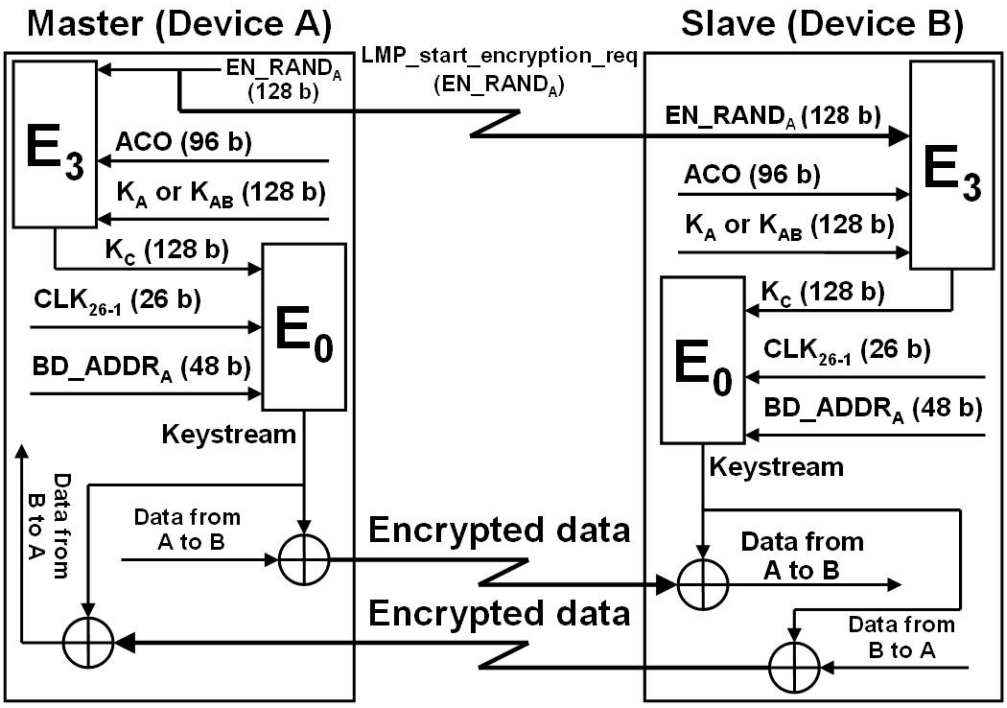

Figure 6. Bluetooth data encryption. [Blu07a]

The sender encrypts the plaintext by XORing it with the keystream, i.e. *Plaintext*⊕*Keystream=Ciphertext*, and sends the produced ciphertext to the receiver. The receiver decrypts the ciphertext by XORing it with the same keystream, i.e. *Ciphertext*⊕*Keystream=(Plaintext*⊕*Keystream)*⊕*Keystream=Plaintext*. It is worth noting that only the payload of the Bluetooth Baseband packet is encrypted (not an access code or a header), and therefore an attacker cannot use the regularly repeating information (that is easy to guess by the attacker) of the access code and the header in order to facilitate a cryptanalysis of the cipher.

In Bluetooth versions up to 2.0+EDR, pairing is based exclusively on the fact that both devices share the same PIN code or passkey. The PIN is the only source of entropy for the shared secret. As the PINs often contain only four decimal digits, the strength of the resulting keys is not enough for protection against passive eavesdropping on communication. Even

with longer 16-character alphanumeric PINs, full protection against active eavesdropping cannot be achieved: it has been shown that MITM attacks on Bluetooth communications (see Subsection 4.2.2 and Section 6.11) can be performed [JaW01, Küg03, LCA04].

As already mentioned in Section 2.1, Bluetooth version 2.1+EDR adds a new specification for the pairing procedure: SSP. Its main goal is to improve the security of pairing by providing protection against passive eavesdropping and MITM attacks.

Instead of using (often short) passkeys as the only source of entropy for building the link keys, SSP employs ECDH public-key cryptography (see Section 3.2). To construct the link key, devices use public-private key pairs, a number of nonces, and the Bluetooth addresses of the devices. Passive eavesdropping is effectively thwarted by the SSP, as running an exhaustive search on a private key with approximately 95 bits of entropy is currently considered to be infeasible in a short time.

In order to provide protection against MITM attacks, SSP either uses NFC as an OOB channel (see Section 2.1), or asks for the user's help: for example, when both devices have displays and keyboards, the user is asked to compare two six-digit numbers. Such a comparison can also be thought of as an OOB channel which is not controlled by the MITM. If the values used in the pairing process have been tampered with by the MITM, the six-digit integrity checksums will differ with the probability of 0.999999.

SSP uses four *association models*. In addition to the two association models mentioned previously (*OOB* and *Numeric Comparison*), models named *Passkey Entry* and *Just Works* are defined. The Passkey Entry model is used in the cases when one device has input capability, but no screen that can display six digits. A six-digit checksum is shown to the user on the device that has output capability, and the user is asked to enter it on the device with input capability. The Passkey Entry model is also used if both devices have input, but no output capabilities. In this case the user chooses a 6-digit checksum and enters it in both devices. Finally, if at least one of the devices has neither input nor output capability, and an OOB cannot be used, the Just Works association model is used. In this model the user is not

asked to perform any operations on numbers; instead, the device may simply ask the user to accept the connection.

The choice of the association model depending on the device capabilities is shown in Table 3. DisplayYesNo indicates that the device has a display and at least two buttons that are mapped to "yes" and "no": using the buttons, the user can either accept the connection or decline it. Other notation in the table is self-explanatory.

Device 1:	Device 2:	Association model:
DisplayYesNo	DisplayYesNo DisplayOnly KeyboardOnly NoInputNoOutput	Numeric Comparison* Numeric Comparison Passkey Entry* Just Works
DisplayOnly	DisplayOnly KeyboardOnly NoInputNoOutput	Numeric Comparison Passkey Entry* Just Works
KeyboardOnly	KeyboardOnly NoInputNoOutput	Passkey Entry* Just Works
NoInputNoOutput	NoInputNoOutput	Just Works

* The resulting link key is considered *authenticated*.

Table 3. Bluetooth device capabilities and SSP association models. [Blu07a]

SSP consists of six phases:

1. *Capabilities exchange*: Devices that have never met before or want to perform re-pairing for some reason, first exchange their IO (Input/Output) capabilities (see Table 3) to determine the proper association model to be used.

2. *Public key exchange*: The devices generate their public-private key pairs and send the public keys to each other. They also compute the Diffie-Hellman key.

3. *Authentication stage 1*: The protocol that is run at this stage depends on the association model. One of the goals of this stage is to ensure that there is no MITM in

the communication between the devices. This is achieved by using a series of nonces, commitments to the nonces, and a final check of integrity checksums performed either through the OOB channel or with the help of the user.

4. *Authentication stage 2*: The devices complete the exchange of values (public keys and nonces) and verify their integrity.

5. *Link key calculation*: The parties compute the link key using their Bluetooth addresses, the previously exchanged values and the Diffie-Hellman key constructed in phase 2.

6. *LMP authentication and encryption*: Encryption keys are generated in this phase, which is the same as the final steps of pairing in Bluetooth versions up to 2.0+EDR (see Figures 4 and 6).

The contents of messages sent during the SSP are outlined in Figure 7, and the notations used are explained in Table 4.

Notation:	**Definition:**
PKx	Public key of device X
SKx	Private key of device X
DHKey	Diffie-Hellman key generated after key exchange
Nx	Nonce generated by device X
rx	Random number generated by device X; equals 0 in the Numeric Comparison association model
Cx	Commitment value from device X
f1	One-way function used to compute commitment values
f2	One-way function used to compute the link key
f3	One-way function used to compute check values
g	One-way function used to compute numeric check values
IOcapX	Input/Output capabilities of device X

Table 4. SSP protocol notations. [Blu07a]

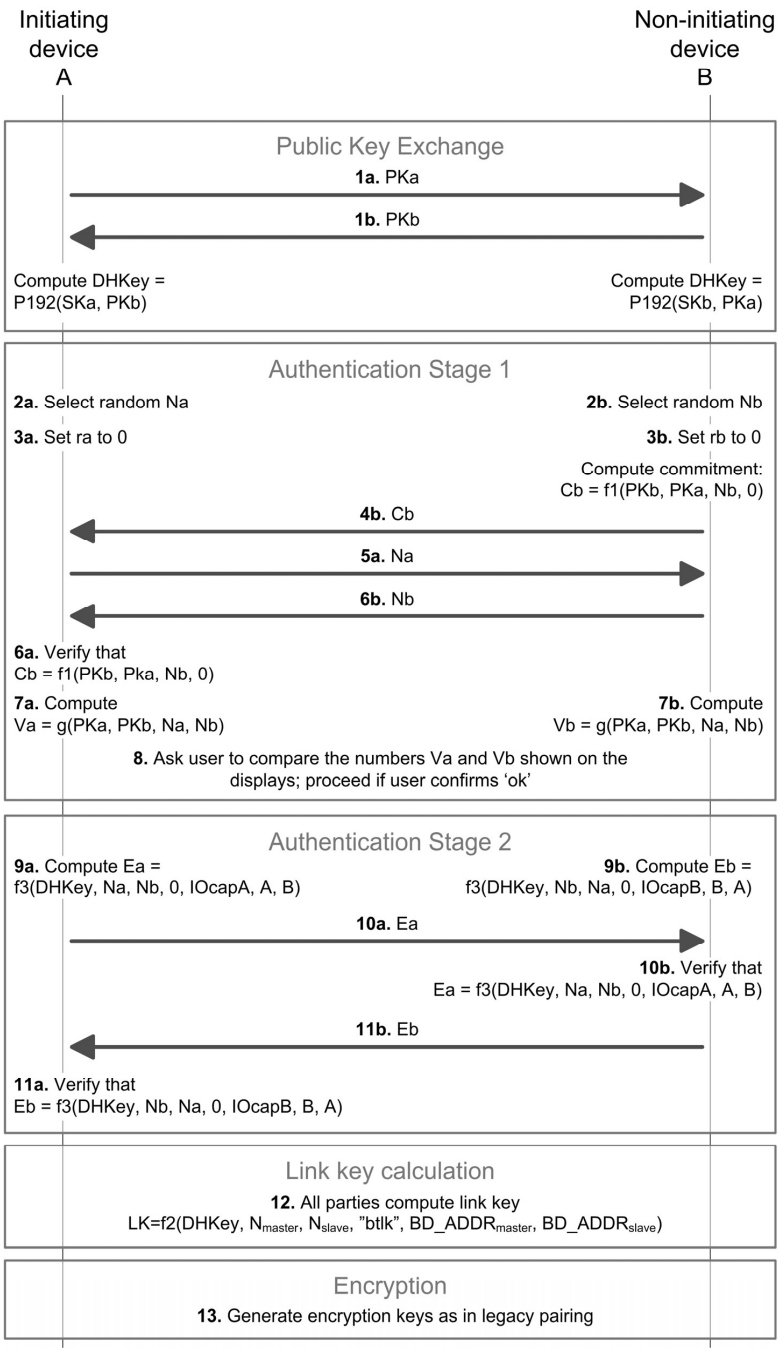

Figure 7. SSP with the Numeric Comparison association model. [Blu07a]

As a result of the work of the Bluetooth SIG, SSP has gone through a series of reviews by experts; the released version does a good job in improving the security of Bluetooth pairing. However, it has been shown that MITM attacks against Bluetooth 2.1+EDR devices (see Section 6.11) are also possible [HaH08, HyH07, SVA07]. More information about Bluetooth security can be found in [ArK03, Blu07a, FlL01, Flu02, GPS04, LMV05, LuV04, Mor02].

4.2 Bluetooth network vulnerabilities

Since there are now (and will be in the near future) billions of Bluetooth devices in use without the SSP's improved security features (see Sections 2.1 and 4.1), malicious security violations are not expected to decrease in the near future. On the contrary, these old Bluetooth devices will be sold for many years to come, thus making security concerns even more alarming. Moreover, attacks against SPP are also possible (see Section 6.11). Therefore, Bluetooth security architecture needs to be further upgraded to prevent these new threats. The next major security improvements are roadmapped to the upcoming Bluetooth specification, Seattle (see Section 2.1), which is expected to be released by the Bluetooth SIG in 2009.

Security threats in distributed networks (such as Bluetooth) can be divided into three categories: disclosure threat, integrity threat and DoS threat. *Disclosure threat* means that information can leak from the target system to an eavesdropper that is not authorized to access the information. *Integrity threat* concerns the deliberate alteration of information in an attempt to mislead the recipient. *DoS threat* involves blocking access to a service, making it either unavailable or severely limiting its availability to an authorized user. [Mor02]

Bluetooth security is currently a very active research area, because Bluetooth devices are widely used all over the world. In addition to our Bluetooth research, there are several research papers, research reports and homepages about the security vulnerabilities of Bluetooth, i.e. descriptions of many different kinds of disclosure, integrity and DoS attacks. Disclosure and integrity attacks typically compromise some sensitive information and therefore can be very dangerous, while DoS attacks typically only annoy Bluetooth network users and are considered to be less dangerous.

Powerful directional antennas can be used to considerably increase the scanning, eavesdropping and attacking range of almost any kind of Bluetooth attack. One very good example of a long-distance attacking tool is the BlueSniper Rifle [Che05a, Che05b]. It is a rifle stock with a powerful directional antenna attached to a small Bluetooth-compatible computer, so there is no need to carry heavy laptops in a backpack just to gather data. The scanning, eavesdropping and attacking can be done over a mile away from the target devices. Moreover, everyone with some basic skills and a few hundred dollars can build her BlueSniper Rifle. Therefore, the possibility that an attacker is using range enhancement for improving the performance of DID (Disclosure, Integrity and DoS) attacks should be taken seriously.

Nowadays it is also possible to transform a standard $30 Bluetooth dongle into a full-blown Bluetooth sniffer [Blu07c, Mos07]. We have also verified this fact in our laboratory (see Chapter 5) with many different CSR-based (Cambridge Silicon Radio) Bluetooth USB (Universal Serial Bus) dongles supporting Bluetooth versions up to 2.0+EDR. In addition, tools for reverse engineering the firmware of CSR-based Bluetooth dongles are available [Dar07]. The tools include a disassembler for the official firmware, and an assembler that can be used for writing a custom firmware. With these tools anyone can now write a custom firmware for CSR-based Bluetooth dongles to include raw access for Bluetooth sniffing. The tools also include the source code for sniffing Bluetooth under Linux. Moreover, it is expected that in the near future the techniques for finding hidden (non-discoverable) Bluetooth devices in an average of one minute [SpB07a, SpB07b] (see Subsection 4.2.4) will be ported onto a standard CSR dongle via a custom firmware. This will open new doors for practical Bluetooth security research and it will also provide a cheap basic weapon to all attackers for Bluetooth sniffing. Therefore, it is expected that Bluetooth sniffing will soon become a very popular sport among attackers and hackers, thus making Bluetooth security concerns even more alarming.

Subsections 4.2.1-4.2.3 explain some typical disclosure, integrity and DoS threats. Some typical threats which cannot be classified as only one single threat (so-called *multithreats*) are explained in Subsection 4.2.4.

4.2.1 Disclosure threats

BlueSnarfing attack [Her04, LaL04] (also referred to as *BlueStumbling attack*) means that an attacker connects to the target device without alerting its owner and steals some sensitive information such as entire phonebook, calendar notes and text messages. At least three BlueSnarfing applications exist: Adam Laurie's BlueSnarf [LaL04], Ollie Whitehouse's RedSnarf 1.0 [Whi04] and Bluediving Project's Bluediving 0.8 [Blu07d]. The source codes and binaries of BlueSnarf and RedSnarf 1.0 have not been released, while the source codes and binaries of Bluediving 0.8 have been made public. Bluediving 0.8 runs on Linux and is based on the BlueZ [Blu08b] protocol stack. BlueZ is the official Bluetooth protocol stack for Linux environments and is included in the Linux 2.4 (or later) kernel series.

As described in Section 2.4, OBEX can be used to exchange business cards between Bluetooth devices using the OBEX protocol's *object push* feature, i.e. pushing (sending) business cards (objects) to Bluetooth devices using the OBEX protocol's *Put operation*. In most cases, this service does not require authentication. BlueSnarfing is based on the exploitation of the OBEX protocol's *object pull* feature instead of the *object push* feature. This feature is used for pulling (receiving) objects from Bluetooth devices using the OBEX protocol's *Get operation*. BlueSnarfing attack conducts an *OBEX Get request* for known filenames such as telecom/pb.vcf (the phone book) or telecom/cal.vcs (the calendar file). A detailed description of the OBEX protocol and its operations can be found in [Inf03], while a detailed description on how to implement BlueSnarfing attack in practice can be found in [Blu07d].

The success of BlueSnarfing attack depends very much on the vendor's implementation of the Bluetooth protocol stack for the target device. Therefore, the attack works only if the protocol stack of the target device is poorly implemented, i.e. there are serious flaws in the authentication and data transfer mechanisms of some Bluetooth devices. A list of the devices known to be vulnerable to BlueSnarfing attack without firmware/software update can be found in [LaL04]. Moreover, BlueSnarfing is normally only possible and dangerous if the target device's security level is set as public, i.e. the target device is discoverable, but there are also ways to find non-discoverable devices, for example by using a Bluetooth protocol

analyzer (see Chapter 5 and Section 6.1), brute-force scanning (see Subsection 4.2.4 and Section 6.4), or the techniques introduced in [SpB07a] (see Subsection 4.2.4).

An *Off-Line PIN Recovery attack* [JaW01, Whi04] is based on intercepting the IN_RAND value, LK_RAND values, AU_RAND value and SRES value, and after that trying to calculate the correct SRES value by guessing different PIN values until the calculated SRES equals the intercepted SRES (see Figures 4 and 5 in Section 4.1). It is worth noting that SRES is only 32 bits long. Therefore, a SRES match does not necessarily guarantee that an attacker has discovered the correct PIN code, but the chances are quite high especially if the PIN code is short [GPS04, Mor02].

An Off-Line PIN Recovery attack is dangerous only if the PIN code is short and has no case-sensitive alphanumerical characters (and perhaps some other characters as well). Moreover, an attacker must intercept the traffic of the initial pairing process between two Bluetooth devices when they meet for the first time (see Section 4.1). This can be arranged in many different ways, for example by sending a Bluetooth device anonymously to the target person as a prize in some competition. Another possible way to witness the initial pairing process is to disrupt the connection establishment process between two devices, for example by disrupting the PHY in such a way that a user thinks something is wrong and deletes previously stored link keys. After that the user initiates a new pairing process and the attacker can intercept all the required inputs for an Off-Line PIN Recovery attack.

An *Enhanced implementation of Off-Line PIN Recovery attack* [ShW05] is an average of 30% faster than the original Off-Line PIN Recovery attack described in [JaW01, Whi04]. It is based on the optimization of SAFER+ (see Section 3.1) using the algebraic manipulation of the SAFER+ round. Three methods which can force two target devices to repeat the initial pairing process are also proposed in [ShW05]:

- The *first method* (and two other methods as well) is based on the fact that Bluetooth specifications allow Bluetooth devices to forget a link key: when the master sends AU_RAND to the slave during the authentication, the slave sends an *"LMP not accepted"* message in return for letting the master know that it has forgotten the link

key, i.e. the slave does not send the SRES value as in normal authentication. Therefore, the master is convinced that the slave has lost the link key and the pairing process is restarted.

- The *second method* works in the following way: at the beginning of the authentication process, the master is supposed to send AU_RAND to the slave. If an attacker sends IN_RAND to the slave before the master sends AU_RAND, the slave device is convinced that the master has lost the link key and pairing is restarted.

- The *third method* works in the following way: when the master sends AU_RAND to the slave during the authentication, the attacker sends a random SRES message to the master, causing the authentication process to restart, and the same kind of repeated attempts will be made. After a certain number of failed authentication attempts, the master is expected to declare that the authentication process has failed and the pairing process is restarted. The required number of failed authentication attempts is implementation dependent.

These three methods require that an attacker has a custom Bluetooth device, such as a protocol analyzer, for cloning the BD_ADDR values of the target devices, i.e. the methods are based on the impersonation of target devices. Moreover, the Bluetooth user is required to enter a PIN code again during the new pairing process, and therefore a suspicious user may realize that her device is under attack. [ShW05]

An *Off-Line Encryption Key Recovery attack* [Whi04] extends an Off-Line PIN Recovery attack, and is based on intercepting the IN_RAND value, LK_RAND values, AU_RAND value, SRES value and EN_RAND value, i.e. it requires that an attacker intercepts all the values required for the Off-Line PIN Recovery attack and also the EN_RAND value. When the PIN code is discovered via an Off-Line PIN Recovery attack, the attacker can produce the required ACO by the $E_1(AU\_RAND_A, BD\_ADDR_B, Link\ key)$ function, where the *Link key* is the K_A or the K_{AB}. After that the attacker can recover the encryption key (see Figure 6 in Section 4.1) using the formula $K_C=E_3(EN\_RAND_A, ACO, Link\ key)$, where the *Link key* is the K_A or the K_{AB}. Therefore, an Off-Line Encryption Key Recovery attack is dangerous only

when an Off-Line PIN Recovery attack or its enhanced implementation has been completed successfully.

Every Bluetooth device has some characteristics which are unique (BD_ADDR), manufacturer specific (the first three bytes of BD_ADDR), and model specific (SDP records). Moreover, every Bluetooth device that offers services to other Bluetooth devices will announce its SDP records via the SDP protocol (see Section 2.4). Therefore, remote Bluetooth devices can query other Bluetooth devices based on the offered capabilities. SDP records, which consist of information on how to access the particular service, are returned to the querying device. Certain values of SDP records can be used to calculate a fingerprint value that is used for determining the device model and the firmware version of the target device.

A *BluePrinting attack* [HeM04] is used to determine the manufacturer, device model and firmware version of the target device. An attacker can use Blueprinting to generate statistics about Bluetooth device manufacturers and models, and to find out whether there are devices in the range of vulnerability that have issues with Bluetooth security, for example. BluePrint 0.1 [Tri06a] is a tool for performing BluePrinting attack. It runs on Linux and is based on the BlueZ protocol stack. BluePrinting attacks work only when the BD_ADDR of the target device is known.

4.2.2 Integrity threats

Reflection attacks [LCA04] (also referred to as *Relay attacks*) are based on the impersonation of target devices. An attacker does not have to know any secret information, because she only relays (reflects) the received information from one target device to another during the authentication (see Figure 5 in Section 4.1), i.e. a Reflection attack in Bluetooth can be seen as a type of a MITM attack (see Section 3.4) against authentication, but not against encryption. The only information needed is the BD_ADDRs of the target devices.

Reflection attacks are possible only when the target devices do not hear each other, i.e. the communication between the target devices is not possible because they are out of each other's range, and the attacker has Bluetooth devices with adjustable BD_ADDRs (for example, protocol analyzers). Moreover, the attacker must be capable of relaying the received

information between the devices that perform the Reflection attack, because the target devices can be very far away from each other. These kinds of special conditions are not rare in Bluetooth networks, because Bluetooth is a short range communication technology and it is very likely that devices move occasionally out of each other's range.

There are two different kinds of Reflection attacks: *One-Sided Reflection attack* and *Two-Sided Reflection attack*. Only one target device is impersonated in a One-Sided Reflection attack, while a Two-Sided Reflection attack requires that both target devices are impersonated. These two attacks and possible countermeasures are described in [LCA04]. Bluetooth specifications up to 2.0+EDR do not provide any proper countermeasures against Reflection attacks, but the latest 2.1+EDR version of Bluetooth (see Section 4.1) provides protection against these kinds of active eavesdropping attacks (MITM attacks). However, it has been shown that MITM attacks against Bluetooth 2.1+EDR devices are also possible (see Section 6.11). An attacker can successfully perform authentication using a Reflection attack, but she cannot continue the attack if the target devices want to have encrypted communication, i.e. the attacker does not know the secret PIN code, link key or encryption key. Therefore, Reflection attacks are dangerous only when encryption is not used.

A very dangerous form of integrity threat takes place when an attacker uses a stronger RF signal in order to displace the active piconet device. The main principle for successfully completing an *Exploitation of a stronger RF signal attack* [Mor02] is to send the target device's receiver an RF signal that is at least 11 dB stronger than the signal that the legitimate piconet device is sending. The attack can be performed in the following way, for example. Let us assume that the attacker *A'* wants to have a sensitive file *F* that is located on a server *S*. First, *A'* eavesdrops communication until she can identify the BD_ADDR of *S* on which *F* is located. *A'* also identifies the BD_ADDR of device *A,* which is authorized to access *F*. Secondly, *A'* waits until *A* connects to *S* and is properly authenticated. Thirdly, *A'* captures the channel of *A* by impersonating *A* and transmitting signals that are at least 11 dB stronger at the receiver of S than those *A* was sending. Fourthly, *A'* continues to pose as *A*, and finally *A'* requests the desired *F* from *S*. Exploitation of a stronger RF signal attack is dangerous only when the BD_ADDRs of the target devices are known. [Mor02]

A *Backdoor attack* [LaL04] means that an attacker establishes a trusted relationship with the target device through authentication, and ensures that this trusted relationship no longer appears to be in the target device's register of paired (authenticated) devices. When the backdoor is installed in the target device via a Backdoor attack, the attacker can continue the attack in many different ways: for example, trying to exploit the resources of the target device via a trusted relationship, trying to perform a BlueSnarfing attack (see Subsection 4.2.1), or trying to slip a virus or worm to the target device (see Subsection 4.2.4). A Backdoor attack works only if the BD_ADDR of the target device is known. Moreover, the target device has to be vulnerable to a Backdoor attack. A list of the devices known to be vulnerable to Backdoor attacks without a firmware/software update can be found in [LaL04].

4.2.3 Denial-of-Service threats

DoS threats can be roughly divided into two parts: *attacks against the PHY* and *attacks against protocols above the PHY*.

At the physical layer (PHY), an attacker can jam the piconet entirely or capture the channel from the legitimate piconet device. Let us assume that a jammer wants to disrupt (jam) the PHY by hopping along with the piconet devices and sending random data in every timeslot. How far away can the jammer be? We can calculate the distance by making a number of realistic assumptions such as:

1. The target piconet devices are using omnidirectional antennas with 0 dBi gain.

2. The C/I (Carrier-to-Interference ratio; also referred to as CIR) of the Bluetooth radio must be at least 0 dB for effective jamming, i.e. jamming power is at least equal to the desired signal's power at the target receiver.

3. All target piconet receivers have the desired signal power level of either -60 dBm or -40 dBm.

4. The jammer's transmit power is either 20 dBm (class 1 device) or 30 dBm (Bluetooth transmitter with a power amplifier).

5. The level of obstacles is either light (*n=2.5*) or moderate (*n=3.0*).

6. The jammer uses a directional antenna with a gain of 20 dBi.

Table 5 presents the *range of susceptibility* for the jammer on a target piconet with these prerequisites calculated using the formula $d_j = 10^{(P_t - P_r - 40)/(10n)}$, where d_j is the range of susceptibility for the jammer, P_t is the jammer's transmit power, P_r is the minimum power needed at a target piconet receiver for jamming to occur, and n is the PL exponent. [Mor02]

Level of obstacles:	n:	Jammer's TX power (dBm):	Minimum jamming signal power at target receiver (dBm):	Range of susceptibility for a jammer (m):
Light	2.5	20	-40	6
Light	2.5	30	-40	16
Light	2.5	20	-60	40
Light	2.5	30	-60	100
Moderate	3.0	20	-40	5
Moderate	3.0	30	-40	10
Moderate	3.0	20	-60	22
Moderate	3.0	30	-60	46

Table 5. The range of susceptibility for a jammer on a target piconet. [Mor02]

As Table 5 illustrates, a jammer can perform *Disruption of the PHY attack* [Mor02] relatively far away from the communicating devices (up to 100 meters) by using a Bluetooth transmitter with a power amplifier and a directional antenna with a gain of 20 dBi. This type of attack can be very dangerous if the attacker is using a stronger RF signal to displace the existing legitimate piconet device (see Subsection 4.2.2) and then trying to steal some sensitive information from the target device.

Attacks on higher levels of the Bluetooth protocol stack try to exploit some of the characteristics of higher level protocols in an attempt to occupy the attention of one or more devices of the piconet in such a way that they are unable to serve other legitimate devices within a reasonable time. These kinds of attacks are not normally very dangerous, because an

attacker does not steal any information from the target device. For example, BD_ADDR Duplication attacks, SCO/eSCO attacks, Big NAK (Negative Acknowledgement) attacks, L2CAP Guaranteed Service attacks and Battery Exhaustion attacks can be classified as attacks against protocols above the PHY.

A *BD_ADDR Duplication attack* [Mor02] (see Subsection 6.8) is based on the idea that an attacker places a bug in the range of susceptibility. The bug duplicates the BD_ADDR of the target device. When any Bluetooth device tries to make a connection with the target device, either the target device or both devices, i.e. the target device and the bug, will respond and jam each other. In this way, the attacker has denied access from the legitimate device. The most effective way to perform this attack is to duplicate the BD_ADDR of the piconet master, because all information within the piconet goes through the master device.

A *SCO/eSCO attack* [Mor02] (see Subsection 6.9) is based on the fact that a realtime two-way voice reserves a great deal of a Bluetooth piconet's attention so that the legitimate piconet devices are not getting the service within a reasonable time. The most effective way to perform this type of attack is to establish a SCO or an eSCO link with the piconet master.

A *Big NAK attack* [Mor02] (see Subsection 6.10) is based on the idea of putting the target device on an endless retransmission loop so that the legitimate piconet devices have considerably slowed throughput. In this attack, an attacker requests any information from the target device and every time the requested information is received, the attacker sends NAK, i.e. the transmission has failed.

A *L2CAP Guaranteed Service attack* [Mor02] is based on the idea that an attacker requests the highest possible data rate or the smallest possible latency from the target device so that all other connections are refused, and all throughput is reserved for the attacker.

A *Battery Exhaustion attack* [Mor02] is based on the idea of occupying the target device in such a way that it also consumes rather quickly the battery of the target device.

4.2.4 Multithreats

There are also many attacks which cannot be classified as only one single threat. For example, BlueBugging attacks, Blooovering attacks, On-Line PIN Cracking attacks, and Brute-Force BD_ADDR Scanning attacks can be classified as *disclosure and integrity threats*.

A *BlueBugging attack* [Tri06b] (see Section 6.2) means that an attacker connects to the target device (typically a Bluetooth mobile phone) without alerting its owner, steals some sensitive information, such as an entire phonebook, calendar notes and text messages, and has full access to the GSM (Global System for Mobile communications) AT command set, i.e. a BlueBugging attack is based on the exploitation of AT commands. It means that the attacker can, in addition to stealing information, send text messages to premium numbers, initiate phone calls to premium numbers, write phonebook entries, connect to the Internet, set call forwards, try to slip a Bluetooth virus or worm to the target device, and many other things.

A BlueBugging attack is even more dangerous than a BlueSnarfing attack, because the attacker can do almost anything with the vulnerable target device. A BlueBugging attack is possible and dangerous if the BD_ADDR of the target device is known to the attacker. Moreover, the Bluetooth protocol stack of the target device has to be poorly implemented, i.e. there are serious flaws in the authentication and data transfer mechanisms of some Bluetooth devices. A list of the devices known to be vulnerable to a BlueBugging attack without a firmware/software update can be found in [LaL04]. Several public BlueBugging tools exist: for example, btxml [Obe04] (see Section 6.2) and Blooover [Tri06c, Tri06d].

Blooover [Tri06c] and its successor Blooover II [Tri06d] are derived from Bluetooth Hoover, because they run on handheld devices, such as PDAs or mobile phones, and are capable of stealing sensitive information by using a BlueBugging attack. A *Blooovering attack* [Tri06c, Tri06d] can be done secretly, by using only a Bluetooth mobile phone with Blooover or Blooover II installed, for example. Blooover and Blooover II run on almost every J2ME (Java 2 Micro Edition) compatible handheld device. They are intended to serve as auditing tools which can be used for checking whether Bluetooth devices are vulnerable or not, but they can be used for attacking against Bluetooth devices as well. The source codes of Blooover and

Blooover II have not been published (only binaries). A Blooovering attack is dangerous only if the target device is vulnerable to BlueBugging. Moreover, an attacker has to know the BD_ADDR of the target device.

An *On-Line PIN Cracking attack* [Whi04] (see Section 6.3) means that an attacker is trying to connect with the target device by guessing different PIN values. It is based on the idea of changing the BD_ADDR of the attacking device every time a PIN guess fails, i.e. the attacker bypasses the ever increasing delay between retries. An On-Line PIN Cracking attack works only when the target device has a fixed or short adjustable PIN code and its BD_ADDR is known to the attacker. Bluetooth specifications up to 2.0+EDR do not provide any proper countermeasures for On-Line PIN Cracking attacks, while Bluetooth 2.1+EDR provides SSP (see Section 4.1) to protect against such attacks.

The 48-bit BD_ADDR is divided into three parts: 16-bit NAP (Nonsignificant Address Part), 8-bit UAP (Upper Address Part) and 24-bit LAP (Lower Address Part). The first three bytes of BD_ADDR (NAP and UAP) refer to the manufacturer of the Bluetooth chip and represent *company_id*. The last three bytes of BD_ADDR (LAP), so-called *company_assigned*, are used more or less randomly in different Bluetooth device models. Company_id values are public information and listed in IEEE's OUI database [IEE08].

A *Brute-Force BD_ADDR Scanning attack* [Whi04] (see Section 6.4) uses a brute-force method only on the last three bytes of a BD_ADDR, because the first three bytes are publicly known and can be set as fixed. A Brute-Force BD_ADDR Scanning attack is perhaps the most feasible when target devices are Bluetooth mobile phones, because millions of vulnerable Bluetooth mobile phones are used every day all over the world.

Besides a Brute-Force BD_ADDR Scanning attack, *techniques for finding hidden Bluetooth devices in an average of one minute* have been proposed [SpB07a] and even implemented [SpB07b] recently. Spill et al. also implemented an *open-source Bluetooth sniffer* [SpB07b] that operates on a single channel. The USRP (Universal Software Radio Peripheral) [Ett07] was used as a radio device to eavesdrop on Bluetooth packets. It is the hardware device associated with the GNU Radio Project [GNU08], which develops an open-source framework

for implementing software radio systems, i.e. systems in which radio devices are implemented in software.

Due to the buffering and asynchronous nature of the GNU Radio framework and the hardware restrictions of the USRP, no working prototype of the Bluetooth sniffer that supports frequency hopping has been implemented yet. However, the current version of the Bluetooth sniffer is still capable of finding hidden (non-discoverable) Bluetooth devices in the range of susceptibility in an average of one minute by using the following techniques: [SpB07a]

- First ($phase_1$), the LAP can be determined in a straightforward manner since it is present in every Baseband packet in the form of a constant 72-bit pattern called the access code: it contains the 24-bit LAP along with its 34-bit checksum and 14 bits of synchronization and error detection data. Therefore, the LAP can be simply read from an intercepted Baseband packet and validated by its checksum. In order to eavesdrop a Baseband packet, it is sufficient to stay tuned to a single channel and wait for a Baseband packet to fly by. Since the channel hopping rate is very high (1600 hops/second), waiting for one second is more than enough in order to intercept a Baseband packet.

- Secondly ($phase_2$), each Baseband packet has a 10-bit header with an 8-bit HEC (Header Error Check) field that is calculated from the UAP. Spill et al. noticed that it was possible to reverse the HEC in order to reveal the UAP in almost realtime.

- Finally ($phase_3$), since the first byte of the NAP is almost always zero in practice, the remaining byte can be brute-forced by sending at most 256 pings to all the possible remaining BD_ADDR combinations.

Pinging a Bluetooth device takes approximately one second. The devices also need to find one another and identify themselves. This takes up to a second, because both devices have unique hopping patterns and these patterns need to coincide on a frequency before communication can take place. Therefore, it takes only an *average of 4.3 minutes* ($phase_1+phase_2+phase_3 \approx 1s+0s+2s \times 256/2 \approx 4.3$ minutes) to find a hidden Bluetooth device in the range of vulnerability. Moreover, IEEE's OUI database [IEE08] can be used to make

educated guesses regarding the last byte of NAP rather than blindly brute-forcing it. Typically, filtering the OUI list for vendor prefixes yields only a few dozen brute-force candidates, thus further reducing the time requirement.

Spill et al. performed a practical experiment [SpB07a, SpB07b] in which they first revealed the first four bytes (LAP and UAP) of the hidden BD_ADDR (5B:00:FA:C2) via $phase_1$ and $phase_2$. In $phase_3$, they filtered the OUI list for vendor prefixes ending in 5B and got 41 brute-force candidates. In their practical experiment, it took *at most two minutes* to find a hidden Bluetooth device using the techniques defined in $phase_1$, $phase_2$ and $phase_3$. Therefore, it takes only an *average of one minute* to find a hidden Bluetooth device.

Attacks based on using Bluetooth worms or viruses can be classified as *integrity and DoS threats*. Recently *Bluetooth worms and viruses* have been often mentioned in the media, for example in [Reu05, Sap07], because there are several Bluetooth worms and viruses, such as Cabir [FSe06], Skulls.D [FSe05a] and Lasco.A [FSe05b], which use Bluetooth-compatible mobile phones for infecting other Bluetooth mobile phones.

Cabir (also referred to as *SymbOS/Cabir.A*, *EPOC/Cabir.A*, *Worm.Symbian.Cabir.a*, or *Caribe virus*) is a Bluetooth worm running in Symbian mobile phones which support the Series 60 platform. It arrives via Bluetooth in a target mobile phone's messaging inbox as a *caribe.sis* file, which contains caribe.app (main executable), flo.mdl (system recognizer), and caribe.rsc (resource file). When the user opens the caribe.sis file and chooses to install it, the worm activates and immediately starts searching for new Bluetooth devices to infect. When another Bluetooth device is found, Cabir will start sending caribe.sis file to it. However, the file will not arrive automatically in the target device, because the user has to answer "yes" to the transfer question when the infected device, i.e. the attacking device, is still in range. It is worth noting that Cabir only spreads itself, i.e. it is not designed to do any harmful things such as erasing a target device's files. [FSe06]

Skulls.D (also referred to as *SymbOS/Skulls.D*) is a malicious SIS (Symbian Installation System) file trojan that pretends to be Macromedia Flash player for Symbian mobile phones which support the Series 60 platform. It arrives in the target mobile phone via Bluetooth in a

similar way to Cabir. When the user opens the SIS file and chooses to install it, the SymbOS/Cabir.M worm, i.e. the variation of the Cabir worm described earlier, will be installed in the target mobile phone, both the system applications and third party applications that are needed to disinfect viruses and worms will be disabled, and animation showing a flashing skull picture will also be displayed on the background of the target device's display during every application that the user is trying to use. When the worm is activated, it immediately starts searching for new Bluetooth devices to infect. [FSe05a]

Lasco.A (also referred to as *SymbOS/Lasco.A* or *EPOC/Lasco.A*) is a Bluetooth worm and a SIS file infecting virus running in Symbian mobile phones which support the Series 60 platform. It arrives in the target mobile phone via Bluetooth in a similar way to Cabir and Skulls.D. When the user opens the velasco.sis file and chooses to install it, the worm activates and immediately starts searching for new Bluetooth devices to infect. In addition to sending itself via Bluetooth, it is also capable of inserting itself into other SIS files in the target device. Therefore, if infected SIS files are copied to another device and installed, installation of Lasco.A will also start. [FSe05b]

In January 2005, Brazilian software developer Marcos Velasco released all source codes of his Lasco.A worm/virus on his homepage [Vel08], but later on removed them. Lasco.A sources can still be downloaded from many Brazilian file servers. It means that practically anyone can now write their own Bluetooth viruses just by modifying Lasco.A sources. In addition, a mobile phone infected with Cabir, Skulls.D or Lasco.A will try to infect other Bluetooth-enabled devices even if the user tries to disable Bluetooth from the device's settings. Moreover, Bluetooth functionality in Series 60 mobile phones is independent from the GSM side. Therefore, if the infected mobile phone is rebooted, the virus/worm will try to spread itself even if the user does not enter the PIN code. [FSe05b, Vel08]

Normally, Bluetooth worms and viruses require that the user accepts their transfer and installation in the target device. In addition, the target device has to be discoverable. Bluetooth worms and viruses can be very dangerous if the target device is vulnerable to BlueBugging, because in that way an attacker can slip in a virus or worm without alerting the user. Therefore, if a user has a vulnerable Bluetooth device, its firmware/software should be

updated as soon as possible. In addition, the user should not install any unknown software in her Bluetooth device and should use antivirus/firewall software (see Section 3.5) when possible. It is also expected that attackers will exploit the techniques for finding hidden Bluetooth devices in an average of one minute [SpB07a, SpB07b] in order to spread viruses and worms more efficiently.

4.3 Reasons for Bluetooth network vulnerabilities

Overall security in Bluetooth networks is based on *the security of the Bluetooth medium*, *the security of Bluetooth protocols* and *the security parameters used in Bluetooth communication*. There are several weaknesses in the Bluetooth medium, Bluetooth protocols and Bluetooth security parameters which can significantly weaken the overall security of Bluetooth networks.

Subsection 4.3.1 discusses Bluetooth network vulnerability to eavesdropping. Subsection 4.3.2 explains the weaknesses in the encryption mechanisms. Subsection 4.3.3 discusses weaknesses in PIN code selection. Subsection 4.3.4 explains the weaknesses in association models of SSP. Subsection 4.3.5 discusses the weaknesses in Bluetooth device configuration.

4.3.1 Vulnerability to eavesdropping

Because Bluetooth is a wireless RF communication system using mainly omnidirectional antennas, an eavesdropper is often not detected. The eavesdropper and eavesdropping equipment can be very far away from the communicating devices. Unencrypted transmissions are always easy prey for eavesdroppers. All the contents of unencrypted transmissions can be seen clearly. Even if Bluetooth data encryption (see Figure 6 in Section 4.1) is used, all intercepted packets can be recorded for later cryptographical analysis, so the length of the K_C should be as long as possible, i.e. the maximum K_C length of 128 bits should be used when possible.

The range of vulnerability to eavesdropping can be calculated using the formula $d_e = 10^{(P_t + G_r + 40)/(10n)}$, where d_e is the range of vulnerability to eavesdropping, P_t is the

transmit power of the target piconet, G_r is the eavesdropper's antenna gain, and n is the PL exponent. A more precise definition of the range of vulnerability calculations can be found in [Mor02].

Several assumptions must be made before proceeding with the range of vulnerability calculations. These assumptions concern the typical physical properties of Bluetooth piconets. The calculations presented in Table 6 are made based on the following standard assumptions:

1. The target piconet devices are using omnidirectional antennas with a 0 dBi gain.
2. The level of obstacles is either light ($n=2.5$) or moderate ($n=3.0$), and there is 20 dB of additional attenuation on the signal as it exits the building.
3. The sensitivity level of the eavesdropper's radio is -100 dBm (enhanced sensitivity level).
4. The eavesdropper has two antennas, one with a gain of 0 dBi (omnidirectional) and another with a gain of 20 dBi (directional).
5. The target piconet devices use a transmit power of either 0 dBm (class 3 device) or 20 dBm (class 1 device).

Level of obstacles:	n:	Target piconet TX power (dBm):	Eavesdropper's antenna gain (dBi):	Range of vulnerability for eavesdropping (m):
Light	2.5	0	0	40
Light	2.5	20	0	251
Light	2.5	0	20	251
Light	2.5	20	20	1585
Moderate	3.0	0	0	22
Moderate	3.0	20	0	100
Moderate	3.0	0	20	100
Moderate	3.0	20	20	464

Table 6. Range of vulnerability to eavesdropping. [Mor02]

As can be seen from Table 6, it would be easy for an eavesdropper to park a car in a parking lot at a reasonable distance (up to 1585 meters; see Table 6) from the target Bluetooth network, for example near a company's Bluetooth network, and start to intercept all Bluetooth activity within the range of vulnerability by using a laptop and a Bluetooth protocol analyzer.

Consider the following attack scenario. If the actual user data (the payload of a Bluetooth Baseband packet) is sent unencrypted, all the contents of the Baseband packet (an access code, a header and the payload) and other relevant information (packet sequence number, direction of the transmission, frequency information and Bluetooth clock information) can be shown clearly on the screen of the laptop by using a Bluetooth protocol analyzer, for example, as Figure 8 illustrates.

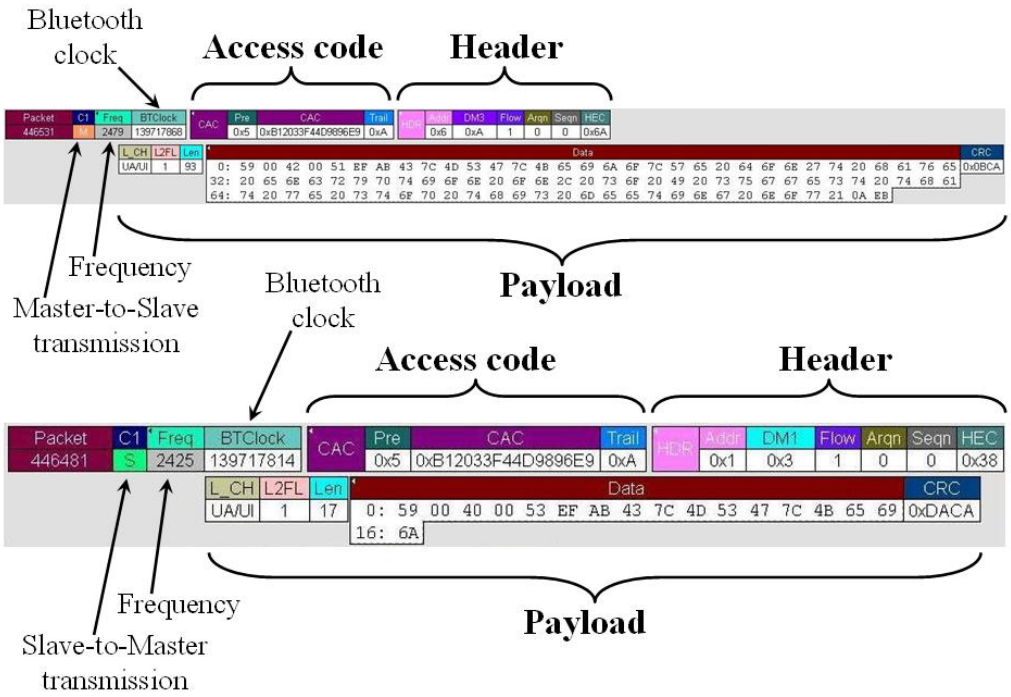

Figure 8. An example of packet interception with a Bluetooth protocol analyzer when Bluetooth encryption is not used. [Haa05a]

The access code and the header are always sent unencrypted, so even when encryption is used an eavesdropper can always see the general piconet information, such as the piconet address of the active slave and the Baseband packet type used, from all of the packets. Based on this information, the eavesdropper may be able to figure out the authorization levels of the legitimate piconet devices, i.e. which Bluetooth device has access to a certain sensitive file.

If the physical protection of the Bluetooth devices is insufficient, the intruder can steal the device and use it to obtain the desired sensitive file. Bluetooth authenticates devices, not users, so this is also very important to keep in mind.

The eavesdropper can also easily see whether the payload is encrypted or not. This can be seen directly from the CRC (Cyclic Redundancy Check) field. In this example (see Figure 8), the received CRC field matches the CRC checksum calculated from the received user data, i.e. the payload is unencrypted. If the CRC field does not match the data, the protocol analyzer displays the CRC field in red, for example, indicating that the payload is encrypted.

4.3.2 Weaknesses in encryption mechanisms

As described in Section 4.1, Bluetooth encryption has many strengths, but it has also a few weaknesses. Perhaps the most significant weakness occurs when 128-bit encryption cannot be used.

When two Bluetooth devices negotiate the parameters for encryption, the length of the encryption key is restricted by the Bluetooth device that has the shorter maximum encryption key length. For example, if one Bluetooth device can support only a 32-bit encryption key, the other Bluetooth device has to adjust to the situation and also use a 32-bit encryption key − otherwise encryption cannot be used at all.

Let us assume that an eavesdropper wants to use a brute-force method to find out a correct encryption key. This can be done by intercepting several encrypted Baseband packets and trying to decrypt them using a keystream generated by different experimental keys. Even though the keystream generator function has the piconet master's BD_ADDR and clock information as additional inputs, both of these values are supposed to be known by the

eavesdropper. The eavesdropper can mark every incoming packet with its associated clock information before storing the packet, and if a packet contains any plaintext, the correct keystream will reveal it. The eavesdropper can use a dictionary, for example, in his key searching algorithm to find out when a string contains real words.

An eavesdropper can also calculate a 16-bit CRC checksum from the payload, and check whether it matches the received CRC field. If the CRC checksums match, then decryption has been successful (if there are no bit errors in payload caused by the packet transfer via the RF link). Almost all commercially available Bluetooth protocol analyzers can automatically check whether the CRC checksums match, so this is not a problem for the eavesdropper.

Table 7 illustrates an encryption weaknesses. The average search time, when using a naive guess-and-try *brute-force method*, measured in seconds is $2^{L-1}/n$, where L is the length of the encryption key in bits and n is the number of key search trials per second. Let us assume that an eavesdropper can make 2^{20} key search trials per second ($n=2^{20}$) on the processing power of one computer, or 2^{40} key search trials per second ($n=2^{40}$) by using parasitic computing over the Internet, for example. Further, let us assume that the eavesdropper's key searching algorithm can in all cases realize when the correct encryption key has been found. If the average search time is determined by 2^{20} trials per second, an adequate level of security can be achieved by using a 64-bit or longer encryption key. If the average search time is determined by 2^{40} trials per second, an adequate level of security can be achieved by using at least an 80-bit encryption key. [Mor02, Sta03]

K_C length (bits):	Average search time at 2^{20} trials per second:	Average search time at 2^{40} trials per second:
8	≈ 122 microseconds	≈ 116 picoseconds
16	≈ 31 milliseconds	≈ 30 nanoseconds
24	≈ 8 seconds	≈ 8 microseconds
32	≈ 34 minutes	≈ 2 milliseconds
40	≈ 6 days	≈ 500 milliseconds
48	≈ 4 years	≈ 128 seconds
56	≈ 1090 years	≈ 9 hours
64	≈ *278922 years*	≈ 97 days
72	≈ 71 million years	≈ 68 years
80	≈ 18 billion years	≈ *17433 years*
128	≈ 5.1×10^{24} years	≈ 4.9×10^{18} years

Table 7. Encryption weaknesses. [Mor02, Sta03]

Let us assume that there are four slave devices (slaves *A*, *B*, *C* and *D*) and one master device in a Bluetooth piconet. Slaves *A*, *B*, *C* and *D* want to use 32-bit, 64-bit, 128-bit, and 128-bit encryption with the master, respectively. As can be seen from Table 7, slave *A* has only primitive protection against eavesdroppers. In addition, an eavesdropper may be able (at least in theory) to decrypt the packets of slave *B* in a reasonable time, because it takes only an average of 97 days assuming that the average search time is determined by 2^{40} trials per second. All this can be achieved with a naive guess-and-try method.

A more sophisticated method for the eavesdropper would be exploiting the results described in [ShW05] (see Subsection 4.2.1), in which the improved method yields approximately 30% faster decryption times: the guess-and-try method's 97 days versus the more sophisticated method's 68 days. Moreover, if slaves *A* and *B* are exchanging sensitive files with slaves *C* and *D*, the better security of slaves *C* and *D*, i.e. the protection of 128-bit encryption, is also

lost. Therefore, if security is very important, the master should not accept encryption key lengths shorter than 128 bits.

4.3.3 Weaknesses in PIN code selection

The weakest point in the security chain of events (see Figure 4 in Section 4.1) of Bluetooth versions up to 2.0+EDR is the first phase, when a user selects a PIN code. The PIN code can be as long as 128 bits (16 bytes), so it can contain up to sixteen 8-bit characters. However, long PIN codes are quite hard to remember, so users usually use only four digits. This makes an eavesdropper's work much easier, because she needs to go through only 10000 possible PIN values and witness the initial pairing process between the target devices in order to get all the required information for various attacks (see Subsections 4.2.1-4.2.4). It is worth noting that the attacker needs only an average of 5000 PIN guesses to find out the correct value when a four-digit PIN code is used (see Section 3.1). On the other hand, if the user decides to use sixteen 8-bit characters, it is very likely that she will write down the PIN code on a piece of paper. This is another weak point, because this piece of paper must be kept secret.

A 16-digit PIN code composed of the characters *0,...,9* achieves $16 \times \log_2 10 \approx 53$ *bits* of entropy, while a PIN code of 16 case-sensitive alphanumerical characters yields $16 \times \log_2 62 \approx 95$ *bits* of entropy when a 62-character set is used, i.e. the 62-character set consists of the characters *0,...,9*, *'a',...,'z'* and *'A',...,'Z'*. Therefore, a Bluetooth PIN code that achieves 128 bits of entropy can be provided by using sixteen 8-bit characters, i.e. a 256-character set is used ($16 \times \log_2 256 = 128$ *bits*). For example, the extended ASCII (American Standard Code for Information Interchange) character set has 256 characters (2^8=256), so there are eight bits of entropy for each 8-bit character (byte).

Two devices become *paired* when they start communicating with the same PIN code, generate the same link key, and then use the link key for authenticating at least the current communication session (see Figures 4 and 5 in Section 4.1). When devices are paired, they can either store their link keys for use in subsequent authentications, or discard them and repeat the pairing process each time they connect. If the link keys are stored, the devices are *bonded*. Users that are using bonded Bluetooth devices do not have to remember long PIN

codes. On the other hand, these bonded devices can be a security risk if the physical protection of the devices is insufficient.

Many different kinds of Bluetooth devices, such as headsets, keyboards and printer adapters, have very short fixed PIN codes, often only four digits long. This is clearly a big security risk, so Bluetooth device manufacturers should take security issues more seriously. We strongly recommend that the sixteen 8-bit character PIN codes should be used when possible. However, in case of a limited User Interface, it may not be possible to use the 256-character set for providing a PIN code that achieves 128 bits of entropy. Moreover, user understanding of security issues is very important for protecting sensitive data against eavesdroppers and hackers. Many users have no idea how to configure their Bluetooth devices' security settings correctly.

4.3.4 Weaknesses in association models of SSP

As described in Section 4.1, Bluetooth 2.1+EDR specification supports SSP to improve the security of pairing by providing protection against passive eavesdropping and MITM attacks (see Section 3.4). SSP uses four association models: *OOB*, *Numeric Comparison*, *Passkey Entry* and *Just Works* (see Section 4.1). The choice of association model depends on the device's IO capabilities (see Table 3 in Section 4.1).

Perhaps the most significant weakness occurs when at least one of the devices has neither input nor output capability and an OOB cannot be used. In this case, the Just Works association model is used, in which the user is simply asked to accept the connection. Therefore, the Just Works association model clearly provides no MITM protection.

SSP consists of six phases: *Capabilities exchange*, *Public key exchange*, *Authentication stage 1*, *Authentication stage 2*, *Link key calculation*, and *LMP authentication and encryption* (see Section 4.1). The last phase in SSP is the same as the final steps of pairing in Bluetooth versions up to 2.0+EDR (see Figures 4 and 6 in Section 4.1). Just as in the Bluetooth versions up to 2.0+EDR, the weakest point in a Bluetooth 2.1+EDR device's security chain of events is the first phase. Instead of selecting a PIN code, Bluetooth 2.1+EDR devices will exchange their IO capabilities (see Table 3 in Section 4.1) to determine the proper association model to

be used. If an OOB, Numeric Comparison or Passkey Entry association model is used, the MITM protection will be automatically provided. However, it has been shown that MITM attacks against Bluetooth 2.1+EDR devices are possible by forcing the victim devices to use the Just Works association model (see Section 6.11). This can be done by using one of the three different MITM attack scenarios that are defined in [HyH07]. By far the best way to prevent MITM attacks is to use NFC as an OOB channel (see Section 2.1).

4.3.5 Weaknesses in device configuration

The default settings of Bluetooth devices usually provide no security at all: the device is set as discoverable (i.e. public security level) and nonsecure (i.e. nonsecure security mode). Therefore, an attacker can discover the BD_ADDR of the target device in a few seconds and perform various attacks (see Subsections 4.2.1-4.2.4) against it. It is worth noting that Bluetooth is rarely switched on by default, so a user has to switch it on from the device's settings before any Bluetooth attacks against that device are possible. Moreover, many users want to save the batteries of their Bluetooth devices so Bluetooth is often switched off when there is no need to use it for a long time.

It is very important that users know how to configure their Bluetooth devices correctly to achieve the best available level of security. In addition, Bluetooth device manufacturers should implement their Bluetooth devices in a more secure way by default factory settings: for example, if a device uses a fixed PIN code, it should be as long as possible and also as hard as possible to guess. If both communicating devices support SSP and NFC, NFC should always be used as an OOB channel. Moreover, application layer key exchange and encryption methods (see Chapter 3) can be used as an extra security in addition to the Bluetooth built-in security (see Section 6.1).

In the following two chapters we examine security attacks in more depth. In Chapter 5 we explain our Bluetooth security laboratory in which several security attacks were demonstrated. In Chapter 6 we explain practical experiments carried out in our Bluetooth security laboratory. We describe new Bluetooth security analysis tools and introduce new Bluetooth security attacks. Based on practical experience, we propose countermeasures

against these attacks and provide Bluetooth vulnerability evaluation. In addition, we provide a comparative analysis of the existing MITM attacks on Bluetooth. Moreover, we describe a novel system for detecting and preventing intrusions in Bluetooth networks, and we also provide a further classification of Bluetooth-enabled ad-hoc networks depending on a risk analysis within each classified group.

5 THE BLUETOOTH SECURITY LABORATORY

Our research work deals with weaknesses in the Bluetooth medium, Bluetooth protocols and Bluetooth security parameters. Currently, weaknesses in Bluetooth security parameters seem to be the biggest problem in Bluetooth security (see Subsections 4.2.1-4.2.4 and Section 4.3). Our practical research work requires a testing environment and Bluetooth equipment in order to get results. Therefore, we built our Bluetooth research laboratory environment for implementing and demonstrating Bluetooth security attacks in practice (see Sections 6.1-6.11).

Our Bluetooth research laboratory environment consists of the following hardware:

- *A CATC (LeCroy) Protocol Analyzer System 2500H* [Lec07]: Due to the branching hierarchical protocol structure of Bluetooth technology (see Figure 3 in Section 2.4), it is one of the most complicated protocols from a design and analysis point of view. To simplify the analysis process, a Bluetooth protocol analyzer is needed. Our CATC Protocol Analyzer System 2500H is a flexible and efficient integrated environment providing features such as 512 MB of recording memory, Hi-Speed USB 2.0 Interface to the host PC/laptop, upgradeable firmware/BusEngine/Baseband, and support for plug-in modules. We also have *two LeCroy Bluetooth 1.1/1.2 compatible radio units* as plug-in modules for the CATC Protocol Analyzer System 2500H (see Figures 9 and 10). It is similar to two separate Bluetooth protocol analyzers, but better because of a common Bluetooth clock, i.e. separate traces are easier to combine after the capture. In addition, eavesdropping (see Figure 8 in Subsection 4.3.1) and jamming of Bluetooth scatternet (see Figure 2 in Section 2.2) or two separate piconets can be done simultaneously. Moreover, our Bluetooth protocol analyzer can emulate any Bluetooth device and it can also clone any BD_ADDR. Therefore, it is a good tool for implementing Bluetooth security attacks in practice. However, the price of a CATC Protocol Analyzer System 2500H with two radio units is very high: some tens of thousands of dollars. Our protocol analyzer works only in Windows environments.

- *A Frontline FTS4BT Wireless Bluetooth Protocol Analyzer & Packet Sniffer* [Fro08]: Our FTS4BT Protocol Analyzer currently supports Bluetooth versions up to 2.1+EDR. It runs on a laptop or a PC using finger-sized Bluetooth USB ComProbes as the air sniffing hardware. Therefore, it provides a very easy and flexible mobile Bluetooth sniffing environment for various practical Bluetooth security experiments. Our FTS4BT is equipped with three ComProbes: it is equivalent to three Bluetooth protocol analyzers, thus making simultaneous eavesdropping of several piconets or an entire scatternet possible. However, the price of FTS4BT Protocol Analyzer with three ComProbes is quite high: over ten thousand dollars. In addition, it is only a sniffer and therefore it does not support the BD_ADDR cloning feature. Moreover, it works only in Windows environments with the official sniffing software. However, as described in Section 4.2, it is possible to transform a standard $30 Bluetooth dongle into a full-blown Bluetooth sniffer [Blu07d, Mos07] and sniff Bluetooth traffic also in Linux environments by using the tool called *frontline* [Dar07]. We verified this in our laboratory by replacing (upgrading) the official firmwares of various CSR-based Bluetooth USB dongles with the official FTS4BT ComProbe firmwares, then sniffing with them in Windows via the official FTS4BT software and in Linux via the *frontline* tool. We also tested how the official FTS4BT ComProbes will sniff in Linux, and they proved to work without any problems. Surprisingly, the $30 Bluetooth USB dongles also worked as if they were the original FTS4BT ComProbes, i.e. without any problems, both in Windows and in Linux. Therefore, it is a very economical solution to transform dozens of standard Bluetooth USB dongles to work as a Bluetooth protocol analyzer in Linux environments. In addition, the source code of the *frontline* tool has been released, thus finally giving everyone a chance to do practical Bluetooth security research. It is also possible to write custom firmwares that include the techniques for finding hidden Bluetooth devices in an average of one minute [SpB07a, SpB07b], thus making the free Linux protocol analyzer a very powerful and practical research tool. Moreover, in Linux environments it is very easy to add a *frontline* tool to support BD_ADDR cloning feature to make impersonation attacks and research possible.

- *Bluetooth user devices*: Our practical research work requires several Bluetooth devices in order to get results. Therefore, we have Bluetooth software development kits, Bluetooth USB adapters, Bluetooth PCMCIA (Personal Computer Memory Card International Association) cards, Bluetooth Hands-Free and Headset devices, Bluetooth mobile phones, a Bluetooth mouse, a Bluetooth keyboard, Pocket PCs with Bluetooth CF (Compact Flash) cards, Pocket PCs with embedded Bluetooth chips, and Bluetooth printer adapters.

- *Bluetooth scatternet environment*: In order to create a real Bluetooth communication environment, we need Bluetooth network traffic. Therefore, we have *14 PCs* and several *laptops*. The PCs are connected with each other via Bluetooth, i.e. they form a Bluetooth scatternet. The laptops are mobile workstations and can be connected to any PC via Bluetooth. We also have a *server PC* for storing and restoring harddisk images when needed. The physical protection of Bluetooth devices is also very important to prevent unauthorized use of bonded Bluetooth devices. Therefore, we have a *locker* for safely storing Bluetooth equipment when they are not used. This environment clearly provides a very easy and fast way to start a new Bluetooth research project.

Our Bluetooth research laboratory environment also has the following software:

- *LeCroy BTTracer/Trainer v2.2 software* [Lec07]: It is needed in the PC/laptop that is using the protocol analyzer. The software also provides *CATC Scripting Language* [Lec04], which allows users to create C language scripts that work with LeCroy protocol analyzers. It is quite easy to automate the use of a protocol analyzer by creating a suitable script to do the desired work. CATC Scripting Language supports all the functionality of our Bluetooth protocol analyzer. Therefore, it is an easy and efficient way for implementing Bluetooth security attacks in practice.

- *Frontline FTS4BT v.8.7.12.0 software* [Fro08]: It is needed in the laptop or PC that is using the FTS4BT protocol analyzer in Windows environment.

- *A Frontline v. 1.1.1.1 tool* [Dar07]: It is needed in the laptop (or PC) that is using the official FTS4BT ComProbes or modified $30 Bluetooth dongles in Linux environments for Bluetooth sniffing.

- *Bluetooth Chat Software* [Haa05a]: We created our own IRC-style (Internet Relay Chat) [IRC08] chat software that consists of *BTChatd* (a Bluetooth Chat server for Linux), *BTChat* (a Bluetooth Chat client for Linux) and *BTChatJava* (a Bluetooth Chat Java client for Linux and Windows). Bluetooth Chat Software runs on Linux/Windows and it requires only normal PCs/laptops and Bluetooth USB dongles to work. It provides an easy way to demonstrate the importance of data encryption, and to show how easy it is for an eavesdropper to intercept all packets exchanged via air (see Section 6.1).

- *An On-Line PIN Cracking Script* [Haa05b]: Our proof-of-concept Bluetooth security analysis tool that makes an On-Line PIN Cracking attack [Haa05b, Whi04] possible (see Subsection 4.2.4 and Section 6.3). The script was created using the CATC Scripting Language.

- *An On-Line PIN Cracking Security Analysis Tool* [Haa07a]: Our new proof-of-concept Bluetooth security analysis tool that also makes an On-Line PIN Cracking attack possible, and works in Linux environments (see Subsection 4.2.4 and Section 6.3) instead of Windows, i.e. an expensive Bluetooth protocol analyzer is not required.

- *A Brute-Force BD_ADDR Scanning Script* [Haa05b]: Our proof-of-concept Bluetooth security analysis tool that makes a Brute-Force BD_ADDR Scanning attack [Haa05b, Whi04] possible (see Subsection 4.2.4 and Section 6.4). The script was created using CATC Scripting Language.

- *A BTPrinterBugging via Packet Interception Security Analysis Tool* [Haa07b]: Our new proof-of-concept Bluetooth security analysis tool that makes BTPrinterBugging via Packet Interception attack (see Subsection 6.7.1) possible.

- *A BTPrinterBugging via Impersonation Security Analysis Tool* [Haa07b]: Our new proof-of-concept Bluetooth security analysis tool that makes a BTPrinterBugging via Impersonation attack (see Subsection 6.7.2) possible.

- *A BTPrinterBugging via Access Denial Security Analysis Tool* [Haa07b]: Our new proof-of-concept Bluetooth security analysis tool that makes a BTPrinterBugging via Access Denial attack (see Subsection 6.7.3) possible.

- *A BTPrinterBugging via Access Denial Security Analysis Tool II* [Haa07b]: Our new proof-of-concept Bluetooth security analysis tool that also makes a BTPrinterBugging via Access Denial attack possible, and works in Linux environments (see Subsection 6.7.3) instead of Windows, i.e. an expensive Bluetooth protocol analyzer is not required.

- *A BD_ADDR Duplication Security Analysis Tool* [Haa07a]: Our new proof-of-concept Bluetooth security analysis tool that makes a BD_ADDR Duplication attack (see Subsection 4.2.3 and Section 6.8) possible.

- *A SCO/eSCO Security Analysis Tool* [Haa07a]: Our new proof-of-concept Bluetooth security analysis tool that makes SCO/eSCO attack (see Subsection 4.2.3 and Section 6.9) possible.

- *A Big NAK Security Analysis Tool* [Haa07a]: Our new proof-of-concept Bluetooth security analysis tool that makes a Big NAK attack (see Subsection 4.2.3 and Section 6.10) possible.

Figure 9 illustrates some equipment in our Bluetooth research laboratory: the Bluetooth scatternet environment consisting of several Bluetooth compatible PCs, Bluetooth USB dongles, Bluetooth PCMCIA card, Bluetooth headset, Bluetooth mobile phone, and Bluetooth protocol analyzer with two radio units. All research work and practical experiments were performed in our Bluetooth security laboratory using our Bluetooth equipment.

Figure 9. Bluetooth scatternet environment and several
Bluetooth devices with adjustable or fixed PIN codes.

Figure 10 illustrates a typical attack where an attacker is eavesdropping unencrypted Bluetooth communication by using a laptop and a Bluetooth protocol analyzer (see Figure 8 in Subsection 4.3.1 for more details). In this example, a Bluetooth-compatible PC and PDA are exchanging data files via Bluetooth using a nonsecure security mode.

Figure 10. A typical eavesdropping attack with a Bluetooth protocol analyzer.

6 PRACTICAL EXPERIMENTS AND VULNERABILITY EVALUATION

In this chapter we discuss why several Bluetooth security attacks are possible and demonstrate them in our practical experiments. Bluetooth vulnerability evaluation is also provided. We analyse the results of the practical experiments, draw conclusions, and propose countermeasures. In addition, we introduce new security analysis tools and present new attacks against Bluetooth security. We also propose countermeasures that render these attacks impractical, although without totally eliminating their potential danger. The scripts and/or source codes of our security analysis tools exist, but they will not be released in any public domain because they can be very dangerous due to their efficiency. Moreover, we provide a comparative analysis of the existing MITM attacks on Bluetooth, describe our novel system for detecting and preventing intrusions in Bluetooth networks, and also provide further classification of Bluetooth-enabled ad-hoc networks depending on a risk analysis within each classified group.

Section 6.1 explains our practical experiment, *Interception of Packets attack* [Haa05a], in which we demonstrate the importance of data encryption and show how easy it is for an eavesdropper to intercept all packets exchanged via air. Another practical experiment, *BlueBugging attack* [Haa06], in which we demonstrate the dangerousness of a BlueBugging attack (see Subsection 4.2.4), is explained in Section 6.2.

Section 6.3 introduces our *On-Line PIN Cracking Security Analysis Tools* [Haa05b, Haa07a] which are, as far as we know, the only security analysis tools for On-Line PIN Cracking so far. Our *Brute-Force BD_ADDR Scanning Security Analysis Tool* [Haa05b], which can be used for discovering hidden (non-discoverable) Bluetooth devices, is introduced in Section 6.4.

Section 6.5 explains our new Bluetooth security attack, *BTKeylogging attack* [Haa05b], which can be used against Bluetooth-enabled keyboards. Another new Bluetooth security attack, *BTVoiceBugging attack* [Haa05b], in which legitimate Bluetooth-enabled devices can be used

as bugging devices, is explained in Section 6.6. Section 6.7 introduces our *BTPrinterBugging Security Analysis Tools* [Haa07b], which make attacks against Bluetooth-enabled printers practical.

Our *BD_ADDR Duplication Security Analysis Tool* [Haa07a], which can be used for performing BD_ADDR Duplication attacks (see Subsection 4.2.3), is explained in Section 6.8. Section 6.9 introduces our *SCO/eSCO Security Analysis Tool* [Haa07a], which can be used for performing SCO/eSCO attacks (see Subsection 4.2.3). Our *Big NAK Security Analysis Tool* [Haa07a], which can be used for performing Big NAK attacks (see Subsection 4.2.3), is explained in Section 6.10.

Section 6.11 introduces our new attacks on Bluetooth SSP, *BT-Niño-MITM attack* [HyH07] and *BT-SSP-Printer-MITM attack* [HaH08], and it also provides *a comparative analysis of the existing MITM attacks on Bluetooth* [HaH08] including our two MITM attacks on Bluetooth SSP. Our novel *Intrusion Detection and Prevention System for Bluetooth Networks* [Haa08a], which can be used for detecting and preventing intrusions in Bluetooth networks, is described in Section 6.12. Finally, Section 6.13 provides *Further Classification of Bluetooth-enabled Ad-hoc Networks* [Haa08b] depending on a risk analysis within each classified group.

6.1 Interception of Packets attack

The purpose of the practical experiment was to demonstrate the importance of data encryption, and to show how easy it is for an eavesdropper to intercept all packets exchanged via air. The equipment needed for our practical experiment were a laptop connected to the LeCroy BTTracer/Trainer protocol analyzer [Lec07], LeCroy BTTracer/Trainer v2.2 software [Lec07], our Bluetooth Chat Software, and PCs with Bluetooth USB dongles (see Chapter 5).

All PCs had a Bluetooth USB adapter and Bluetooth protocol stack up and running. One PC was the piconet master running BTChatd over BlueZ in Linux. The other seven PCs were piconet slaves running BTChat in Linux or BTChatJava in Linux/Windows. Figure 11 illustrates the startup of the BlueZ protocol stack (see rows 1-3, where the notation *hcid* means HCI daemon and *sdpd* means SDP daemon) when encryption is not used (i.e. a

nonsecure security mode), startup of BTChatd (see rows 4-8), and connection establishments of seven slaves (see rows 9-15).

```
(1)  May 27 10:25:25 itmw-ope24 hcid[7092]: HCI daemon ver 2.4 started
(2)  May 27 10:25:25 itmw-ope24 bluetooth: hcid startup succeeded
(3)  May 27 10:25:25 itmw-ope24 bluetooth: sdpd startup succeeded
(4)  May 27 10:25:31 itmw-ope24 btchatd: Serial Port service registered
(5)  May 27 10:25:31 itmw-ope24 btchatd: Logging chat to /tmp/btlog.txt
(6)  May 27 10:25:31 itmw-ope24 btchatd: BtCHATD, BlueZ RFCOMM chat server running!
(7)  May 27 10:25:31 itmw-ope24 btchatd: Waiting for connection on RFCOMM channel 10
(8)  May 27 10:25:31 itmw-ope24 btchatd: Tuomas Kepanen, kepanen@cs.uku.fi
(9)  May 27 10:25:55 itmw-ope24 btchatd: Connection from [00:05:4E:00:68:64]
(10) May 27 10:26:52 itmw-ope24 btchatd: Connection from [00:05:16:48:0C:F5]
(11) May 27 10:27:07 itmw-ope24 btchatd: Connection from [00:05:16:48:12:4C]
(12) May 27 10:27:32 itmw-ope24 btchatd: Connection from [00:05:16:48:0C:F2]
(13) May 27 10:27:48 itmw-ope24 btchatd: Connection from [00:05:16:48:0C:F3]
(14) May 27 10:28:02 itmw-ope24 btchatd: Connection from [00:05:16:48:10:58]
(15) May 27 10:29:32 itmw-ope24 btchatd: Connection from [00:05:16:48:12:55]
```

Figure 11. The startup of the BlueZ protocol stack when encryption is not used, the startup of BTChatd, and connection establishments of seven slaves. [Haa05a]

BTChatd, BTChat and BTChatJava use RFCOMM to emulate a standard serial port. The chat software is implemented on top of the RFCOMM, because it is supported in every Bluetooth protocol stack, and applications over the RFCOMM are easy to implement. Because the piconet master is in a nonsecure security mode, each of the slaves can establish a connection without authentication, authorization and encryption, i.e. the link is unprotected.

Figure 12 illustrates the chat session between the active piconet slaves using BTChatJava on Windows when encryption is not used. The purpose of the chat software is to enable IRC-style communication between a maximum of seven active piconet slaves via the piconet master, which only relays messages to all the connected slaves.

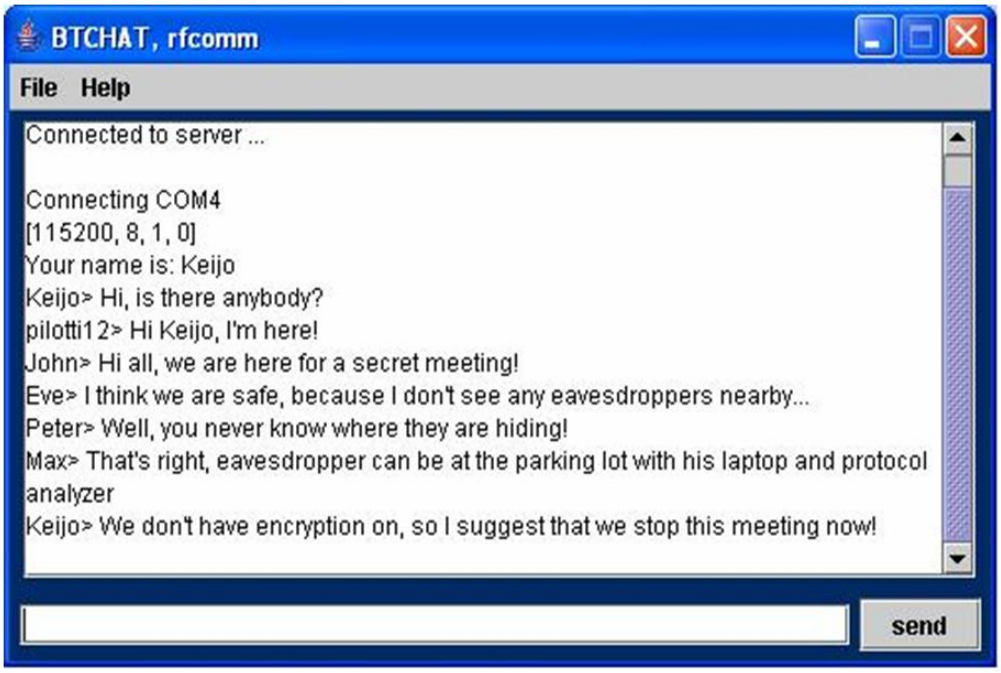

Figure 12. The chat session between the active piconet slaves when encryption is not used. [Haa05a]

An eavesdropper has to synchronize with the piconet master in order to intercept the packets exchanged via air. Commercially available Bluetooth protocol analyzers, such as LeCroy's Bluetooth protocol analyzer, usually require only a brief contact with the piconet master to extract the BD_ADDR of the master and hop sequence information. It can be done by using a general inquiry in which all devices in range will return their FHS (Frequency Hop Synchronization) packets, for example. Therefore, if the security level of the piconet master is set as public, the required synchronization with the master can be done very quickly.

On the other hand, a serious eavesdropper does not want to transmit anything that might disclose her location or intentions. It is also possible to synchronize with the piconet's hop sequence without transmitting at all. It can be done in the following way, for example. The eavesdropper listens to one of the 32 inquiry hop frequencies in order to detect an inquiry. When one is detected, the eavesdropper's radio begins to hop along with the inquirer, checks

each response frequency, and records the FHS packet of each responder. In this way, over a period of time, the eavesdropper will discover the identities of the Bluetooth devices that are within the range of vulnerability.

More difficult for an eavesdropper is the situation where all the target piconet devices are configured as non-discoverable devices, because no inquiries will occur in the range of vulnerability, i.e. the security level is set as private for each device. In this case, the eavesdropper has to monitor traffic on various hop frequencies perhaps over a long period of time in order to intercept a page or FHS packet. A page packet informs the eavesdropper that a page process, including the transmission of FHS packet, is taking place. It is also possible to use several receivers in parallel to increase the probability of intercepting all useful information: for example, 79 receivers will intercept all Bluetooth activity within the range of vulnerability. Moreover, as described in Section 4.2 and in Subsection 4.2.4, techniques for finding hidden Bluetooth devices in an average of one minute have been developed. It is expected that in the near future they will be ported onto a standard CSR dongle via a custom firmware.

Figure 13 illustrates the results of an Interception of Packets attack when encryption is not used. It is very clear that an eavesdropper can easily understand the content of the intercepted data and save it to a text file for later use, for example. The same message is shown several times on the screen of the eavesdropper's laptop, because the piconet master relays all messages to all participants of the chat session, i.e. the eavesdropper has to synchronize only with the piconet master in order to eavesdrop on a chat session between all active piconet slaves. The 16-bit CRC field calculated from the Baseband packet payload also matches the received CRC field, because our Bluetooth protocol analyzer displays the CRC field in white, i.e. white in the CRC field is used to indicate the match between the CRC checksums, while red indicates a mismatch between the CRC checksums. Now the eavesdropper knows for sure that the messages are being sent via a nonsecure link.

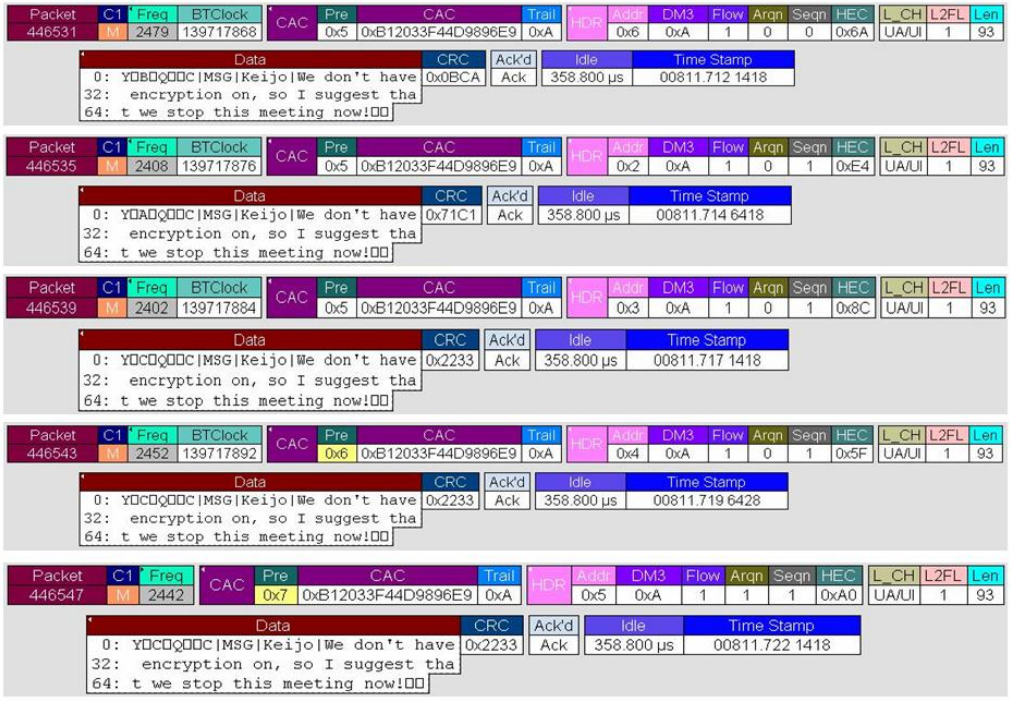

Figure 13. The results of an Interception of Packets
attack when encryption is not used. [Haa05a]

Figure 14 illustrates the startup of the BlueZ protocol stack (see rows 1-5) when encryption is used (Bluetooth security mode 2 is used), the startup of BTChatd (see rows 6-10), and the connection establishments of the piconet slaves (see rows 11-30, where the notation *sba* means the BD_ADDR of the source device, i.e. the BD_ADDR of the master device, and *dba* means the BD_ADDR of the destination device).

```
(1)  May 27 12:11:23 itmw-ope24 hcid[7374]: HCI daemon ver 2.4 started
(2)  May 27 12:11:23 itmw-ope24 hcid[7374]: Starting security manager 0
(3)  May 27 12:11:23 itmw-ope24 bluetooth: hcid startup succeeded
(4)  May 27 12:11:23 itmw-ope24 sdpd[7380]: sdpd v1.5 started
(5)  May 27 12:11:23 itmw-ope24 bluetooth: sdpd startup succeeded
(6)  May 27 12:11:25 itmw-ope24 btchatd: Serial Port service registered
(7)  May 27 12:11:25 itmw-ope24 btchatd: Logging chat to /tmp/btlog.txt
(8)  May 27 12:11:25 itmw-ope24 btchatd: BtCHATD, BlueZ RFCOMM chat server running!
(9)  May 27 12:11:25 itmw-ope24 btchatd: Waiting for connection on RFCOMM channel 10
(10) May 27 12:11:25 itmw-ope24 btchatd: Tuomas Kepanen, kepanen@cs.uku.fi
(11) May 27 12:11:40 itmw-ope24 hcid[7374]: link_key_request (sba=00:05:16:48:0C:FD, dba=00:05:4E:00:68:64)
(12) May 27 12:11:40 itmw-ope24 hcid[7374]: pin_code_request (sba=00:05:16:48:0C:FD, dba=00:05:4E:00:68:64)
(13) May 27 12:11:40 itmw-ope24 hcid[7374]: link_key_notify (sba=00:05:16:48:0C:FD)
(14) May 27 12:11:40 itmw-ope24 hcid[7374]: Saving link key 00:05:16:48:0C:FD 00:05:4E:00:68:64
(15) May 27 12:11:40 itmw-ope24 btchatd: Connection from [00:05:4E:00:68:64]
(16) May 27 12:14:14 itmw-ope24 hcid[7374]: link_key_request (sba=00:05:16:48:0C:FD, dba=00:05:16:48:0C:F5)
(17) May 27 12:14:14 itmw-ope24 hcid[7374]: pin_code_request (sba=00:05:16:48:0C:FD, dba=00:05:16:48:0C:F5)
(18) May 27 12:14:14 itmw-ope24 hcid[7374]: link_key_notify (sba=00:05:16:48:0C:FD)
(19) May 27 12:14:14 itmw-ope24 hcid[7374]: Saving link key 00:05:16:48:0C:FD 00:05:16:48:0C:F5
(20) May 27 12:14:14 itmw-ope24 btchatd: Connection from [00:05:16:48:0C:F5]
(21) May 27 12:16:13 itmw-ope24 hcid[7374]: link_key_request (sba=00:05:16:48:0C:FD, dba=00:05:16:48:12:4C)
(22) May 27 12:16:13 itmw-ope24 hcid[7374]: pin_code_request (sba=00:05:16:48:0C:FD, dba=00:05:16:48:12:4C)
(23) May 27 12:16:13 itmw-ope24 hcid[7374]: link_key_notify (sba=00:05:16:48:0C:FD)
(24) May 27 12:16:13 itmw-ope24 hcid[7374]: Saving link key 00:05:16:48:0C:FD 00:05:16:48:12:4C
(25) May 27 12:16:13 itmw-ope24 btchatd: Connection from [00:05:16:48:12:4C]
(26) May 27 12:17:02 itmw-ope24 hcid[7374]: link_key_request (sba=00:05:16:48:0C:FD, dba=00:05:16:48:0C:F2)
(27) May 27 12:17:02 itmw-ope24 hcid[7374]: pin_code_request (sba=00:05:16:48:0C:FD, dba=00:05:16:48:0C:F2)
(28) May 27 12:17:02 itmw-ope24 hcid[7374]: link_key_notify (sba=00:05:16:48:0C:FD)
(29) May 27 12:17:02 itmw-ope24 hcid[7374]: Saving link key 00:05:16:48:0C:FD 00:05:16:48:0C:F2
(30) May 27 12:17:02 itmw-ope24 btchatd: Connection from [00:05:16:48:0C:F2]
```

Figure 14. The startup of the BlueZ protocol stack when encryption is used, the startup of BTChatd, and the connection establishment of the piconet slaves. [Haa05a]

The startup of the BlueZ protocol stack illustrated in rows 1-5 is similar to that in Figure 11 except that now a Bluetooth security manager (see row 2) is also needed, i.e. if any part of Bluetooth security takes place automatically, a security manager should be part of the host software package. The startup of BTChatd illustrated in rows 6-10 is also similar to that in Figure 11. The Bluetooth devices used in the chat session have never met before, so they do not have the link key, as rows 11-12, 16-17, 21-22 and 26-27 illustrate. Because the piconet master is in the service-level enforced security mode (Bluetooth security mode 3 can also be used if desired), all slaves must enter the PIN code (see rows 12, 17, 22 and 27, in which the master requests each slave to enter the PIN code) that matches with the master's PIN code. After this, the initialization key and the combination key can be generated (see Section 4.1).

The master and each slave perform two-way authentication using the generated combination key, and the results of authentication are used to generate the 128-bit encryption key. Then both ends of the link store the combination key (see rows 13-14, 18-19, 23-24 and 28-29) for later use (i.e. the devices are bonded) and encrypt all data transferred via air. The next time

the same bonded devices communicate with the same master device, they will use the stored combination key for authentication and encryption key generation.

Piconet slaves have a similar chat session again as illustrated in Figure 12, but now the data is encrypted with 128-bit encryption keys. It makes the eavesdropper's work very hard (see Table 7 in Subsection 4.3.2). The eavesdropper has to synchronize again with the piconet master in order to intercept the packets exchanged via air. Even if encryption is used, all packets can be intercepted and stored for later cryptographical analysis.

Figure 15 illustrates the results of an Interception of Packets attack when encryption is used. Now an eavesdropper cannot understand the contents of the intercepted data. The 16-bit CRC field calculated from the Baseband packet payload does not match the received CRC field, so our Bluetooth protocol analyzer displays the CRC field in red to indicate the mismatch between the CRC checksums. Now the eavesdropper knows for sure that the messages are encrypted.

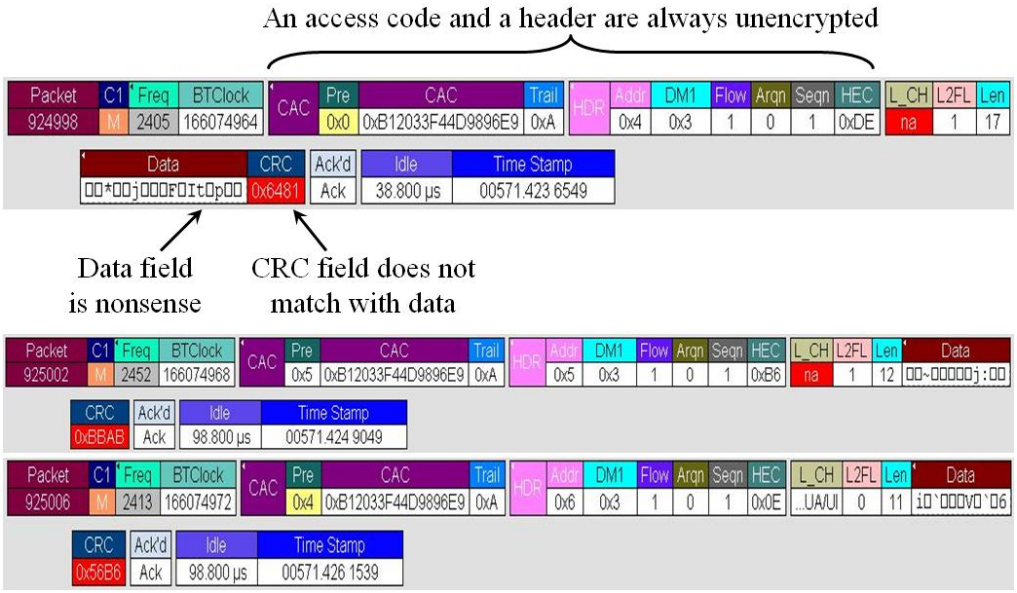

Figure 15. The results of an Interception of Packets attack when encryption is used. [Haa05a]

The countermeasures for an *Interception of Packets attack* are: [Haa06]

- *Data/voice encryption*: All sensitive material should be encrypted with a 128-bit encryption key to prevent unauthorized use (see Table 7 in Subsection 4.3.2).

- *Private or silent security level*: BD_ADDR should not be public, because it is more difficult for an attacker to synchronize with the piconet's hop sequence when the security level is set as private or silent. If there is no need to use the Bluetooth device for a long time, Bluetooth can be switched off completely.

- *Increasing user understanding of security issues*: A user should be aware of Bluetooth security architecture and know how to set it up successfully (see Section 4.1).

- *Minimization of transmit powers*: Sensitive data should be sent using as small transmit power as possible. On the other hand, this increases the range of susceptibility to jamming (see Table 5 in Subsection 4.2.3).

- *Careful selection of place*: This is very important especially when two devices meet for the first time and generate initialization keys (see Figure 4 in Section 4.1).

- *Using only long PIN codes*: Sixteen 8-bit character PIN codes should be used when possible (see Subsection 4.3.3). If a Bluetooth device, such as a headset, a keyboard or a printer adapter, has a fixed PIN code, it should be as long as possible and as hard as possible to guess.

- *Using additional security at application level*: Application layer key exchange and encryption methods (see Chapter 3) can be used as extra security in addition to the Bluetooth built-in security: indeed, our open PKI-based (Public Key Infrastructure) Mobile Payment System [HHH06] can be seen as an example of a real world Bluetooth-enabled system that uses application layer key exchange and encryption methods to secure communication on top of the existing Bluetooth security measures. Although all data exchanged via Bluetooth is encrypted using built-in encryption with 128-bit keys (see Table 7 in Subsection 4.3.2), we use Bluetooth as an untrusted

transport medium. All sensitive data is encrypted on the application level. The integrity and freshness of messages is ensured by digital signatures, timestamps, and nonces. Compared with other mobile payment systems, the main advantage of our system is that it does not require any mediator. This reduces the total cost of a payment. Our system utilizes a governmental PKI infrastructure, namely FINEID (Finnish Electronic Identification), making it an affordable solution, since administration of the system is provided by the government. Furthermore, as citizens have adopted this system for secure electronic transactions, it has a high level of trustworthiness. Our system is built using Java to gain the best possible portability across device platforms. Our solution provides strong authentication of communicating parties, integrity of data, non-repudiation of transactions, and confidentiality of communication. Based on the governmental PKI, the system is open to all merchants, financial institutions and mobile users. More details about our open PKI-based Mobile Payment System can be found in [HHH06].

6.2 BlueBugging attack

Our other practical experiment, *BlueBugging attack* [Haa06], demonstrates the dangerousness of a BlueBugging attack (see Subsection 4.2.4). The equipment needed for the practical experiment were a laptop with Linux Fedora Core 3 and BlueZ protocol stack [Blu08b] installed, one Bluetooth 1.1 compatible USB dongle, and an unmodified Bluetooth 1.1 compatible Nokia 6310i mobile phone [Nok02] (see Chapter 5). In addition, a special tool, btxml [Obe04], was installed on the laptop.

Andreas Oberritter's *btxml* is a tool for a BlueBugging attack. It is capable of stealing the contents of the target mobile phone via Bluetooth and outputting the data in a standard XML (Extensible Markup Language) format. Originally, btxml was designed to work for Nokia 6310 and 6310i mobile phones, but it also works for Ericsson T610 and T68i mobile phones, and may work for some other Bluetooth mobile phones as well. It simply uses GSM AT commands over a RFCOMM connection, and no initial pairing process is required between the attacking device and the target device.

The purpose of this practical experiment was to determine the average time required for a BlueBugging attack by using btxml. Figure 16 illustrates the results of this experiment (see rows 1-27).

```
(1)  <?xml version="1.0" encoding="UTF-8"?>
(2)  <phone btaddr="00:02:EE:B0:29:4D" name="Nokia 6310i">
(3)             <manufacturer>Nokia</manufacturer>
(4)             <model>Nokia 6310i</model>
(5)             <revision>V5.50   03-03-03 NPL-1 (c) NMP. </revision>
(6)             <imei>351453208359469</imei>
(7)             <phonebook name="ME" size="500">
(8)                       <contact>
(9)                                 <name>Test contact number</name>
(10)                                <number>+358501234567</number>
(11)                      </contact>
(12)                      <contact>
(13)                                <name>Another contact number</name>
(14)                                <number>+358447654321</number>
(15)                      </contact>
(16)                      <contact>
(17)                                <name>Yet another contact number</name>
(18)                                <number>+358112233445</number>
(19)                      </contact>
(20)            </phonebook>
(21)            <msgstorage name="ME">
(22)                      <message>"STO UNSENT","", This is a test message for btxml program..</message>
(23)                      <message>"STO UNSENT","", This is another test message...</message>
(24)            </msgstorage>
(25)            <msgstorage name="SM">
(26)            </msgstorage>
(27) </phone>
```

Figure 16. The results of a BlueBugging attack using btxml. [Haa06]

As Figure 16 illustrates, BD_ADDR (see row 2), a user-friendly name (a 1-248 byte user-defined string describing a Bluetooth device; see row 2), the device manufacturer (see row 3), device model (see row 4), firmware version (see row 5), IMEI (International Mobile Equipment Identity) code (see row 6) as well as the entire phonebook (see rows 7-20) and all text messages (see rows 21-26) can be easily discovered and stolen. IMEI can be used for illegal mobile phone cloning.

A BlueBugging attack was repeated 50 times and the average time required for one successful attack when the target mobile phone had three contact numbers and two text messages stored was about 10.7 seconds. For half of the time btxml was performing an inquiry scan, i.e. searching for Bluetooth devices in range. Therefore, the actual time for connecting and stealing information is only about five seconds per BlueBugging attack if the BD_ADDR of the target device is known beforehand. Based on the results of this practical experiment, we can assume that the average time required for a BlueBugging attack varies from five seconds

up to several minutes, depending on the amount of information, such as pictures, music files, text messages, phonebook entries and calendar notes, stored on a vulnerable Bluetooth device. A BlueBugging attack is very dangerous, because millions of vulnerable Bluetooth devices [Her04, LaL04, LHH04, Sap06, Sap07, Spe04], especially Bluetooth mobile phones, are used every day all over the world.

The countermeasures for a *BlueBugging attack*, in addition to those described in Section 6.1, are: [Haa06]

- *Updating latest firmware/software on vulnerable Bluetooth devices*: For example, most of the Bluetooth mobile phone firmware/software versions since summer 2004 are assumed to be safe against a BlueBugging attack and several other Bluetooth security attacks, because most mobile phone manufacturers fixed Bluetooth security flaws (i.e. flaws in the authentication and data transfer mechanisms) of their bad Bluetooth firmware/software implementations during summer 2004.

- *Requiring an additional Bluetooth-independent reauthentication always prior to the access of a sensitive information or service*.

- *Automatic power off capability*: Bluetooth devices with fixed PIN codes should automatically (when possible) turn their power off if no successful connection attempt is made within some predetermined time.

- *Using RF signatures*: Every transmitter has a unique RF signature [Mor02, Sha06], which can be used to differentiate legitimate devices from devices that have alien RF signatures. For this, a sample RF signature is needed from each legitimate device in order to detect alien RF signatures. Bluetooth devices can be equipped with signal processing capabilities to check every RF signature before accepting any connections. On the other hand, this kind of countermeasure can be very expensive if dozens of Bluetooth devices must support it.

6.3 On-Line PIN Cracking Security Analysis Tools

Our On-Line PIN Cracking Security Analysis Tools*, On-Line PIN Cracking Script* [Haa05b] and *On-Line PIN Cracking Tool* [Haa07a], are (as far as we know) the only security analysis tools for an On-Line PIN Cracking attack (see Subsection 4.2.4) implemented so far.

In our first experiment, we successfully performed an On-Line PIN Cracking attack by using a laptop connected to the LeCroy's Bluetooth protocol analyzer [Lec07] with one Bluetooth 1.1 compatible radio unit and Nokia's Bluetooth 1.1 compatible Wireless Headset HDW-2 [Nok03]. A second radio unit will not speed up the process, because only one PIN trial can be performed with the same headset at the same time. However, it can be used for On-Line PIN Cracking with another headset or other Bluetooth device that has a fixed PIN code. LeCroy BTTracer/Trainer v2.2 software [Lec07], which provides CATC Scripting Language [Lec04] (see Chapter 5), was also used in this practical experiment.

CATC Scripting Language was used for creating our *On-Line PIN Cracking Script*, which works in the following way: [Haa05b]

1. Change the local BD_ADDR of the protocol analyzer and set a PIN value for the next PIN trial. In this way, *the ever increasing delay between retries is bypassed* by changing the BD_ADDR of the attacking device every time a PIN guess fails.

2. Create a basic ACL link between the protocol analyzer and the target device.

3. Perform authentication with the target device by using the PIN value set in step 1. If authentication fails, go back to step 1. Otherwise, On-Line PIN Cracking has been completed successfully.

The success of our practical experiment is based on the fact that most Bluetooth headsets use only four-digit fixed PIN codes. Figure 17 illustrates an example of a successful On-Line PIN Cracking attack using our On-Line PIN Cracking Script (see rows 1-25).

```
(1)  HCI_Evt> Write_Authentication_Enable_Complete
(2)  TCI_Evt> CATC_SetBdAddr_Complete
(3)    BD_ADDR                 : 000000002330
(4)  HCI_Evt> PIN_Code_Request
(5)    PIN reply               : 2330
(6)  HCI_Evt> Connection_Error
(7)    Error                   : Authentication Failure
(8)  TCI_Evt> CATC_SetBdAddr_Complete
(9)    BD_ADDR                 : 000000002331
(10) HCI_Evt> PIN_Code_Request
(11)   PIN reply               : 2331
(12) HCI_Evt> Connection_Error
(13)   Error                   : Authentication Failure
(14) TCI_Evt> CATC_SetBdAddr_Complete
(15)   BD_ADDR                 : 000000002332
(16) HCI_Evt> PIN_Code_Request
(17)   PIN reply               : 2332
(18) HCI_Evt> Pairing_Complete
(19)   BD_ADDR                 : 00038935446F
(20) HCI_Evt> Connection_Complete
(21)   BD_ADDR                 : 00038935446F
(22)   HCI Handle              : 0x000B
(23) HCI_Evt> Disconnection_Complete
(24)   BD_ADDR                 : 00038935446F
(25)   Reason                  : No Connection
```

Figure 17. An example of a successful On-Line PIN Cracking attack. [Haa05b]

As Figure 17 illustrates, the protocol analyzer is set to require authentication for each connection with the target device (see row 1). The BD_ADDR value of the protocol analyzer is changed to a new value after every failed authentication attempt (see rows 2-3, 8-9 and 14-15). Two failed authentication attempts are performed with the target device (see rows 4-7 and 10-13). The third authentication attempt is successful (see rows 16-22) and therefore disconnection with the target device can be performed (see rows 23-25). Now an attacker has discovered the fixed PIN code of the target device and further attacks (see Subsections 4.2.1-4.2.4) against that device can be performed.

In our practical experiment, the average time required for one PIN trial was 4.7 seconds including the BD_ADDR change after every PIN trial. Therefore, the average On-Line PIN Cracking time, i.e. the time for 5000 PIN trials (see Subsection 4.3.3) using Bluetooth 1.1 compatible devices was 6.5 hours (5000×4.7s=23500s≈6.5 hours). The same practical experiment with Bluetooth 1.2 or 2.0+EDR devices would have been faster to perform, because Bluetooth specifications 1.2 and 2.0+EDR support faster (less than two seconds per device) connection establishment. Assuming that the average time required for one PIN trial using Bluetooth 1.2 or 2.0+EDR compatible devices is 2.0 seconds, the average On-Line PIN Cracking time for all 5000 PIN trials can be as short as 2.8 hours (5000×2.0s=10000s≈2.8 hours).

Since our On-Line PIN Cracking Script runs only in Windows environments and requires an expensive special hardware, we decided to implement another version of the On-Line PIN Cracking Security Analysis Tool in order to eliminate these restrictions.

This new *On-Line PIN Cracking Tool* [Haa07a] works in Linux environments. It requires the BlueZ protocol stack [Blu08b] and at least one Bluetooth USB dongle to work, i.e. an expensive Bluetooth protocol analyzer is not required. The best performance with the new On-Line PIN Cracking Tool can be achieved when two Bluetooth USB dongles are used simultaneously for an On-Line PIN Cracking attack, i.e. the second USB dongle changes its BD_ADDR while the first USB dongle is performing the On-Line PIN Cracking attack and vice versa.

Our On-Line PIN Cracking Tool uses the standard tools shipped with BlueZ and our own modifications of some BlueZ tools. Manufacturer-specific commands are used to change the BD_ADDR of the attacking device every time the PIN guess fails, i.e. not all Bluetooth USB dongles are supported by our security analysis tool. The supported manufacturers are CSR, TI (Texas Instruments), Ericsson and Zeevo, i.e. most Bluetooth USB dongles in the market support our On-Line PIN Cracking Tool.

Changing the BD_ADDR value of a typical Bluetooth USB dongle takes only an average of two seconds, which is much faster than the time required for one PIN trial. This allows us to

use a simple parallerization technique to speed up the attack, which works as follows. The average time for one PIN trial (excluding the BD_ADDR change) with our On-Line PIN Cracking Tool is ten seconds for the "old" Bluetooth 1.0/1.1 devices and four seconds for the "new" Bluetooth 2.0+EDR devices, which support faster connection establishment. Therefore, a second Bluetooth USB dongle can be used to save an average of two seconds per PIN trial, i.e. the attack can be performed 17%-33% faster (100%×2/12=17% and 100%×2/6=33%) by using two Bluetooth USB dongles in parallel. Figure 18 illustrates our On-Line PIN Cracking Tool in action (see rows 1-16).

```
(1)   PIN Code for the Next PIN Trial Is: 1232
(2)   Connecting to the Remote Bluetooth Device..
(3)   Can't create connection: Input/output error
(4)   New (Local) BD_ADDR Will Be: 29:29:29:29:29:29
(5)   Local BD_ADDR Has Been Changed to New Value!
(6)   Device Reset Has Been Completed Successfully!
(7)   PIN Code for the Next PIN Trial Is: 1233
(8)   Connecting to the Remote Bluetooth Device..
(9)   Can't create connection: Input/output error
(10)  New (Local) BD_ADDR Will Be: 30:30:30:30:30:30
(11)  Local BD_ADDR Has Been Changed to New Value!
(12)  Device Reset Has Been Completed Successfully!
(13)  PIN Code for the Next PIN Trial Is: 1234
(14)  Connecting to the Remote Bluetooth Device..
(15)  Authentication Has Been Successfully Completed!
(16)  PIN Code of the Remote Bluetooth Device Is: 1234
```

Figure 18. The On-Line PIN Cracking Tool in action. [Haa07a]

As Figure 18 illustrates, the On-Line PIN Cracking Tool works in a similar way as the On-Line PIN Cracking Script described in Figure 17. The local BD_ADDR value of the attacking device is changed to a new value after every failed authentication attempt (see rows 4-6 and 10-12). Two failed authentication attempts are performed with the target device (see rows 1-3 and 7-9). The third authentication attempt is successful (see rows 13-16) and therefore an attacker has discovered the secret PIN code of the target device.

The On-Line PIN Cracking Script is faster than the On-Line PIN Cracking Tool, because it runs on a special hardware, LeCroy's Bluetooth protocol analyzer [Lec07], which can use a Bluetooth radio much more efficiently than a normal PC with a Bluetooth USB dongle. On the other hand, it is also a much more expensive approach to On-Line PIN Cracking, thus making our On-Line PIN Cracking Tool a very economical solution.

An On-Line PIN Cracking attack is very feasible and dangerous if the fixed PIN code of the target device is short and has no case-sensitive alphanumerical characters (and perhaps some other characters as well). In addition, an attacker does not necessarily need to go through all PIN values in one day. She can continue the attack some other day when the target device is back within the range of vulnerability. Moreover, Bluetooth specifications up to 2.0+EDR do not provide any proper countermeasures for On-Line PIN Cracking attacks. Therefore, all countermeasures (if any) are up to the device manufacturer. However, Bluetooth 2.1+EDR provides SSP (see Section 4.1) to protect against On-Line PIN Cracking attacks.

One countermeasure for an *On-Line PIN Cracking attack*, provided that all countermeasures from Sections 6.1-6.2 are in place, seems to be:

- *Requiring a button press from the user always prior to accepting the connection establishment*: A Bluetooth connection is accepted only if the user physically confirms the connection establishment by pressing a button.

6.4 Brute-Force BD_ADDR Scanning Security Analysis Tool

The main features of a Brute-Force BD_ADDR Scanning attack were described in Subsection 4.2.4. *RedFang* [Whi03] is a security analysis tool for finding non-discoverable Bluetooth devices by brute-forcing the last three bytes of BD_ADDR and doing a name inquiry. It runs on Linux and requires a BlueZ protocol stack [Blu08b] and at least one Bluetooth USB dongle to work.

Based on the idea of [Whi03], we designed, implemented and tested our own tool to carry out this attack. Our *Brute-Force BD_ADDR Scanning Security Analysis Tool [Haa05b] is on average four times faster than RedFang*, because it runs on a special hardware, LeCroy's Bluetooth protocol analyzer [Lec07], which can use Bluetooth radio much more efficiently than a normal PC with a Bluetooth USB dongle.

We successfully performed a Brute-Force BD_ADDR Scanning attack using LeCroy's Bluetooth protocol analyzer [Lec07] with one Bluetooth 1.1 compatible radio unit and an unmodified Bluetooth 1.1 compatible Nokia 6310i mobile phone [Nok02]. LeCroy BTTracer/Trainer v2.2 software [Lec07], which provides CATC Scripting Language [Lec04] (see Chapter 5), was also used in our practical experiment.

We used CATC Scripting Language to create our *Brute-Force BD_ADDR Scanning Script*, which works in the following way: [Haa05b]

1. Set the scanning area.
2. Set remote BD_ADDR for the next BD_ADDR trial, i.e. set a BD_ADDR value for the next connection attempt.
3. Try to create a basic ACL link between the protocol analyzer and a remote device by using the BD_ADDR value set in step 2. If the connection attempt fails, go back to step 2. Otherwise, the Brute-Force BD_ADDR Scanning Script has found a non-discoverable device (see Section 4.1). Perform a remote name inquiry and a disconnection with the target device. If there is more scanning left to do, go back to step 2.

All scanning information can be stored in a logfile for later analysis. Figure 19 illustrates an example of a successful Brute-Force BD_ADDR Scanning attack using our Brute-Force BD_ADDR Scanning Script (see rows 1-19).

```
(1)  Remote BD_ADDR for this trial is: 0002eeb0294b
(2)  HCI_Evt> Connection_Error
(3)    Error                  : Page Timeout
(4)  Remote BD_ADDR for this trial is: 0002eeb0294c
(5)  HCI_Evt> Connection_Error
(6)    Error                  : Page Timeout
(7)  Remote BD_ADDR for this trial is: 0002eeb0294d
(8)  HCI_Evt> Connection_Complete
(9)    BD_ADDR                : 0002EEB0294D
(10)   HCI Handle             : 0x0004
(11) HCI_Evt> Remote_Name_Request_Complete
(12)   BD_ADDR : 0002EEB0294D
(13)   Name    : "Nokia 6310i"
(14) HCI_Evt> Disconnection_Complete
(15)   BD_ADDR                : 0002EEB0294D
(16)   Reason                 : No Connection
(17) Remote BD_ADDR for this trial is: 0002eeb0294e
(18) HCI_Evt> Connection_Error
(19)   Error                  : Page Timeout
```

Figure 19. An example of a successful Brute-Force BD_ADDR Scanning attack. [Haa05b]

As Figure 19 illustrates, the remote BD_ADDR value is changed to a new value (see rows 1, 4, 7 and 17) after every connection attempt if there is more scanning left to do. Two failed connection attempts are performed (see rows 2-3 and 5-6) before the successful connection establishment (see rows 8-10) in which a remote name inquiry (see rows 11-13) and disconnection (see rows 14-16) with the target device is also performed. Now an attacker has discovered the BD_ADDR of the target device, and further attacks (see Subsections 4.2.1-4.2.4) against that device can be performed. Because there is more scanning left to do, the remote BD_ADDR value is changed again to a new value (see row 17), and a new connection attempt is performed (see rows 18-19).

In our practical experiment, the total time for scanning through 2000 BD_ADDRs was 174 minutes and 20 seconds. Hence, the average time required for one reliable BD_ADDR trial is 5.2 seconds (10460s/2000≈5.2 seconds). The average scanning time for all 8388608 BD_ADDR trials (i.e. a 24-bit address space gives 16777216 different BD_ADDR values and

an attacker needs on average 8388608 BD_ADDR guesses to find out the correct value) with Brute-Force BD_ADDR Scanning Script is 1.4 years (8388608×5.23s≈43872420s≈1.4 years) when using only one radio unit. A second radio unit or another LeCroy Bluetooth protocol analyzer can be used to speed up the scanning process. If, for example, 25 compact size LeCroy Merlin II [Lec07] protocol analyzers are used for a Brute-Force BD_ADDR Scanning attack with our Brute-Force BD_ADDR Scanning Script, it takes on average 20.3 days (43872419.84s/25≈1754897s≈20.3 days).

For comparison, RedFang needs as much as 100 concurrent Bluetooth USB dongles to achieve the same result, and it is very likely that due to greater RF interference they will not work as reliably as 25 concurrent protocol analyzers, i.e. there are only 79 different Baseband frequencies and therefore 100 concurrent Bluetooth USB dongles within the range will cause RF interference. Moreover, an attacker has to use a laptop or laptops with several USB hubs to make this kind of attack feasible.

Brute-Force BD_ADDR Scanning attacks can be feasible and dangerous if an attacker has enough resources, namely equipment, money, time and will, and good software tools such as Brute-Force BD_ADDR Scanning Script or RedFang. In addition, the attacker has to know the manufacturer of the target device, because brute-forcing a 48-bit address space is not feasible. Moreover, just discovering the BD_ADDR of the target device will not give access to any sensitive files. Therefore, some additional attacks must also be performed and the target device must be somehow vulnerable to them.

A Brute-Force BD_ADDR Scanning attack is perhaps most feasible when target devices are Bluetooth mobile phones, since Nokia is the world's leading mobile phone manufacturer. Another good guess for the mobile phone's manufacturer is Motorola, which is the world's second largest mobile phone manufacturer. Moreover, millions of vulnerable Bluetooth mobile phones [Her04, LaL04, LHH04, Sap06, Sap07, Spe04] are used every day all over the world.

An attacker can also try to use a virus that turns all infected Bluetooth-compatible PCs and laptops into scanning devices to speed up the Brute-Force BD_ADDR Scanning attack. Moreover, the attacker can speed up the searching process and increase the probability of finding several Bluetooth mobile phones by first scanning discoverable mobile phones. Based on the BD_ADDRs of the discoverable Bluetooth mobile phones, the attacker can determine the most commonly used company_assigned values (see Subsection 4.2.4) in this particular geographical area, and scan first the BD_ADDRs that are near those of the discoverable mobile phones. It is very likely that the BD_ADDR values are almost the same within the same geographical area, because the process of assigning company_assigned values is not completely random.

Several practical experiments in our Bluetooth security laboratory showed that when testing Brute-Force BD_ADDR Scanning even with very small scanning ranges, such as scanning only 100 or 200 BD_ADDRs that are near the BD_ADDR of our laboratory's Bluetooth mobile phone, typically several additional Bluetooth mobile phones were discovered accidentally.

It is worth noting that besides *RedFang* and *Brute-Force BD_ADDR Scanning Script*, techniques for finding hidden Bluetooth devices in an average of one minute have been developed (Section 4.2 and Subsection 4.2.4), and it is expected that in the near future they will be ported onto a standard CSR dongle via a custom firmware.

The countermeasures for a *Brute-Force BD_ADDR Scanning attack* seem to be the same as described in Sections 6.1-6.2.

6.5 BTKeylogging attack

Our new Bluetooth security attack, *BTKeylogging attack* [Haa05b], extends both the Brute-Force BD_ADDR Scanning attack (see Subsection 4.2.4 and Section 6.4) and On-Line PIN Cracking attack (see Subsection 4.2.4 and Section 6.3). In most cases, a BTKeylogging attack is carried out on a wireless connection between a Bluetooth keyboard, which is also used for typing a PIN code during the initial pairing process, and a PC. This new attack is possible

when the target keyboard has a fixed or short adjustable PIN code and its BD_ADDR is known to an attacker. Moreover, the attacker must witness the initial pairing process between the target keyboard and the target computer. There are different ways to arrange or force target devices to repeat the initial pairing process (see Subsection 4.2.1).

If an attacker uses a Brute-Force BD_ADDR Scanning attack to discover the BD_ADDRs of the target devices (the keyboard and the computer) and then an On-Line PIN Cracking attack to discover the fixed or short adjustable PIN code of the target keyboard, she can use the keyboard as a keylogger by intercepting all packets (i.e. all keypresses) sent via air and decrypting them. A BTKeylogging attack also requires that the attacker intercepts the IN_RAND value, LK_RAND values, AU_RAND value, and EN_RAND value (see Figures 4, 5 and 6 in Section 4.1). Thereafter, all intercepted information can be decrypted.

A BTKeylogging attack was performed in the following way. We discovered the BD_ADDRs of the target devices via a Brute-Force BD_ADDR Scanning attack, and we also discovered the fixed PIN code of the target keyboard via an On-Line PIN Cracking attack. We also used a Bluetooth protocol analyzer (see Chapter 5) to intercept all the required information (the IN_RAND value, LK_RAND values, AU_RAND value, and EN_RAND value) for the BTKeylogging attack. Then the keyboard was used as a keylogger by intercepting all keypresses.

We also successfully decrypted all the intercepted information. As described in Section 4.1, K_{init} can be produced by the formula $K_{init}=E_{22}(PIN',L',IN\_RAND)$. It can be used to decrypt the intercepted LK_RAND values (LK_RAND_A and LK_RAND_B), i.e. $(\text{LK\_RAND} \oplus K_{init}) \oplus K_{init}=\text{LK\_RAND}$. K_{AB} can be produced using the formula $K_{AB}=K_A \oplus K_B=E_{21}(BD\_ADDR_A, LK\_RAND_A) \oplus E_{21}(BD\_ADDR_B,LK\_RAND_B)$, and ACO can be produced by the $E_1(AU\_RAND_A,BD\_ADDR_B,K_{AB})$ function. K_C can be produced using the formula $K_C=E_3(EN\_RAND_A,ACO,K_{AB})$, and finally the keystream can be generated by the $E_0(K_C,CLK_{26-1},BD\_ADDR_A)$ function.

Each intercepted Baseband packet can be decrypted by XORing its encrypted payload with the correct keystream, i.e. $Ciphertext \oplus Keystream=(Plaintext \oplus Keystream) \oplus Keystream= Plaintext$. It is also worth noting that each intercepted Baseband packet must be stamped with the associated $CLK_{26\text{-}1}$ value before storing it (most commercially available Bluetooth protocol analyzers automatically support this feature) in order to produce the correct keystream for each Baseband packet.

Another way of performing a BTKeylogging attack is to use an Off-Line PIN Recovery attack (see Subsection 4.2.1) instead of an On-Line PIN Cracking attack to discover the fixed or short adjustable PIN code of the target keyboard. This approach requires that an attacker intercepts the IN_RAND value, LK_RAND values, AU_RAND value, SRES value, and EN_RAND value, i.e. it requires that the attacker intercepts the SRES value in addition to the values in the example described above. Then the attacker tries to calculate the correct SRES value by guessing different PIN values until the calculated SRES is equal to the intercepted SRES. The SRES can be produced by the $E_1(AU\_RAND_A,BD\_ADDR_B,K_{AB})$ function, i.e. the same function that also produces ACO. As described in Subsection 4.2.1, a SRES match does not necessarily guarantee that the attacker has discovered the correct PIN code, but the chances are quite high especially if the PIN code is short. The other phases of a BTKeylogging attack in this approach are the same as described in our example above.

Some Bluetooth keyboards allow users to change the PIN code of the keyboard, but unfortunately in most cases this new secret PIN code has to be typed at the computer side. Then it is sent via the Bluetooth link to the keyboard and stored. This is clearly a big security risk, because an attacker can intercept the new PIN code by using a Bluetooth protocol analyzer. It means that the attacker will not even need to perform an On-Line PIN Cracking attack or Off-Line PIN Recovery attack to discover the new PIN code.

One countermeasure for a *BTKeylogging attack*, in addition to those described in Sections 6.1-6.3, is: [Haa05b]

- *Changing the PIN code without sending the new PIN code via a Bluetooth link*: PIN code changing should be done directly using the keyboard itself without any help form

the target computer. This can be done by pressing a particular button for a predetermined time (for example, 10 seconds), typing the desired new PIN code, and then pressing the same button again to accept the PIN code storage, for example. It is worth noting that the same new PIN code must also be typed at the computer side using a traditional wired keyboard.

6.6 BTVoiceBugging attack

Another new Bluetooth security attack, *BTVoiceBugging attack* [Haa05b], also extends both a Brute-Force BD_ADDR Scanning attack (see Subsection 4.2.4 and Section 6.4) and an On-Line PIN Cracking attack (see Subsection 4.2.4 and Section 6.3). This attack is possible when the target device has a fixed or short adjustable PIN code, its BD_ADDR is known to an attacker, and it has support for SCO or eSCO links (see Section 2.2).

The main features of the attack are as follows. If an attacker uses a Brute-Force BD_ADDR Scanning attack to discover the BD_ADDR of the target device and then an On-Line PIN Cracking attack to discover the fixed or short adjustable PIN code of the target device, it is possible to open a two-way realtime SCO or eSCO link with the target device. This means that a Bluetooth headset, for example, can be used as a bugging device. In such a case, the attacker can listen to sensitive conversations (for example, important business or other meetings taking place in the vicinity of the target device) via a SCO or eSCO link, and she can also record these conversations for later use. The target device could also be a Bluetooth-enabled PC or laptop with a microphone and speakers located in the conference room where the business meeting is taking place.

Because the link between the attacking device and the target device is two-way, it is also possible to send voice packets, such as a lubricious voice message or a funny music song, to the target device. However, a BTVoiceBugging attack does not make it possible to eavesdrop on the voice communication of another SCO or eSCO link that the same target device is using simultaneously with another device. Therefore, in order to eavesdrop on the voice communication of two target devices, additional attacks, such as an Off-Line Encryption Key

Recovery attack (see Subsection 4.2.1) and Interception of Packets attack (see Section 6.1), are required.

We performed a BTVoiceBugging attack in the laboratory environment in the following way. We discovered the BD_ADDR of the target headset via a Brute-Force BD_ADDR Scanning attack, and we also discovered the fixed PIN code of the target headset via an On-Line PIN Cracking attack, i.e. we obtained all the information required for a BTVoiceBugging attack. Then we used a Bluetooth protocol analyzer (see Chapter 5) to open a two-way realtime SCO connection with the target headset, i.e. the headset was used as a bugging device. We also intercepted all voice packets and exported them to a WAV (Waveform) file that was stored for later use.

We defined three different BTVoiceBugging attack scenarios in order to eavesdrop on a typical business meeting (see Figures 20-22). The first scenario is illustrated in Figure 20.

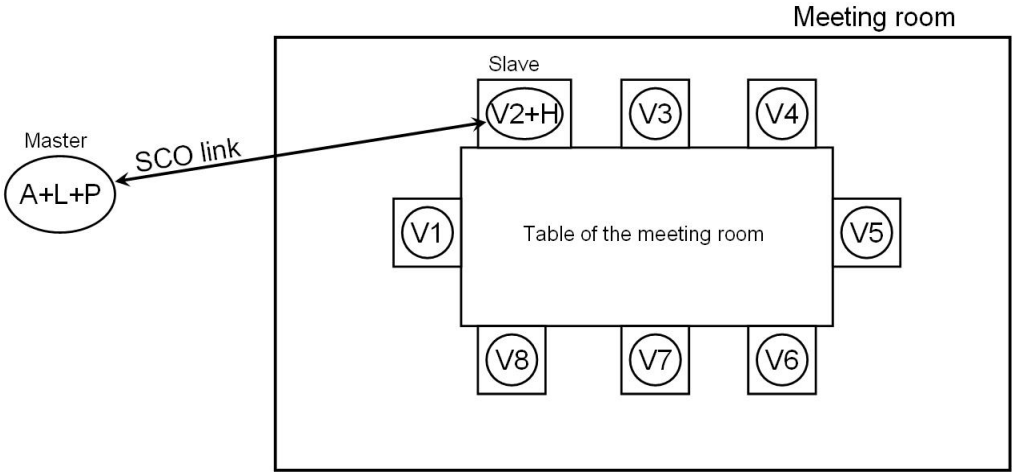

Figure 20. The first BTVoiceBugging attack scenario.

In this scenario, an attacker (A) has a laptop (L) and a Bluetooth protocol analyzer (P). Seven participants (victims V1, V3, V4, V5, V6, V7 and V8) in the business meeting do not have any Bluetooth devices with them. However, one participant (V2) has in his pocket a Bluetooth headset (H), or any other Bluetooth device that supports a SCO/eSCO link and is also

equipped with a microphone/speakers to enable a realtime two-way voice link of Bluetooth, i.e. V2 is not aware that she has a "Bluetooth bugging device" in his pocket. The attacking device (L+P) is a piconet master, because it initiates the connection with V2's headset (H). Correspondingly, H is a piconet slave that established a realtime two-way SCO link with the piconet master (L+P).

The second BTVoiceBugging attack scenario is illustrated in Figure 21. A, L, P, V1, V3, V4, V5, V6, V7 and V8 in this scenario are the same as in the first scenario. A meeting participant, V2, has a Bluetooth headset (H) and a Bluetooth-enabled mobile phone (M) in his pocket. H is connected to M and therefore V2 is able to receive calls wirelessly via H. When V2 is not wirelessly receiving a call, there exists only a basic ACL link between H and M, i.e. a SCO link is automatically established only when a call is received. Therefore, if H supports multiple connections, A can establish a realtime two-way SCO link with it and thus V2 has again a "Bluetooth bugging device" in his pocket. H is a piconet master, because it automatically initiates a connection with M when the power is switched on. Correspondingly, M is a piconet slave that established a basic ACL link with the piconet master (H). The attacking device (L+P) is also a piconet slave, because it joins the existing piconet in which H already is the piconet master.

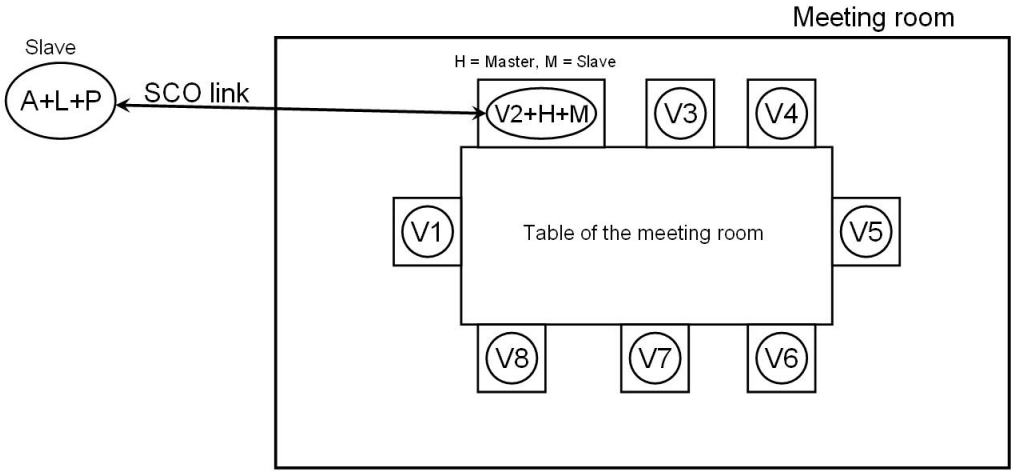

Figure 21. The second BTVoiceBugging attack scenario.

The third BTVoiceBugging attack scenario is illustrated in Figure 22. A, L and P in this scenario are the same as in the first scenario. None of the eight participants (victims V1, V2, V3, V4, V5, V6, V7 and V8) in the business meeting has any Bluetooth devices on her. However, there is a Bluetooth-enabled computer (C) equipped with a microphone (M) and speakers (S) to enable a realtime two-way voice link of Bluetooth, i.e. the participants (victims) are not aware that a "Bluetooth bugging device" is on the table of the meeting room. C can be a PC in the meeting room, used to give PowerPoint presentations via a video projector or video-conferencing services via the Internet. Alternatively, C can be a Bluetooth-enabled laptop or PDA belonging to one of the participants (victims), equipped with M and S, which has been taken into the meeting room "just in case", i.e. it is not necessary that a participant is using C (or any Bluetooth services) during the meeting, but its power must be switched on. The attacking device (L+P) is a piconet master, because it initiates a connection with C. Correspondingly, C is a piconet slave that established a realtime two-way SCO link with the piconet master (L+P).

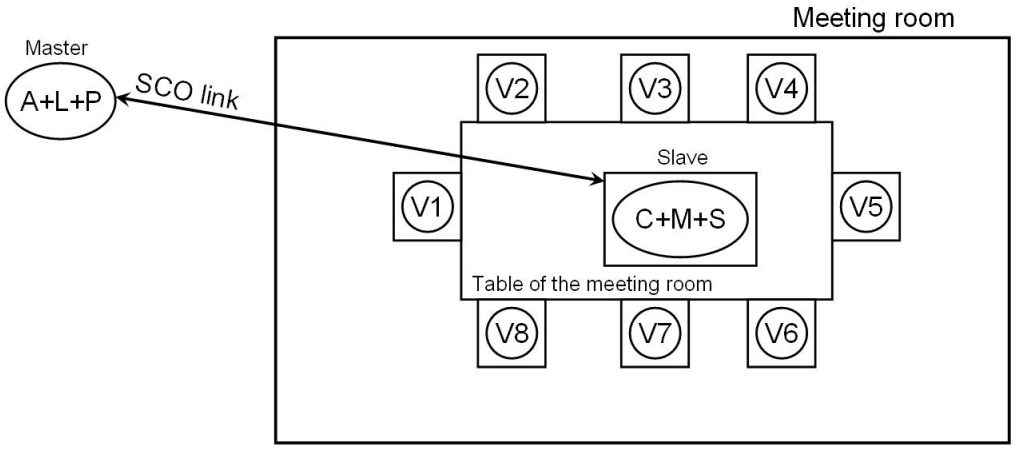

Figure 22. The third BTVoiceBugging attack scenario.

The countermeasures for a *BTVoiceBugging attack*, provided that all the countermeasures from Sections 6.1-6.3 are in place, are: [Haa05b]

- *Switching off all unnecessary SCO/eSCO links*.

- *Requiring an additional Bluetooth-independent reauthentication prior to accepting the establishment of a SCO/eSCO link.*

6.7 BTPrinterBugging Security Analysis Tools

New Bluetooth applications create new security threats, for example in printing. Bluetooth printers and printer adapters are widely used all over the world, especially in home environments. Sensitive infomation such as documents containing personal identification or social security numbers, documents related to business issues, and web pages related to bank account information, are printed via Bluetooth without considering the possible security risks. Moreover, Bluetooth-enabled printers are in most cases always powered on, allowing long-lasting attacks.

The typical communication range of a Bluetooth-enabled printer is up to 100 meters indoors, because in most cases it is a class 1 device (see Section 2.3). Moreover, most Bluetooth-enabled printers have an enhanced sensitivity level, such as -80 dBm or better, and therefore the communication range of such printers is even higher than that of printers with a standard sensitivity level. It means that attacks against Bluetooth-enabled printers can typically be carried out relatively far away from the targets.

Most Bluetooth-enabled printers are configured as discoverable devices (see Section 4.1) by the fixed factory setting, i.e. it is not possible to configure the printer as non-discoverable. Therefore, the BD_ADDR of a Bluetooth-enabled printer can typically be discoverd in a few seconds by an attacker.

Our new BTPrinterBugging attacks are based on the idea that an attacker abuses the target Bluetooth-enabled printer in order to do various harmful things. The attacker can, for example, both intercept and decrypt all the information that is sent to the printer (a *BTPrinterBugging via Packet Interception attack* [Haa07b]; see Subsection 6.7.1), use the printer remotely as if it was her own (a *BTPrinterBugging via Impersonation attack* [Haa07b]; see Subsection 6.7.2), deny access to the printer from the legitimate piconet users

(a *BTPrinterBugging via Access Denial attack* [Haa07b]; see Section 6.7.3), and do many other harmful things.

In Subsections 6.7.1-6.7.3 we describe our new BTPrinterBugging attacks. We demonstrate with experimental figures that attacks against Bluetooth-enabled printers become practical by using our security analysis tools. To remedy the situation, we propose countermeasures that render these attacks impractical although without totally eliminating their potential danger.

The attacks make use of our new efficient implementations of security analysis tools. The laboratory environment in which our practical experiments were performed is also described.

6.7.1 BTPrinterBugging via Packet Interception Security Analysis Tool

If an attacker wants to intercept and decrypt all the information that is sent to a Bluetooth-enabled printer via air, the BD_ADDR of the printer, the BD_ADDR of the legitimate piconet device that is using the printer, and the secret PIN code that is used between these two target devices must be known to the attacker. Moreover, the attacker must intercept the traffic of the initial pairing process between these two devices when they meet for the first time (see Section 4.1).

This kind of attack is normally possible only if the target devices are configured as discoverable devices (see Section 4.1). However, there are also ways to find non-discoverable devices, for example by using a Bluetooth protocol analyzer (see Section 6.1), brute-force scanning (see Subsection 4.2.4 and Section 6.4), or the techniques for finding hidden Bluetooth devices in an average of one minute (see Section 4.2 and Subsection 4.2.4).

Most Bluetooth-enabled printers have only four-digit fixed or short adjustable PIN codes, and in many cases this PIN code is as naive as four zeros, i.e. "0000" in ASCII, which is definitely almost every attacker’s first PIN code guess. In addition, based on the “user-friendly” name (see Section 6.2) and the company_id value (see Subsection 4.2.4) of a Bluetooth-enabled printer, the attacker can determine its manufacturer and the model. With this information, it is very easy for the attacker to download the user manual of the printer from the Internet in order to find out the required fixed PIN code. Moreover, many Bluetooth-enabled printers have no

support for a PIN code at all, i.e. no PIN code is available. According to the Bluetooth specification [Blu07a], the default value of zero, i.e. "0" in ASCII, will be used as a PIN code for Bluetooth security operations (see Section 4.1) if no PIN code is available. Therefore, “0” is also a very good PIN guess for the attacker.

Clearly then, the PIN code of a Bluetooth-enabled printer is in most cases very easy to guess. However, some Bluetooth-enabled printers may have fixed or adjustable PIN codes, which are very hard to guess. In such a case, an On-Line PIN Cracking attack (see Subsection 4.2.4 and Section 6.3) or Off-Line PIN Recovery attack (see Subsection 4.2.1) can be used to discover PIN codes.

As described in Subsection 4.2.1, there are many ways to arrange or force target devices to repeat the initial pairing process. Therefore, the requirement to intercept the traffic of the initial pairing process between two target devices is not a big problem for an attacker.

Let us assume the following attack scenario. First, an attacker uses a Brute-Force BD_ADDR Scanning attack or a Bluetooth protocol analyzer to discover the BD_ADDRs of two target devices. Secondly, the attacker discovers the fixed or short adjustable PIN code that is used between the target devices via an On-Line PIN Cracking attack or Off-Line PIN Recovery attack. Thirdly, the attacker uses a Bluetooth protocol analyzer to intercept the traffic of the initial pairing process between the target devices. Fourthly, the attacker intercepts all the information that is sent to the Bluetooth-enabled printer. The attacker is now able to decrypt all the intercepted information (see Section 4.1). Finally, the attacker uses a Bluetooth security analysis tool to produce the original information that was printed via Bluetooth.

We designed and implemented a new tool, the *BTPrinterBugging via Packet Interception Security Analysis Tool* [Haa07b], which was successfully used to perform BTPrinterBugging via Packet Interception attacks against four Bluetooth 1.1 compatible USB printer adapter models: Conceptronic [Con08], Mentor [Bon07], Tecom [Tec07] and Belkin [Bel08].

All these printer adapters are configured as discoverable devices by the fixed factory setting. In addition, Conceptronic and Tecom printer adapters have no support for a PIN code at all. Therefore, the default PIN code "0" is used when Bluetooth security operations are required

with them. The PIN codes for Mentor and Belkin printer adapters are "1234" and "belkin", respectively. Each of these so-called "secret PIN codes" can be easily found in the printer adapter manual, which can be freely downloaded from the Internet.

Because all four printer adapters have either default or fixed PIN codes, all Bluetooth-enabled devices that are using them via an encrypted link must also use the same PIN code − otherwise printing is not possible. Moreover, all four printer adapters are class 1 Bluetooth devices with enhanced sensitivity levels (see Section 2.3), making long-distance attacks against them possible.

In our practical experiments we used a USB printer with a Bluetooth USB printer adapter as the Bluetooth-enabled printer, a laptop running a Frontline FTS4BT [Fro08] protocol analyzer (see Chapter 5) with one Bluetooth 2.0+EDR compatible USB ComProbe as the attacking device, and a Bluetooth-enabled laptop (Bluetooth 1.2 compatible) as the legitimate Bluetooth-enabled device that was using the printer. Frontline FTS4BT v6.10.4.0 software [Fro08] was also used in our practical experiments.

Our Bluetooth security analysis tool works in the following way: [Haa07b]

1. All information that is sent to the Bluetooth-enabled printer is intercepted by using a Frontline FTS4BT protocol analyzer. If the intercepted data is encrypted, it is automatically decrypted by the protocol analyzer.

2. Intercepted raw data is exported to a comma-separated ASCII text file (see Figure 23).

3. The ASCII text file is parsed in such a way that only payload coded in hexadecimal, i.e. the actual intercepted user data, will remain, as shown in Figure 24.

4. Hexadecimal values in the parsed ASCII text file are used to produce the original printed binary file that was sent earlier to the printer by the Bluetooth-enabled device, i.e. the file produced contains the same information as the file that the user can produce, for example, using her printer's standard "print to file" feature.

5. Finally, the binary file produced is converted to a PDF (Portable Document Format) file, which is the final result of our Bluetooth security analysis tool.

```
Bookmark,Frame#,Role,Addr.,HCRP Data,Frame Size,Delta,Timestamp,
,1969,Master,2,0x 1b 25 2d 31 32 33 34 35 58 40 50 4a 4c 20 4a 4f 42 20 4e 41
4d 45 3d 22 48 69 67 68 6c 79 43 6c 61 73 73 69 66 69 65 64 44 6f 63 75 6d 65
6e 74 32 2e 70 64 66 22 0a 40 50 4a 4c 20 53 45 54 20 53 54 52 49 4e 47 43 4f
44 45 53 45 54 3d 55 54 46 38 0a 40 50 4a 4c 20 43 4f 4d 4d 45 4e 54 20 22 48
50 20 43 6f 6c 6f 72 20 4c 61 73 65 72 4a 65 74 20 34 36 35 30 20 50 43 4c 20
36 20 28 42,147, 00:00:02.845179,1.12.2006 10:15:03.249720 ,
,1970,Master,2,0x 65 6c 6b 69 6e 29 20 28 36 30 2e 33 32 2e 31 30 31 2e 34 31
29 3b 20 4d 69 63 72 6f 73 6f 66 74 20 57 69 6e 64 6f 77 73 20 58 50 20 35 2e
31 2e 32 36 30 30 2e 31 3b 20 55 6e 69 64 72 76 20 30 2e 33 2e 31 32 39 36 2e
31 22 0a 40 50 4a 4c 20 43 4f 4d 4d 45 4e 54 20 22 55 73 65 72 6e 61 6d 65 3a
20 57 4b 53 41 44 4d 49 4e 3b 20 41 70 70 20 46 69 6c 65 6e 61 6d 65 3a 20 48
69 67 68 6c,147, 00:00:00.005000,1.12.2006 10:15:03.254720 ,
,1971,Master,2,0x 79 43 6c 61 73 73 69 66 69 65 64 44 6f 63 75 6d 65 6e 74 32
2e 70 64 66 3b 20 31 32 2d 31 2d 32 30 30 36 22 0a 40 50 4a 4c 20 53 45 54 20
4a 4f 42 41 54 54 52 3d 22 4a 6f 62 41 63 63 74 31 3d 57 4b 53 41 44 4d 49 4e
22 0a 40 50 4a 4c 20 53 45 54 20 4a 4f 42 41 54 54 52 3d 22 4a 6f 62 41 63 63
74 32 3d 54 4b 54 2d 4c 54 2d 4d 35 31 33 32 22 0a 40 50 4a 4c 20 53 45 54 20
4a 4f 42 41,147, 00:00:00.005000,1.12.2006 10:15:03.259720 ,
,1972,Master,2,0x 54 54 52 3d 22 4a 6f 62 41 63 63 74 33 3d 54 4b 54 2d 4c 54
2d 4d 35 31 33 32 22 0a 40 50 4a 4c 20 53 45 54 20 4a 4f 42 41 54 54 52 3d 22
4a 6f 62 41 63 63 74 34 3d 32 30 30 36 31 32 30 31 31 30 31 34 35 32 22 0a 40
50 4a 4c 20 44 4d 49 4e 46 4f 20 41 53 43 49 49 48 45 58 3d 22 30 34 30 30 30
34 30 31 30 31 30 32 30 44 31 30 31 30 30 31 31 35 33 32 33 30 33 30 33 36 33
31 33 32 33,147, 00:00:00.006250,1.12.2006 10:15:03.265970 ,
```

Figure 23. An example of a comma-separated ASCII text file. [Haa07b]

```
1b 25 2d 31 32 33 34 35 58 40 50 4a 4c 20 4a 4f 42 20 4e 41 4d 45 3d 22 48 69 67 68 6c 79 43 6c
61 73 73 69 66 69 65 64 44 6f 63 75 6d 65 6e 74 32 2e 70 64 66 22 0a 40 50 4a 4c 20 53 45 54 20
53 54 52 49 4e 47 43 4f 44 45 53 45 54 3d 55 54 46 38 0a 40 50 4a 4c 20 43 4f 4d 4d 45 4e 54 20
22 48 50 20 43 6f 6c 6f 72 20 4c 61 73 65 72 4a 65 74 20 34 36 35 30 20 50 43 4c 20 36 20 28 42
65 6c 6b 69 6e 29 20 28 36 30 2e 33 32 2e 31 30 31 2e 34 31 29 3b 20 4d 69 63 72 6f 73 6f 66 74
20 57 69 6e 64 6f 77 73 20 58 50 20 35 2e 31 2e 32 36 30 30 2e 31 3b 20 55 6e 69 64 72 76 20 30
2e 33 2e 31 32 39 36 2e 31 22 0a 40 50 4a 4c 20 43 4f 4d 4d 45 4e 54 20 22 55 73 65 72 6e 61 6d
65 3a 20 57 4b 53 41 44 4d 49 4e 3b 20 41 70 70 20 46 69 6c 65 6e 61 6d 65 3a 20 48 69 67 68 6c
79 43 6c 61 73 73 69 66 69 65 64 44 6f 63 75 6d 65 6e 74 32 2e 70 64 66 3b 20 31 32 2d 31 2d 32
30 30 36 22 0a 40 50 4a 4c 20 53 45 54 20 4a 4f 42 41 54 54 52 3d 22 4a 6f 62 41 63 63 74 31 3d
57 4b 53 41 44 4d 49 4e 22 0a 40 50 4a 4c 20 53 45 54 20 4a 4f 42 41 54 54 52 3d 22 4a 6f 62 41
63 63 74 32 3d 54 4b 54 2d 4c 54 2d 4d 35 31 33 32 22 0a 40 50 4a 4c 20 53 45 54 20 4a 4f 42 41
54 54 52 3d 22 4a 6f 62 41 63 63 74 33 3d 54 4b 54 2d 4c 54 2d 4d 35 31 33 32 22 0a 40 50 4a 4c
20 53 45 54 20 4a 4f 42 41 54 54 52 3d 22 4a 6f 62 41 63 63 74 34 3d 32 30 30 36 31 32 30 31 31
30 31 34 35 32 22 0a 40 50 4a 4c 20 44 4d 49 4e 46 4f 20 41 53 43 49 49 48 45 58 3d 22 30 34 30
30 30 34 30 31 30 31 30 32 30 44 31 30 31 30 30 31 31 35 33 32 33 30 33 30 33 36 33 31 33 32 33
```

Figure 24. An example of a parsed ASCII text file.

Steps 2-5 of our BTPrinterBugging via Packet Interception Security Analysis Tool can be performed almost in realtime, especially when the exported comma-separated ASCII text file is quite small. Even if the size of the ASCII text file is as large as two megabytes, it takes only

on average 15 seconds on a typical Pentium laptop to produce the PDF file containing all the printed information.

The attack was successfully performed in the following way: [Haa07b]

1. We discovered the BD_ADDR of the non-discoverable legitimate Bluetooth-enabled device by using a Frontline FTS4BT protocol analyzer (the attacking device). The BD_ADDR of the Bluetooth-enabled printer was discovered in a few seconds via a general inquiry. If the Bluetooth link between the target devices is unencrypted, an attacker needs to know only the BD_ADDR of the Bluetooth-enabled printer, because there is no need to decrypt the intercepted information. If data decryption is needed, both BD_ADDRs must be known to the attacker (see Section 4.1).

2. We easily discovered the "secret PIN code" used between the target devices by using the user-friendly name and the company_id value of the Bluetooth-enabled printer in order to download the correct user manual from the Internet.

3. We used the attacking device to intercept the traffic of the initial pairing process between the target devices.

4. We used the attacking device to intercept all the information sent to the Bluetooth-enabled printer. All intercepted data was automatically decrypted when the encrypted link was used.

5. We successfully used our BTPrinterBugging via Packet Interception Security Analysis Tool to produce a PDF file, which was the result of our practical experiment.

Figure 25 illustrates the result of our practical experiment in which the legitimate user printed a "Highly Classified Document" via Bluetooth. The same experiment was successfully performed with each of the four Bluetooth printer adapters, namely Conceptronic, Mentor, Tecom and Belkin, by using first an unencrypted link and then an encrypted link, i.e. a total of eight practical experiments were successfully performed with our BTPrinterBugging via Packet Interception Security Analysis Tool.

Highly Classified Document **28.11.2006**

Secret Agent: Henry Smith
Secret Agent ID#: 007
Personal Identification Number: 221277-243L
Social Security Number: 078-55-1120

This document must be kept secret! All this information is highly classified and must not leak to the wrong hands! Be very careful and destroy this message immediately after reading!

Figure 25. The result of our BTPrinterBugging via Packet Interception attack. [Haa07b]

A BTPrinterBugging via Packet Interception attack is very dangerous, because an attacker can steal sensitive information printed via Bluetooth. In addition, because Bluetooth is a wireless RF communication system which uses mainly omnidirectional antennas, the presence of the attacker is often not seen. Moreover, the attacker with her attacking device can be very far away from the communicating devices, because most Bluetooth-enabled printers are class 1 devices with enhanced sensitivity levels.

The countermeasures for a *BTPrinterBugging via Packet Interception attack*, in addition to those described in Sections 6.1-6.5, are: [Haa07b]

- *Switching Bluetooth off completely when it is not being used or switching the printer's power off when it is not being used*: If there is no need to use the Bluetooth-enabled printer for a long time, its Bluetooth functionality can be switched off or alternatively its power can be switched off.

- *Using additional security at the application level*: If the user has a Bluetooth Access Point with a Print Server providing wireless printing services via Bluetooth, it may be possible to use application layer key exchange and encryption methods as extra security in addition to the Bluetooth built-in security.

- *Printing sensitive information via a traditional cable-based connection to the printer*: The user should not print any sensitive information via Bluetooth. Instead, the traditional cable-based connection to the printer should be used.

6.7.2 BTPrinterBugging via Impersonation Security Analysis Tool

If an attacker wants to use the Bluetooth-enabled printer remotely as if it was her own, she needs to know the BD_ADDR of the printer and that of one legitimate piconet device that is using the printer. In addition, the secret PIN code that is used between these two target devices must be known to the attacker. As described in Subsection 6.7.1, in most cases the attacker can discover this secret PIN code quite fast. Of course, the attacker must also intercept the traffic of the initial pairing process between these two devices when they meet for the first time, but as explained in Subsection 4.2.1 there are many different ways to arrange or force the target devices to repeat the initial pairing process in order to intercept the traffic of the initial pairing process.

This kind of attack is normally possible only if the target devices are configured as discoverable devices. However, there are also ways to find non-discoverable devices (see Subsection 6.7.1). Moreover, the attack requires that an attacker has a Bluetooth device with an adjustable BD_ADDR, i.e. the attacking device must be capable of duplicating the

BD_ADDR of the legitimate device that is using the printer. Some commercially available Bluetooth protocol analyzers, such as LeCroy BTTracer/Trainer [Lec07] (see Chapter 5), support the BD_ADDR duplication feature, so this is not a problem for the attacker.

Let us assume the following attack scenario. First, an attacker uses a Brute-Force BD_ADDR Scanning attack or a Bluetooth protocol analyzer to discover the BD_ADDRs of two target devices. Secondly, the attacker discovers the fixed or short adjustable PIN code that is used between the target devices via an On-Line PIN Cracking attack or Off-Line PIN Recovery attack. Thirdly, she uses a Bluetooth protocol analyzer to intercept the traffic of the initial pairing process between the target devices. Fourthly, the attacking device impersonates the legitimate piconet device by duplicating its BD_ADDR value. Fifthly, the attacking device authenticates itself with the printer by using the traffic of the initial pairing process that was intercepted in step 3. Finally, the attacker is capable of using the printer remotely as if it was her own.

Now the attacker can abuse the printer in any way: for example, she can print funny or lubricious pictures in order to make fun of the printer's owner, print hundreds of pages of random text or colorful pictures in order to waste both printing paper and powder/ink, or print hoax documents or modified "real documents" in order to mislead the legitimate users of the printer. In fact, the printer is often the best and most used source of information in a company.

Our new Bluetooth security analysis tool, the *BTPrinterBugging via Impersonation Security Analysis Tool* [Haa07b], was successfully used to perform BTPrinterBugging via Impersonation attacks [Haa07b] against four Bluetooth USB printer adapter models: Conceptronic, Mentor, Tecom and Belkin (see Subsection 6.7.1).

In our practical experiments we used a USB printer with a Bluetooth USB printer adapter as the Bluetooth-enabled printer, a laptop connected to the LeCroy BTTracer/Trainer protocol analyzer [Lec07] with one Bluetooth 1.1 compatible radio unit as the attacking device, and a Bluetooth-enabled laptop (Bluetooth 1.2 compatible) as the legitimate Bluetooth-enabled device that was using the printer. LeCroy BTTracer/Trainer v2.2 software [Lec07], which

provides CATC Scripting Language [Lec04] (see Chapter 5), was also used in our practical experiments.

Our Bluetooth security analysis tool works in the following way: [Haa07b]

1. We discovered the BD_ADDR of the non-discoverable legitimate Bluetooth-enabled device by using a LeCroy BTTracer/Trainer protocol analyzer (the attacking device). The BD_ADDR of the Bluetooth-enabled printer was discovered in a few seconds via the general inquiry.

2. We easily discovered the "secret PIN code" by using the user-friendly name and the company_id value of the Bluetooth-enabled printer (see Subsection 6.7.1).

3. We used the attacking device to intercept the traffic of the initial pairing process between the target devices.

4. We used the attacking device to impersonate the legitimate piconet device by duplicating its BD_ADDR value.

5. We used the attacking device to authenticate itself with the printer by using the traffic of the initial pairing process that was intercepted in step 3.

6. The attacking device abused the printer by printing funny pictures, dozens of pages of random text, and various hoax documents (see Figure 26).

```
(1)  Discovering the BD_ADDRs of Bluetooth Devices in Range..
(2)  The following BD_ADDRs were discovered: 000B5DA45D8C, 00027242A323
(3)  HCI_Evt> Remote_Name_Request_Complete
(4)     BD_ADDR                 : 000B5DA45D8C
(5)     Name                    : "TKT-LT-M5132"
(6)  User Friendly Name of Device 00 is: TKT-LT-M5132
(7)  HCI_Evt> Remote_Name_Request_Complete
(8)     BD_ADDR                 : 00027242A323
(9)     Name                    : "BELKIN_PRT_42A323"
(10) User Friendly Name of Device 01 is: BELKIN_PRT_42A323
(11) BD_ADDR of the Legitimate Bluetooth Device is: 000B5DA45D8C
(12) BD_ADDR of the Printer is: 00027242A323
(13) PIN Code of the Printer is: belkin
(14) HCI_Evt> Write_Authentication_Enable_Complete
(15) HCI_Evt> Write_Encryption_Mode_Complete
(16) TCI_Evt> CATC_SetBdAddr_Complete
(17)    BD_ADDR                 : 000B5DA45D8C
(18) BD_ADDR of the Legitimate Bluetooth Device Is Now Successfully Duplicated!
(19) HCI_Evt> PIN_Code_Request
(20)    PIN reply               : belkin
(21) HCI_Evt> Pairing_Complete
(22)    BD_ADDR                 : 00027242A323
(23) HCI_Evt> Connection_Complete
(24)    BD_ADDR                 : 00027242A323
(25)    HCI Handle              : 0x0002
(26) L2CAP Channel Successfully Established!
(27) Printing Funny Pictures, Random Text and Hoax Documents..
(28) All Printing Jobs Were Successfully Completed!
(29) L2CAP Channel Successfully Disconnected!
(30) HCI_Evt> Disconnection_Complete
(31)    BD_ADDR                 : 00027242A323
(32)    Reason                  : No Connection
```

Figure 26. The result of our BTPrinterBugging via Impersonation attack. [Haa07b]

As Figure 26 illustrates, the attacking device first discovers the BD_ADDRs of the victim devices (see rows 1-2 and 11-12), the user-friendly names of the victim devices (see rows 3-10), and the "secret PIN code" of the printer (see row 13). The attacking device is set to require authentication and encryption for each connection with the printer (see rows 14-15). The legitimate piconet device is impersonated by duplicating its BD_ADDR value (see rows 16-18). After the successful authentication with the printer (see rows 19-25), the attacking device abuses it by printing funny pictures, dozens of pages of random text, and various hoax

documents (see rows 26-28). Finally, the attacking device disconnects from the printer (see rows 29-32).

The same experiment was successfully performed with each of our four Bluetooth printer adapters by using first an unencrypted link and then an encrypted link, i.e. eight practical experiments were successfully performed with our BTPrinterBugging via Impersonation Security Analysis Tool.

A BTPrinterBugging via Impersonation attack is typically not very dangerous, because an attacker does not steal any information from the target devices. However, this kind of attack can be very annoying if the attacker uses it to do various harmful things, as described above.

One countermeasure for a *BTPrinterBugging via Impersonation attack*, provided that all countermeasures from Sections 6.1-6.5 and from Subsection 6.7.1 are in place, is: [Haa07b]

- *Using an additional Bluetooth-independent reauthentication always prior to accepting the connection establishment to the printer*: If a user has a Bluetooth Access Point with Print Server, it may be possible to configure the Bluetooth Access Point in such a way that it always requires Bluetooth-independent reauthentication prior to accepting the connection establishment to the printer.

6.7.3 BTPrinterBugging via Access Denial Security Analysis Tools

A BTPrinterBugging via Access Denial attack [Haa07b] extends a BTPrinterBugging via Impersonation attack (see Subsection 6.7.2). Therefore, the prerequisites for these two attacks are the same.

Let us assume the following attack scenario. Steps 1-5 in this attack are the same as in a BTPrinterBugging via Impersonation attack (see Subsection 6.7.2). Finally (step 6), the attacker is able to deny the legitimate piconet users access to the printer by making repeated successful connection establishments to the printer, i.e. the attacker makes sure that the printer is always busy and therefore unable to service other devices.

Our new Bluetooth security analysis tool, the *BTPrinterBugging via Access Denial Security Analysis Tool* [Haa07b], was successfully used to perform BTPrinterBugging via Access Denial attacks against four Bluetooth USB printer adapter models: Conceptronic, Mentor, Tecom and Belkin (see Subsections 6.7.1-6.7.2). All other hardware and software used in this practical experiment were the same as in the BTPrinterBugging via Impersonation attack described in Subsection 6.7.2.

Our Bluetooth security analysis tool works in the following way. Steps 1-5 are the same as in a BTPrinterBugging via Impersonation attack (see Section 6.7.2). In step 6, the attacking device successfully established connection with the printer, waited until the printer automatically performed the disconnection, and made similar repeated successful connection establishments. In this way, the attacking device denied the legitimate piconet devices access to the printer.

The same experiment was successfully performed with each of our four Bluetooth printer adapters by using first an unencrypted link and then an encrypted link, i.e. eight practical experiments were successfully performed with our BTPrinterBugging via Access Denial Security Analysis Tool.

Some Bluetooth-enabled printers never perform automatic disconnection, so an attacker can establish basic ACL links with them which are never disconnected. In this way, the attacker denies the legitimate piconet devices access to the printers. Some Bluetooth-enabled printers also support multiple connections at the same time. In this case, the attacker can simply use several Bluetooth radio units to open multiple ACL links with the printer adapter. Another possibility is to use one Bluetooth radio unit, but open multiple L2CAP channels with the printer in order to reserve all resources for the attacker.

We also implemented another new Bluetooth security analysis tool, *BTPrinterBugging via Access Denial Security Analysis Tool II* [Haa07b], which works in Linux environments. It requires a BlueZ protocol stack [Blu08b] and at least one Bluetooth USB dongle to work, i.e. an expensive Bluetooth protocol analyzer is not required. This security analysis tool does not require any tricks that only sophisticated protocol analyzers can do. The only required

information is the BD_ADDR of the Bluetooth-enabled printer, i.e. an attacker does not even have to know the secret PIN code of the Bluetooth-enabled printer.

Our BTPrinterBugging via Access Denial Security Analysis Tool II is capable of keeping Bluetooth-enabled printers busy all the time by making repeated connection attempts to them. In this way, the attacking device very easily denies the legitimate piconet devices access to all Bluetooth-enabled printers. Our security analysis tool supports the automatic discovery and recognition of Bluetooth-enabled printers in range. Another possibility is to select Bluetooth-enabled printers manually from the list of discovered devices. If several Bluetooth-enabled printers are detected, several Bluetooth USB dongles will also automatically be used in parallel, i.e. one Bluetooth USB dongle is used for each detected Bluetooth-enabled printer. This security analysis tool can also be used to perform DoS attacks against any other kinds of Bluetooth devices, because the only information required is the BD_ADDR of the target device.

Figure 27 illustrates our BTPrinterBugging via Access Denial Security Analysis Tool II in action (see rows 1-34). When the attacker runs the BTPrinterBugging via Access Denial Security Analysis Tool II in Linux (see row 1), she can either manually select Bluetooth-enabled printers from the list of the discovered devices (see rows 2-3) or alternatively use the automatic discovery of Bluetooth-enabled printers (see rows 4-5). In this practical experiment, the attacker chooses the automatic discovery option (see rows 6-15). Because four Bluetooth-enabled printers are detected, four Bluetooth USB dongles are used in parallel (see row 16). Finally, the attacking device simultaneously denies all legitimate printer users access to all four Bluetooth-enabled printers (see rows 17-34).

```
(1)  # ./BTPrinterBuggingViaAccessDenial2
(2)  1. Select Bluetooth-enabled Printers Manually From the List of
(3)     Discovered Devices.
(4)  2. Automatically Both Discover And Decide What Devices Are
(5)     Bluetooth-enabled Printers.
(6)  2
(7)  Scanning ...
(8)          00:02:72:42:A3:23       BELKIN_PRT_42A323
(9)          00:04:61:10:05:A0       DESKTOP_PC-FC6
(10)         00:04:61:83:9A:26       Printer Adapter 839A26
(11)         00:0B:0D:10:29:A1       BT printer
(12)         00:0B:5D:A4:5D:8C       TKT-LT-M5132
(13)         00:03:C9:3C:33:32       Sitecom CN-505
(14) Bluetooth Devices Found: 6
(15) Bluetooth-enabled Printers Detected: 4
(16) Using 4 Bluetooth USB Dongles in Parallel.
(17) Simultaneously Denying Access From All Legitimate Printer Users
(18) to All 4 Bluetooth-enabled Printers:
(19)  Connecting to Printer Adapter #1 (00:02:72:42:A3:23) (attempt #1)
(20)  Connecting to Printer Adapter #2 (00:04:61:83:9A:26) (attempt #1)
(21)  Connecting to Printer Adapter #3 (00:0B:0D:10:29:A1) (attempt #1)
(22)  Connecting to Printer Adapter #4 (00:03:C9:3C:33:32) (attempt #1)
(23)   Can't Create Connection to Printer Adapter #1: Connection Timed Out
(24)   Can't Create Connection to Printer Adapter #2: Connection Timed Out
(25)   Can't Create Connection to Printer Adapter #3: Connection Timed Out
(26)   Can't Create Connection to Printer Adapter #4: Connection Timed Out
(27)  Connecting to Printer Adapter #1 (00:02:72:42:A3:23) (attempt #2)
(28)  Connecting to Printer Adapter #2 (00:04:61:83:9A:26) (attempt #2)
(29)  Connecting to Printer Adapter #3 (00:0B:0D:10:29:A1) (attempt #2)
(30)  Connecting to Printer Adapter #4 (00:03:C9:3C:33:32) (attempt #2)
(31)   Can't Create Connection to Printer Adapter #1: Connection Timed Out
(32)   Can't Create Connection to Printer Adapter #2: Connection Timed Out
(33)   Can't Create Connection to Printer Adapter #3: Connection Timed Out
(34)   Can't Create Connection to Printer Adapter #4: Connection Timed Out
```

Figure 27. The BTPrinterBugging via Access Denial
Security Analysis Tool II in action. [Haa07b]

Some Bluetooth-enabled printers allow users to create basic ACL links without authentication, i.e. no PIN code is required prior to accepting the connection establishment between the Bluetooth-enabled printer and a remote Bluetooth-enabled device. Therefore, an attacker can very easily reserve all the resources of such printers by using several Bluetooth USB dongles to establish multiple ACL links.

BTPrinterBugging via Access Denial attacks are not very dangerous, because an attacker does not steal any information from the target devices. However, these kinds of attacks can be very annoying if the attacker uses them non-stop in order to permanently deny the legitimate piconet users access to the printers.

The countermeasures for *BTPrinterBugging via Access Denial attacks*, in addition to those described in Sections 6.1-6.5 and in Subsections 6.7.1-6.7.2, are: [Haa07b]

- *Monitoring Bluetooth communication in order to prevent and stop attacks in progress*: Various Bluetooth security attacks in progress can be prevented and stopped by monitoring Bluetooth communication to discover such attacks (see Section 6.12). It is possible to detect various attacks in progress by noting any strange communication behaviour in Bluetooth devices in range, such as unusually many repeated successful connection establishments or failed authentication attempts, a sudden increase in transmit powers, and unusually many NAK transmissions.

- *Using a portable Bluetooth Direction-Finding device*: It is possible to use a portable Bluetooth Direction-Finding device to determine the area where the attacking device performs repeated successful connection establisments or some other harmful things. Then the attacking device can be physically sought and switched off or even destroyed.

6.8 BD_ADDR Duplication Security Analysis Tool

The main features of a *BD_ADDR Duplication attack* were described in Subsection 4.2.3. The most effective way to perform this attack is to duplicate the BD_ADDR of the piconet master, because all information within the piconet goes through the master device. When the BD_ADDR of the piconet master is duplicated by the bug, all connections within that piconet will be effectively jammed due to simultaneous responses of both the target device and the bug.

A BD_ADDR Duplication attack is normally possible only if the target device is configured as discoverable (see Section 4.1). However, as we explained in Sections 6.1 and 6.4, there are also ways to find non-discoverable devices. A BD_ADDR Duplication attack also requires that an attacker has a Bluetooth device with an adjustable BD_ADDR, because the bug must be capable of duplicating the BD_ADDR of the target device. Some commercially available Bluetooth protocol analyzers, such as LeCroy BTTracer/Trainer [Lec07] (see Chapter 5), support the BD_ADDR duplication feature. Therefore, it is not a problem for the attacker. Moreover, a BD_ADDR Duplication attack requires that the bug must be capable of impersonating the piconet master in order to respond to connection attempts of legitimate Bluetooth devices.

Let us assume the following attack scenario. First, an attacker uses a Brute-Force BD_ADDR Scanning attack or a Bluetooth protocol analyzer to discover the BD_ADDR of the piconet master. Secondly, the bug duplicates the BD_ADDR of the piconet master. Finally, the bug impersonates the piconet master in order to respond to any connection attempts from the legitimate piconet devices. Now, the bug is capable of denying the legitimate devices access via a BD_ADDR Duplication attack.

In our practical experiments we used an unmodified Bluetooth 1.1 compatible Nokia 6310i mobile phone [Nok02] as the piconet master, a laptop connected to the LeCroy BTTracer/Trainer protocol analyzer [Lec07] with one Bluetooth 1.1 compatible radio unit as the bug, and three Bluetooth headsets (Nokia's Bluetooth 1.1 compatible HDW-2 [Nok03], Nokia's Bluetooth 2.0+EDR compatible HS-26W [Nok05], and Sony Ericsson's Bluetooth 2.0+EDR compatible HBH-610 [Son05]) and Epox's Bluetooth 2.0+EDR compatible USB dongle BT-DG07A+ [Epo05] as the legitimate piconet devices. LeCroy BTTracer/Trainer v2.2 software [Lec07], which provides CATC Scripting Language [Lec04] (see Chapter 5), was also used.

CATC Scripting Language was used to implement our new Bluetooth security analysis tool, *BD_ADDR Duplication Security Analysis Tool* [Haa07a], which was successfully used to perform BD_ADDR Duplication attacks.

Our Bluetooth security analysis tool works in the following way: [Haa07a]

1. We discovered the BD_ADDR of the non-discoverable target mobile phone (the piconet master) by using a LeCroy BTTracer/Trainer protocol analyzer (the bug).

2. We used the bug to duplicate the BD_ADDR of the piconet master.

3. We used the bug to impersonate the piconet master. When any legitimate piconet device (the headsets or the USB dongle) tried to communicate with the piconet master, the bug simultaneously responded each time with the piconet master and therefore together they denied the legitimate piconet devices access by jamming each other.

Figure 28 (see rows 1-30) illustrates our practical experiment in which the bug successfully denies the legitimate piconet devices access.

```
(1)  The BD_ADDR of the Piconet Master Is: 0002EEB0294D
(2)  HCI_Evt> Remote_Name_Request_Complete
(3)    BD_ADDR : 0002EEB0294D
(4)    Name    : "Nokia 6310i"
(5)  TCI_Evt> CATC_SetBdAddr_Complete
(6)    BD_ADDR               : 0002EEB0294D
(7)  The Bug Has Successfully Duplicated the BD_ADDR of the Piconet Master!
(8)  HCI_Evt> CATC_Write_PIN_Response_Enable_Complete
(9)  OBEX_Evt> ServerDeinit_Complete
(10)   Status                : Success
(11) SDP_Evt> AddProfileServiceRecord_Complete
(12)   Profile               : Headset Audio Gateway
(13) HCI_Evt> Write_Authentication_Enable_Complete
(14) HCI_Evt> Write_Encryption_Mode_Complete
(15) The Bug Has Successfully Impersonated the Piconet Master!
(16) The Bug Is Waiting for a Connection..
(17) HCI_Evt> Connection_Request
(18)   Incoming connection accepted
(19) HCI_Evt> PIN_Code_Request
(20)   PIN reply             : 0000
(21) HCI_Evt> Connection_Error
(22)   Incoming connection failed, reason : Authentication Failure
(23) The Bug Is Waiting for a Connection..
(24) HCI_Evt> Connection_Request
(25)   Incoming connection accepted
(26) HCI_Evt> PIN_Code_Request
(27)   PIN reply             : 0000
(28) HCI_Evt> Connection_Error
(29)   Incoming connection failed, reason : Authentication Failure
(30) The Bug Is Waiting for a Connection..
```

Figure 28. The result of our BD_ADDR Duplication attack. [Haa07a]

The bug first discovers the BD_ADDR value (see row 1) and the user-friendly name of the victim device (see rows 2-4). It impersonates the victim device by duplicating its BD_ADDR value (see rows 5-7), setting itself to require both authentication (see rows 8 and 13) and encryption (see row 14), and emulating the Headset Audio Gateway Profile that is supported by the victim device (see rows 9-12 and 15). Finally, the bug waits for connections from the legitimate piconet devices (see rows 16, 23 and 30) and every time a connection request is received, the bug performs the authentication by using an incorrect PIN code, so all authentication attempts will fail (see rows 17-22 and 24-29).

A BD_ADDR Duplication attack is typically not very dangerous, because an attacker does not steal any information from the target devices. However, this kind of attack can be dangerous if the attacker uses it to mislead the target devices in such a way that they delete previously stored link keys so that the initial pairing process is restarted.

The countermeasures for a *BD_ADDR Duplication attack* seem to be the same as those described in Sections 6.1-6.7.

6.9 SCO/eSCO Security Analysis Tool

The main features of a *SCO/eSCO attack* were described in Subsection 4.2.3. The most effective way to perform this attack is to establish a SCO or an eSCO link with the piconet master, because all information within the piconet goes through the master device.

A SCO/eSCO attack is possible when the target device has a fixed or short adjustable PIN code, its BD_ADDR is known to an attacker, and it has support for SCO or eSCO links (see Sections 2.2 and 2.4). The secret PIN code of the target device can be discovered via an On-Line PIN Cracking attack (see Subsection 4.2.4 and Section 6.3) or Off-Line PIN Recovery attack (see Subsection 4.2.1).

A SCO/eSCO attack is normally possible only if the target device is configured as discoverable (see Section 4.1). However, as we already discussed in Sections 6.1 and 6.4, there are also ways to find non-discoverable devices. Moreover, a SCO/eSCO attack requires that an attacker intercepts the traffic of the initial pairing process between two Bluetooth

devices when they meet for the first time (see Section 4.1). As explained in Subsection 4.2.1, there are many different ways to arrange or force the target devices to repeat the initial pairing process, so this is not a big problem for the attacker.

Let us assume the following attack scenario. First, an attacker uses a Brute-Force BD_ADDR Scanning attack or a Bluetooth protocol analyzer to discover the BD_ADDRs of the legitimate piconet devices and the piconet master. Secondly, the attacker discovers the fixed or short adjustable PIN codes of the legitimate piconet devices via an On-Line PIN Cracking attack or Off-Line PIN Recovery attack. Thirdly, the attacker uses a Bluetooth protocol analyzer to intercept the traffic of the initial pairing process between the piconet master and one legitimate piconet device. Fourthly, the attacker duplicates the BD_ADDR of the legitimate piconet device by using a Bluetooth protocol analyzer. Fifthly, the attacker authenticates itself with the piconet master by using the traffic of the initial pairing process that was intercepted in step 3. Finally, the attacker opens a two-way realtime SCO or eSCO link with the piconet master. The most effective way to perform this attack is to establish a SCO link that uses HV1 voice packets, because in that way all the piconet master's attention is reserved for the attacker and the legitimate piconet devices do not get service within a reasonable time.

In our practical experiments we used an unmodified Bluetooth 1.1 compatible Nokia 6310i mobile phone [Nok02] as the piconet master, a laptop connected to the LeCroy BTTracer/Trainer protocol analyzer [Lec07] (see Chapter 5) with one Bluetooth 1.1 compatible radio unit as the attacking device, and three Bluetooth headsets (Nokia's HDW-2 [Nok03], Nokia's HS-26W [Nok05], and SonyEricsson's HBH-610 [Son05]) as the legitimate piconet devices. LeCroy BTTracer/Trainer v2.2 software [Lec07], which provides CATC Scripting Language [Lec04] (see Chapter 5), was also used.

CATC Scripting Language was used to implement our new Bluetooth security analysis tool, *SCO/eSCO Security Analysis Tool* [Haa07a], which was successfully used to perform SCO attacks.

Our Bluetooth security analysis tool works in the following way: [Haa07a]

1. We discovered the BD_ADDRs of the headsets (the legitimate piconet devices) and the target mobile phone (the piconet master) by using a LeCroy BTTracer/Trainer protocol analyzer (the attacking device).

2. We discovered the fixed PIN codes of the headsets by using our On-Line PIN Cracking Security Analysis Tool (see Section 6.3).

3. We used the attacking device to intercept the traffic of the initial pairing process (see Section 4.1) between the piconet master and one legitimate piconet device (a headset).

4. We used the attacking device to duplicate the BD_ADDR of the legitimate piconet device.

5. We used the attacking device to authenticate itself with the piconet master by using the traffic of the initial pairing process that was intercepted in step 3.

6. We used the attacking device to open a two-way realtime HV1 SCO link with the piconet master. In this way, the legitimate piconet devices do not get service within a reasonable time.

The same experiment was successfully performed with each of our Bluetooth headset, i.e. three practical experiments were successfully performed with our SCO/eSCO Security Analysis Tool. A SCO/eSCO attack is typically not very dangerous, because an attacker does not steal any information from the target devices. However, this kind of attack can be very annoying if the attacker uses it non-stop to permanently deny the legitimate piconet devices access to the piconet services.

The countermeasures for a *SCO/eSCO attack* seem to be the same as those described in Sections 6.1-6.7.

6.10 Big NAK Security Analysis Tool

The main features of a *Big NAK attack* were described in Subsection 4.2.3. The most effective way to perform this attack is to request information from the piconet master, because all information within the piconet goes through the master device.

A Big NAK attack is possible if the target device is configured to respond to every information request, or if the attacking device is capable of impersonating one legitimate piconet device. In addition, the BD_ADDR of the target device (typically a piconet master) must be known to an attacker. As described in Sections 6.1 and 6.4, there are many ways to find non-discoverable devices. Moreover, if the attack requires the impersonation of the legitimate piconet device, the secret PIN code that is used between the piconet master and the legitimate piconet device must be known to the attacker, and the attacker must also intercept the traffic of the initial pairing process between these two target devices when they meet for the first time (see Section 4.1). The PIN code of the target device can be discovered via an On-Line PIN Cracking attack as described in Subsection 4.2.4 and in Section 6.3, or an Off-Line PIN Recovery attack as explained in Subsection 4.2.1. As explained in Subsection 4.2.1, there are also many ways to arrange or force the target devices to repeat the initial pairing process.

Let us assume the following attack scenario. First, an attacker uses a Brute-Force BD_ADDR Scanning attack or a Bluetooth protocol analyzer to discover the BD_ADDRs of one legitimate piconet device and the piconet master. Secondly, she discovers the PIN code that is used between these two legitimate piconet devices via an On-Line PIN Cracking attack or Off-Line PIN Recovery attack. Thirdly, she uses a Bluetooth protocol analyzer to intercept the traffic of the initial pairing process between these two legitimate piconet devices. Fourthly, the attacker duplicates the BD_ADDR of the legitimate piconet device by using a Bluetooth protocol analyzer. Fifthly, she authenticates herself with the piconet master by using the traffic of the initial pairing process that was intercepted in step 3. Finally, the attacker requests information from the piconet master and every time the requested information is received, the attacker sends NAK. In this way, the attacker puts the piconet

master on an endless retransmission loop so that the legitimate piconet devices do not get service within a reasonable time or at least they have considerably slowed throughput.

In our practical experiments we used an unmodified Bluetooth 1.1 compatible Nokia 6310i mobile phone [Nok02] as the piconet master, a laptop connected to the LeCroy BTTracer/ Trainer protocol analyzer [Lec07] (see Chapter 5) with one Bluetooth 1.1 compatible radio unit as the attacking device, and Epox's Bluetooth 2.0+EDR compatible USB dongle BT-DG07A+ [Epo05] as the legitimate piconet device. LeCroy BTTracer/Trainer v2.2 software [Lec07], which provides CATC Scripting Language [Lec04] (see Chapter 5), was also used.

We implemented the new Bluetooth security analysis tool, *Big NAK Security Analysis Tool* [Haa07a], using CATC Scripting Language. The tool was successfully used to perform Big NAK attacks.

Our tool works in the following way: [Haa07a]

1. We discovered the BD_ADDRs of the Bluetooth USB dongle (the legitimate piconet device) and the target mobile phone (the piconet master) by using a LeCroy BTTracer/Trainer protocol analyzer (the attacking device).

2. We discovered the PIN code used between the piconet master and the legitimate piconet device by using our On-Line PIN Cracking Security Analysis Tool (see Section 6.3).

3. We used the attacking device to intercept the traffic of the initial pairing process between the piconet master and the legitimate piconet device.

4. We used the attacking device to duplicate the BD_ADDR of the legitimate piconet device.

5. We used the attacking device to authenticate itself with the piconet master by using the traffic of the initial pairing process that was intercepted in step 3.

6. We used the attacking device to request information from the piconet master and every time the requested information was received, the attacking device sent NAK. In this way, the attacking device put the piconet master on an endless retransmission loop.

A Big NAK attack is typically not very dangerous, because an attacker does not steal any information from the target devices. However, this kind of attack can be very annoying if the attacker uses it non-stop to deny the legitimate piconet devices access to the piconet services, or at least in such a way that they have considerably slowed throughput.

The countermeasures for a *Big NAK attack* seem to be the same as those described in Sections 6.1-6.7.

6.11 MITM Attacks on Bluetooth

Our two MITM attacks on Bluetooth SSP, *BT-Niño-MITM attack* [HyH07] and *BT-SSP-Printer-MITM attack* [HaH08], are described in Subsections 6.11.1-6.11.2. Subsection 6.11.3 provides a comparative analysis of the existing MITM attacks on Bluetooth [HaH08] including our two MITM attacks on Bluetooth SSP.

6.11.1 BT-Niño-MITM attack

We call our new attack a *BT-Niño-MITM attack* [HyH07] (also referred to as *Bluetooth - No Input, No Output - Man-In-The-Middle attack*). In the attack we exploit the fact that the devices must exchange information about their IO capabilities during the first phase of the SSP (See Section 4.1). The exchange is done over an unauthenticated channel, and an attacker that controls this channel can therefore modify the information about capabilities and force the devices to use the association model of her choice. In our attack, the devices are forced to use the Just Works association model, which does not provide protection against MITM attacks.

The MITM uses two separate Bluetooth devices with adjustable BD_ADDRs for the attack. Such devices are readily available on the market. The MITM clones the BD_ADDRs and

user-friendly names of the victim devices, in order to impersonate them more plausibly. The main features of the attack are depicted in Figure 29. We next describe three scenarios for the attack.

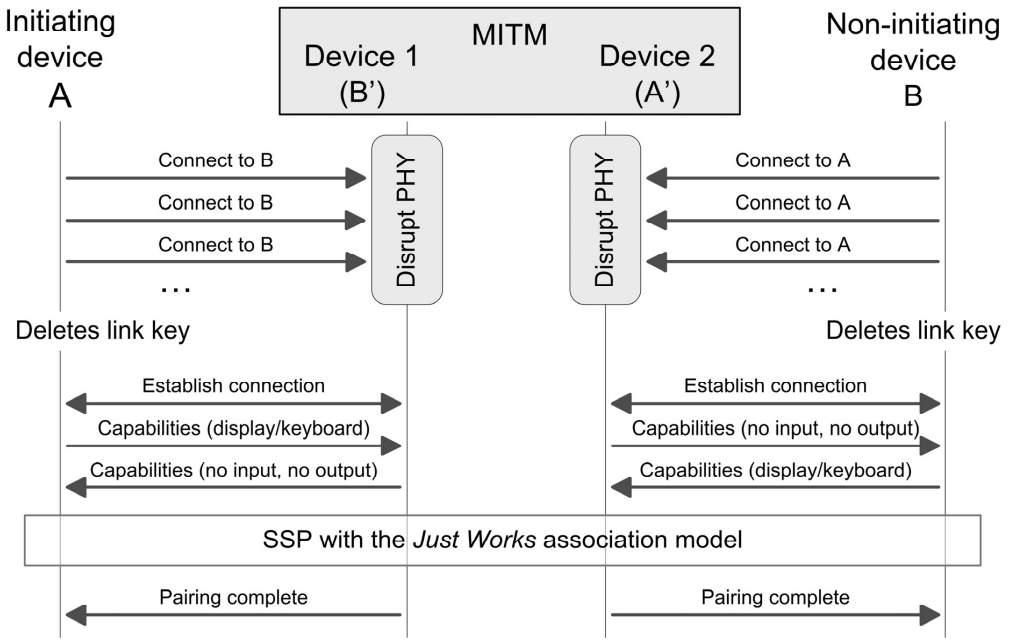

Figure 29. The main features of a BT-Niño-MITM attack. [HyH07]

In the first scenario, the MITM first disrupts (jams) the PHY by hopping along with the victim devices and sending random data in every timeslot. Another possibility is to jam the entire 2.4 GHz band altogether by using a wideband signal. In this way, the MITM shuts down all piconets within the range of susceptibility and there is no need to use a Bluetooth chipset to generate hopping patterns. Finally, the frustrated user thinks that something is wrong with her Bluetooth devices and deletes previously stored link keys. Then the user initiates a new pairing process by using SSP, and the MITM can forge messages exchanged during the IO capabilities exchange phase. When the Just Works association model has been forced into use, the attack continues as illustrated in Figure 30. Most of the notations used in Figure 30 have been explained above (see Table 4 in Section 4.1) and the rest are self-explanatory.

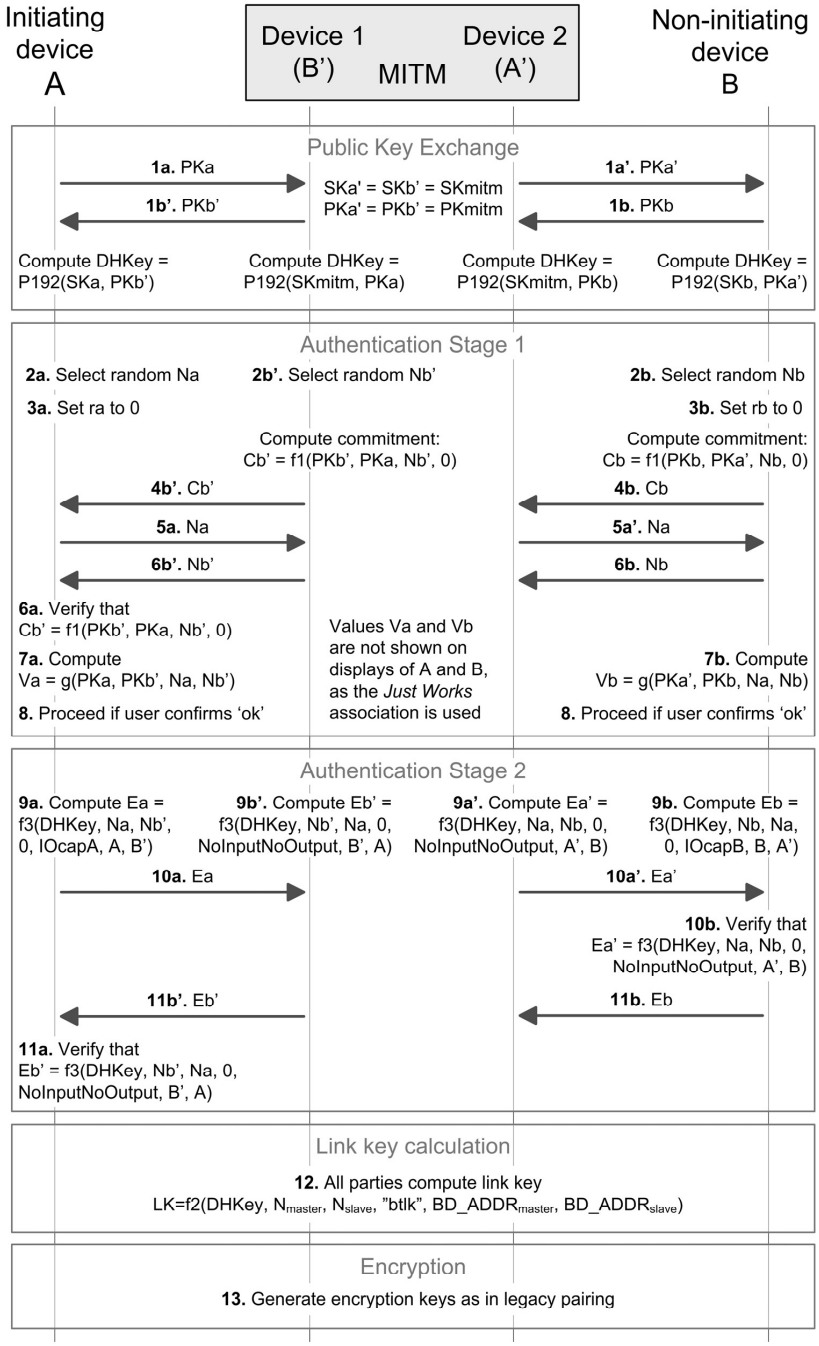

Figure 30. Pairing details of a BT-Niño-MITM attack. [HyH07]

It is worth noting that in this first scenario two victim devices have already performed the initial pairing, including the capabilities exchange, so link keys are saved on the devices for use in subsequent connections, i.e. the victim devices normally use SSP without capabilities exchange (see Section 4.1).

Other scenarios, where victim devices have never met before, are easier for the MITM, because in those cases the first phase of the attack (disrupting the PHY) can be skipped. There can be two different scenarios for such devices: [HyH07]

1. *The victim device (A or B) initiates SSP*: In this scenario, the MITM waits until A or B initiates SSP. Then the attack proceeds as illustrated in Figures 29 and 30.

2. *The MITM (A' and B') initiates SSP*: In this scenario, the MITM first initiates SSP with the victim devices. Then the attack proceeds as illustrated in Figures 29 and 30. Depending on the implementation of the victim devices, it may be possible to perform SSP without asking the user to accept the connection.

Depending on the situation, the MITM can use any of these three attack scenarios. The applicability of a certain attack scenario obviously depends on the implementation of victim devices. After a successful attack, the MITM can intercept and modify all data exchanged between the victim devices, and even use certain services that victim devices offer.

Suomalainen et al. [SVA07] have performed a comparative analysis of Bluetooth SSP, Wi-Fi (Wireless Fidelity) Protected Setup, Wireless USB Association Models, and HomePlugAV security modes. They present an attack against SSP similar to our BT-Niño-MITM Attack. In their attack the MITM prompts one device to use the normal Numeric Comparison association model, while forcing the other device to use the insecure Just Works association model. This leads to one of the devices (the one which uses the Numeric Comparison association model) treating the resulting link key as authenticated, and it might choose to trust it even for an application which requires a high level of security. However, this attack looks somewhat suspicious from the point of view of the user: one of the devices asks the user to compare the integrity checksums, while the other device does not display any numbers.

In the tests performed by Suomalainen et al. [SVA07], only 6 users out of 40 accepted the pairing on both devices. Compared with the attack of Suomalainen et al. [SVA07], our BT-Niño-MITM attack looks less dubious: indeed, the user is only asked to confirm the pairing on both devices by pressing a button. In addition, according to the Bluetooth specification [Blu07a], even this confirmation request is optional, meaning that some of the manufacturers might choose to skip it, to improve usability. Moreover, as the MITM in our attack uses two Bluetooth devices with BD_ADDRs and user-friendly names equal to those of the victim devices, the user gets even more confident that the pairing is proceeding correctly and securely. It is also worth noting that by using two MITM devices, SSP can be performed at the same time with both victim devices and it also ends at the same time with both victim devices, thus making the user even more confident.

Because it is difficult to combine high levels of security with good usability, other studies of SSP have concentrated mostly on analyzing and improving its usability. Uzun et al. [UKA07] have analyzed different ways of prompting the user to perform the comparison of integrity checksums or to enter passkeys. They have also provided guidelines for designing the user interface, to decrease the number of fatal errors and thus improve both the usability and security of SSP.

It is difficult to create a protocol which caters for all possible types of wireless devices, as the security of the protocol is likely to be limited by the capabilities of the least powerful or the least secure device type. Our BT-Niño-MITM attack is based exactly on this problem.

A countermeasure for *BT-Niño-MITM attack*, provided that all countermeasures from Sections 6.1-6.7 are in place, is: [HaH08, HyH07]

- *An additional window at the user interface level*: The attack can be prevented on the user interface level. We strongly recommend that an additional window, "The second device has no display and keyboard! Is this true?", should be displayed at the user interface level of SSP when the Just Works association model is to be used. The user is asked to choose either "Proceed" or "STOP". In practice, future Bluetooth specifications should strongly recommend Bluetooth device/software manufacturers to

implement this new window as a security improvement of SSP. The advantage of this approach is that the Just Works association model can still be a part of the future Bluetooth SSP specifications without any changes. On the other hand, this countermeasure is a clear trade-off between security and usability.

6.11.2 BT-SSP-Printer-MITM attack

We call our new attack a *BT-SSP-Printer-MITM attack* [HaH08] (also referred to as a *Bluetooth - Secure Simple Pairing - Printer - Man-In-The-Middle attack*). In this attack we exploit the fact that almost all Bluetooth-enabled printers that support SSP (especially those connected using Bluetooth USB printer adapters) will use the Just Works association model in order to make printing user-friendly. It is not likely that users will be required to press any printer buttons just to accept the connection establishment in the initial pairing process of SSP. Therefore, the Just Works association model seems to be the most logical choice for SSP-enabled printers. Our attack is possible because the Just Works association model does not provide any protection against MITM attacks.

It is also possible that some SSP-enabled printers that have displays and buttons could use the Numeric Comparison or the Passkey Entry association model, which provide protection against MITM attacks. However, victim devices can be forced to use any association model that the attacker chooses [HyH07, SVA07] (see Subsection 6.11.1). Therefore, our attack works even against SSP-enabled printers that should provide MITM protection.

The main features of the attack are depicted in Figure 31. We next describe two scenarios for the attack.

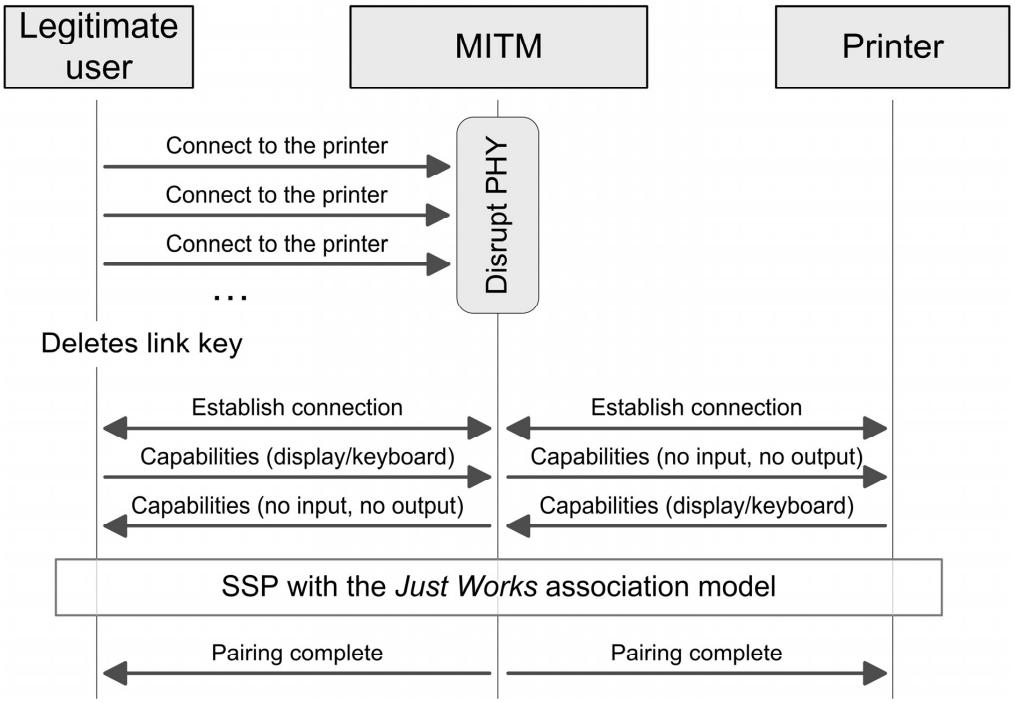

Figure 31. The main features of a BT-SSP-Printer-MITM attack. [HaH08]

We assume in the first scenario that the victim devices are using the Just Works association model. Our attack works in the following way: [HaH08]

1. *The MITM disrupts the PHY*: The MITM disrupts (jams) the PHY (see Subsection 6.11.1) until the frustrated user thinks that something is wrong with her Bluetooth devices and deletes previously stored link keys.

2. *The MITM impersonates the legitimate printer*: Since the user has deleted the previously stored link keys, she will initiate a new pairing process through SSP. The SSP pairing details are illustrated in Figure 32. Most of the notations used in Figure 32 have been explained above (see Table 4 in Section 4.1) and the rest are self-explanatory. It is worth noting that the user has deleted all information about the legitimate printer, including its BD_ADDR, so the MITM is not even required to clone the BD_ADDR of the legitimate printer in order to impersonate it. Now the

MITM only clones the user-friendly name of the legitimate SSP-enabled printer to impersonate it. Moreover, the MITM must be able to disrupt the legitimate printer in such a way that it cannot communicate with other legitimate Bluetooth devices. Therefore, when the user seeks available Bluetooth printers in the range, the only printer that is found will be the MITM with a different BD_ADDR but the same familiar user-friendly name. It is very likely that the user will not notice anything strange, because BD_ADDRs are much harder to remember than the user-friendly names. Therefore, the user will most likely choose the "MITM printer" that looks familiar to her. In this way, the MITM has replaced the legitimate printer in the Bluetooth network with the "MITM printer" with a different BD_ADDR. It is worth noting that by using a BD_ADDR different from that of the legitimate printer, the MITM can also eliminate any BD_ADDR collisions that may occur, i.e. the attack works more reliably and plausibly.

3. *The MITM intercepts all data*: When the legitimate Bluetooth devices are printing via a Bluetooth connection, the MITM captures (receives) all data and is also capable of decrypting it if encryption is used. Moreover, the MITM may even be capable of using certain services that these victim devices offer.

4. *The MITM relays the data to the legitimate printer*: Finally, the MITM relays the captured data to the legitimate printer. In this way, everything seems to work normally from the user's point of view: all documents are printed without any problems.

We assume in the second scenario that the victim devices are using the Numeric Comparison or the Passkey Entry association model. This attack works in the same way as our first attack scenario except that one additional phase is required: *the legitimate devices must be forced to use the Just Works association model* by using the attack scenarios described in Subsection 6.11.1. Note that since our two attack scenarios are designed against Bluetooth 2.1+EDR (SSP-enabled) printers, a MITM device is required between the victim devices for the attacks to work. Attacks against Bluetooth 2.0+EDR and earlier printers are easier in practice, because the MITM device is not required. Such attack scenarios and their practical implementations were described in Section 6.7.

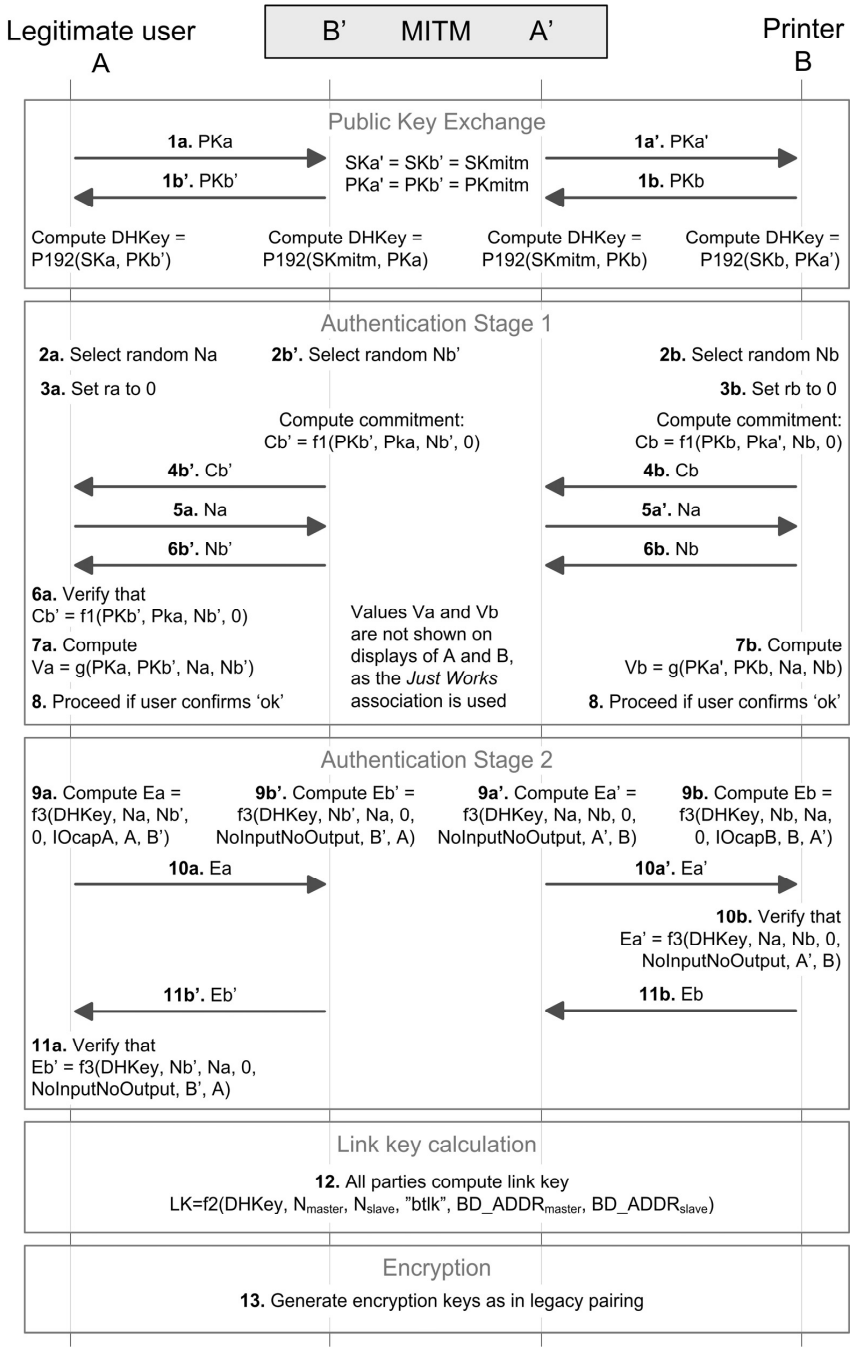

Figure 32. Pairing details of a BT-SSP-Printer-MITM attack. [HaH08]

The countermeasures for a *BT-SSP-Printer-MITM attack*, in addition to those described in Sections 6.1-6.7 and in Subsection 6.11.1, are: [HaH08]

- *Just Works as an optional (not mandatory) association model*: devices that cannot use the new window at the user interface level or alternatively NFC as an OOB channel (a better way) should implement their security either in the same way as old Bluetooth devices (versions up to 2.0+EDR) do, or not use Bluetooth security at all (if no sensitive data are exchanged). In this way, the implementation of the Just Works association model can be made optional and perhaps even removed altogether from the Bluetooth SSP specification. The one advantage of this approach is that it eliminates all MITM attacks against the Just Works association model. Moreover, if the Just Works association model is not supported in future Bluetooth devices, it will not be possible to force victim devices to use it.

- *OOB as a mandatory association model*: Future Bluetooth specifications should make OOB a mandatory association model in order to radically improve the security and usability of SSP. However, it is likely that such a radical change in the specification will not be possible at once. Therefore, future Bluetooth specifications should at least strongly recommend the use of an OOB channel (for example, NFC) to all Bluetooth device manufacturers.

6.11.3 Comparative Analysis of Bluetooth MITM Attacks

The first MITM attack on Bluetooth was devised by Jakobsson and Wetzel [JaW01] for version 1.0B of the standard. However, it works with all Bluetooth versions up to 2.0+EDR, because no major security improvements were implemented in those Bluetooth specifications. The attack assumes that the link key used by two victim devices is known to the attacker. The authors also showed how to obtain the link key using an "Off-Line PIN crunching attack" (see Off-Line PIN Recovery attack in Subsection 4.2.1), by passive eavesdropping on the initialization key establishment protocol. The MITM attack requires that both devices are using public or private security level (see Section 4.1), i.e. both victim devices must be connectable. In the attack, the BD_ADDRs of attacker devices must be cloned to equal the

addresses of the victim devices. Moreover, to prevent the jamming of the communication channel, the victim devices must be both masters or both slaves in two different piconets (see Section 2.2). In this case they transmit in an unsynchronized manner and cannot see each other's messages while communicating with the attacker. After establishing connection to both victims, the attacker sets up two new link keys.

Kügler [Küg03] further improves the attack of Jakobsson and Wetzel. By manipulating the clock settings, the attacker forces both victim devices to use the same channel hopping sequence but different clocks. In this way, the victim devices are unsynchronized, and can see only the messages the attacker sends them.

Kügler also shows how a MITM attack can be performed during the paging procedure. The attacker responds to the page request of the master victim faster than the slave victim, and restarts the paging procedure with the slave using a different clock. The master and slave use the same channel hopping sequence, but a different offset in this sequence. The attack works also in the case where both victim devices send and receive data packets over an encrypted link. Even though the IV (Initialization Vector) used for encryption depends on the clock, the last bit of the clock is unused. Therefore, the attacker can flip this last bit, forcing the victims to use clocks which have a difference of approximately 11.65 hours. Although the integrity of data is protected with CRCs which are appended to the plaintext prior to encryption, the attacker can manipulate intercepted ciphertext. After modifying the ciphertext in a certain way, the attacker updates the CRC bits (see [BGW01] for details); the integrity checks performed by the victims do not detect the modification. It must be noted, however, that the attacker does not have much time for manipulating the transmitted data.

Reflection attacks [LCA04] (also referred to as Relay attacks) aim at impersonating the victim devices. The main features of the Reflection attacks were described in Subsection 4.2.2. The reflection attack can be one-sided, in which only one victim device is impersonated, or two-sided, whereby both victim devices are impersonated. The attacker must use two Bluetooth devices with adjustable BD_ADDRs (for example, protocol analyzers). In addition, the attacker must be capable of relaying the received information between her devices, because victim devices can be far away from each other. During the paging procedure, the attacker

responds to the request of the first victim device (A), and initiates a connection to the second victim device (B), posing as A. If the victim devices can hear each other, the mechanisms described in [Küg03] may be used to achieve this. After this, the attacks work on the LMP layer of Bluetooth. The messages of the protocol are simply relayed by the attacker's devices. In the case of a one-sided attack only some of the messages must be relayed, and the connection to A is dropped when the attacker has impersonated it to B. The attacker can successfully perform authentication by using reflection attacks, but she cannot continue the attack if the target devices encrypt their communication. By combining reflection attacks with a known secret PIN code, link key or encryption key, the attacker can both impersonate the victim devices and decrypt the information transferred between them.

Victim devices can detect the attack by noticing a considerable increase in the latency of getting the response to authentication challenge, caused by relaying. This countermeasure is not described in the standards, and it is up to the discretion of manufacturers to provide it.

Bluetooth 2.1+EDR provides protection against the MITM attacks described in this subsection, by means of SSP (see Section 4.1). However, we have shown that MITM attacks against Bluetooth 2.1+EDR devices are also possible [HaH08, HyH07], and so did [SVA07]. Because SSP supports several association models, the selection of which depends on the capabilities of the target devices, the attacker can force the devices to use a less secure mode by changing the capabilities information. For example, by forcing the use of the Just Works association model the attacker can bypass all the security checks which would normally be in place. The association is then unauthenticated; the devices are aware of this fact, but how they react depends on the manufacturer. If the victim devices have already established a link key, the attacker can use jamming to disrupt the communication, and then initiate the connection under a chosen association model with both devices. As a result, the attacker learns the link key used by the devices and can both intercept and decrypt all data transmitted between the devices.

In Table 8 we summarize the properties of the MITM attacks overviewed in this section. It is interesting to note the connection of MITM attacks to other developments in the Bluetooth security analysis. For instance, at the time when most of the MITM attacks were introduced,

implementing them was not an easy task, as there were no devices with adjustable BD_ADDRs, except sophisticated and expensive protocol analyzers. Now the situation has changed: Bluetooth devices with an adjustable BD_ADDR are readily available, and techniques for finding hidden (non-discoverable) Bluetooth devices have been invented (see Subsection 4.2.4 and Sections 4.2, 6.1 and 6.4). Therefore, the danger of MITM attacks has recently increased considerably.

MITM attack: Attack properties:	[JaW01]	[Küg03]	[LCA04]	[SVA07]	[HyH07]	[HaH08]
Bluetooth versions	1.0 – 2.0+EDR	1.0 – 2.0+EDR	1.0 – 2.0+EDR	2.1+EDR	2.1+EDR	2.1+EDR
Attack goals	Impersonation, modification	Impersonation, modification	Impersonation	Impersonation, modification	Impersonation, modification	Impersonation, modification
Attacking devices	2	2	2	2+1; note that a jamming device is also required	2+1; note that a jamming device is also required	2+1; note that a jamming device is also required
Devices attacked	Connectable	Connectable or non-connectable	Connectable or non-connectable	Connectable or non-connectable	Connectable or non-connectable	Connectable or non-connectable
Distances	Any[1]	Any[1]	Any[1]: note that the victim devices must be out of each other's range	Any[1]	Any[1]	Any[1]
Detection	By user: entering PIN to renegotiate	The attack remains undetected	By devices: delays in getting the LMP authentication response	By user: one of the devices asks to compare numbers, the other one does not	By user: no Numeric Comparison is used although both devices have displays and keyboards	By user: a suspicious user may notice the difference between BD_ADDR values
Main countermeasures	Policies protecting against MITM attacks	Cryptography integrity checks of packets	Detecting the delays	At the user interface level	At the user interface level	At the user interface level, modifications to SSP specification

[1] The attacker must use two Bluetooth adapters; actual distance is limited by speed of the link between the attacker's devices.

Table 8. MITM attacks on Bluetooth: summary and comparison. [HaH08]

In general, MITM attacks are hard to prevent in wireless networks. By far the best way to stop such attacks is to use an OOB channel, and SSP supports this option. However, the usability of the OOB channel is of great importance: if wires must be used to pair wireless devices, one is likely to opt for less secure but more usable options. We concur with the designers of SSP on their suggestion to use NFC as the OOB channel.

6.12 Novel Intrusion Detection and Prevention System

In this section, we explain how various Bluetooth security attacks in progress can be prevented and stopped by monitoring communication to discover such attacks. Moreover, we propose a new efficient Intrusion Detection and Prevention System [Haa08a] for Bluetooth networks to prevent and stop attacks in progress. The proposed system is based on the set of rules that are used to identify strange communication behaviour in Bluetooth devices.

Based on strange communication behaviour in Bluetooth devices which are performing various security attacks, we defined a set of rules to help identify attacks in progress: [Haa08a]

1. *Unusually many repeated failed authentication attempts*: This may indicate that an attacker is using an On-Line PIN Cracking attack (see Subsection 4.2.4 and Section 6.3) to discover the secret PIN code of a victim device.

2. *Unusually many repeated successful authentications and disconnections*: This may indicate that the attacker is performing a DoS attack (see Subsection 4.2.3 and Sections 6.8-6.10).

3. *Unusually many NAK transmissions*: This may indicate that the attacker is performing a Big NAK attack (see Subsection 4.2.3 and Section 6.10), thus putting the victim device on an endless retransmission loop.

4. *Unusually long delays*: This may indicate that a MITM is between the communicating parties (see Sections 3.4, 4.1 and 6.11).

5. *Unusually many repeated POLL packets*: This may indicate that the attacker is keeping victim devices busy so that they will not go into sleep or low-power mode. During the normal piconet operation, the master device can use POLL packets to check that slave devices are still alive (i.e. up and running), and slave devices must always respond to POLL packet.

6. *Unusually high BER*: This may indicate that the attacker is disrupting the PHY.

7. *Unusually heavy traffic between two communicating parties*: This may indicate that the attacker is performing a Battery Exhaustion attack (see Subsection 4.2.3).

8. *Sudden increase in transmit powers*: This may indicate that the attacker is using a stronger RF signal to displace the active piconet device via an Exploitation of a stronger RF signal attack (see Subsection 4.2.2).

9. *Two identical BD_ADDRs in the range of vulnerability*: This may indicate that the attacker is using a BD_ADDR Duplication attack (see Subsection 4.2.3 and Section 6.8) to deny the legitimate piconet devices access to the services. Another possibility is that the attacker is performing an impersonation attack (for example, a BTPrinterBugging via Impersonation attack; see Subsection 6.7.2) to mislead the legitimate piconet devices.

10. *An HV1 SCO link established with the piconet master when another type of SCO or eSCO link could also have been used*: This may indicate that the attacker is performing a SCO/eSCO attack (see Subsection 4.2.3 and Section 6.9) to reserve all piconet resources so that the legitimate piconet devices do not get service within a reasonable time.

11. *An L2CAP level request for the highest possible data rate or the smallest possible latency*: If such a request is accepted, all throughput is reserved for the attacker and the legitimate piconet devices do not get service within a reasonable time, i.e. the attacker is performing an L2CAP Guaranteed Service attack (see Subsection 4.2.3).

12. *Surprising connection attempts and data transfer requests from unknown Bluetooth devices*: This may indicate that a Bluetooth virus or worm (see Subsection 4.2.4) is trying to infect legitimate piconet devices.

13. *A Bluetooth device requests that the length of an encryption key must be shorter than 128 bits*: This may indicate that an attack against the Bluetooth encryption is in progress (see Subsection 4.3.2).

14. *An RF signature mismatch*: This indicates that some kind of attack, such as an impersonation attack, is in progress. Every transmitter has a unique RF signature [Mor02, Sha06] which can be used to differentiate the legitimate devices from those that have alien RF signatures, i.e. a sample RF signature is needed from each legitimate device in order to detect alien RF signatures.

15. *SSP's Just Works association model activated between devices that could use a more secure option (for example, Numeric Comparison or OOB)*: This may indicate that a MITM is between the communicating parties. Only Bluetooth 2.1+EDR (or later) devices support SSP (see Section 4.1).

In order to mitigate various Bluetooth security attacks, we proposed a scheme consisting of two parts: an *Intrusion Detection System* and an *Intrusion Prevention System*. In our system a commercially available Bluetooth protocol analyzer, such as a LeCroy BTTracer/Trainer [Lec07] (see Chapter 5) equipped with signalling processing capabilities (some additional signalling processing hardware is required) takes care of the intrusion detection part.

When an intrusion is detected, the protocol analyzer immediately informs the network administrator (via Bluetooth) that the Bluetooth network is under attack. This is *manual administrative intrusion prevention,* which can be used in all cases regardless of the capabilities of the legitimate Bluetooth devices. This system also requires LeCroy BTTracer/Trainer v2.2 software [Lec07] (or later), which provides CATC Scripting Language [Lec04] (see Chapter 5).

The second part of our system, the *Intrusion Prevention System*, is a small program that runs on all legitimate Bluetooth devices that allow programs to be installed, i.e. at least all PCs, laptops and mobile phones should be supported. This is *automatic intrusion prevention*. It requires that all legitimate Bluetooth devices must run this special program in order to receive warning messages from our Intrusion Detection System. When a warning message is received, devices that are under attack perform automatic disconnection and refuse any further Bluetooth connections for a predetermined time. The Intrusion Detection System also sends enough information (BD_ADDR, device capabilities information, the user-friendly name of

the device, RF signature information, and so on) to the Intrusion Prevention System so that further connections from the same origin can be refused immediately by the Intrusion Prevention System.

Our Intrusion Detection System should work in the following way: [Haa08a]

1. *A Bluetooth protocol analyzer monitors Bluetooth communication of the legitimate piconet devices non-stop*: Protocol analyzers have all the legitimate BD_ADDR values that are allowed to communicate within the piconet and also other useful information about such devices (device capabilities information, the user-friendly name of the device, RF signature information, and so on).

2. *When an intrusion is detected, either manual administrative intrusion prevention or automatic intrusion prevention is applied*: We strongly recommend that automatic intrusion prevention should be implemented. However, if automatic intrusion prevention is not implemented, at least Bluetooth the network administrator is alerted immediately.

Our automatic Intrusion Prevention System should work in the following way: [Haa08a]

1. *The Intrusion Prevention System receives a warning message from the Intrusion Detection System*: When a warning message is received, automatic disconnection is performed and further Bluetooth connections are refused for a predetermined time.

2. *The Intrusion Prevention System also receives enough information to prevent further attacks from the same origin*: We recommend that at least the following information about the attacking device should be received from the Intrusion Detection System and stored in the database: the BD_ADDR value, device capabilities information, the user-friendly name of the device, and RF signature information.

It is difficult to create an intrusion detection and prevention system that caters for all possible types of security attacks, as the security of Bluetooth is likely to be limited by the capabilities of the least powerful or the least secure device type. In fact, most Bluetooth security attacks are based on precisely this problem.

In general, security attacks are hard to prevent in wireless networks, especially when no intrusion detection and prevention system is used. Our proposal is intended to help Bluetooth network administrators and Bluetooth device manufacturers to implement efficient Bluetooth intrusion detection and prevention systems.

6.13 Further Classification of Bluetooth-enabled Ad-hoc Networks

Since ad-hoc networks have no fixed infrastructure, there are many different kinds of Bluetooth-enabled ad-hoc networks with various security procedures and security requirements. In this section, we further classify Bluetooth-enabled ad-hoc networks and their security procedures/requirements depending on a risk analysis within each classified group. We also evaluate the breaches and damage that can be inflicted by various attacks in such scenarios.

We classified Bluetooth-enabled ad-hoc networks into five different groups: [Haa08b]

1. *Personal Home Network*: In a Personal Home Network, a user connects Bluetooth devices, types initial PIN codes manually, and adds/removes Bluetooth devices to/from the Personal Home Network herself within a relatively secure environment.

2. *Personal Office Network*: In a Personal Office Network, the user connects Bluetooth devices, types initial PIN codes manually, and adds/removes Bluetooth devices to/from the Personal Office Network herself within a relatively insecure environment.

3. *Organization Network*: In an Organization Network, employees use their company's Bluetooth devices and form piconets/scatternets for information exchange within a relatively insecure environment. Employees do not have to type any PIN codes manually, because all link keys are stored in the organization's Bluetooth devices. Only the company's network administrator is allowed to add/remove Bluetooth devices to/from the Organization Network (excluding employees' own Bluetooth devices). Depending on the organization's data security policy, employees may or may not be allowed to use their own Bluetooth devices in the Organization Network.

4. *Conference Network*: In a Conference Network, attendees use their own Bluetooth devices and form a piconet for information exchange within an insecure environment. The attendees type initial PIN codes manually and add/remove Bluetooth devices to/from the Conference Network themselves.

5. *Public Network*: In a Public Network, attendees use their own Bluetooth devices and form piconets/scatternets for information exchange within a very insecure environment. The attendees type initial PIN codes manually if security operations are required at all. Typically, public networks do not require any security measures. Moreover, attendees add/remove Bluetooth devices to/from the Public Network themselves.

Table 9 shows a risk analysis for each group and also gives an evaluation of the possible breaches and damage that can be inflicted by various attacks in such scenarios. All text and notations in the table are either self-explanatory or have been explained above.

New groups:	Personal Home Network	Personal Office Network	Organization Network	Conference Network	Public Network
Environment	Relatively secure	Relatively insecure	Relatively insecure	Insecure	Very insecure
PIN codes	Typically very short and easy to guess	Typically very short and easy to guess	Typically quite long and hard to guess	Typically very short and easy to guess	Typically no PIN code required
Physical protection of Bluetooth devices	Closely related to the physical protection of home	Closely related to the physical protection of office	Closely related to the physical protection of organization's facilities	Closely related to the physical protection of conference attendees' hotel rooms	Closely related to the physical protection of user's own devices
DID attacks	Possible and quite easy	Possible and easy	Possible and easy	Possible and easy	Possible and very easy
Intrusion detection system?	Typically no intrusion detection system	Typically no intrusion detection system	Depends on the organization's data security policy	Typically no intrusion detection system	Typically no intrusion detection system
Special weak points	Bluetooth-enabled printers, snoopy neighbours	Bluetooth-enabled printers, snoopy employees/visitors	Bluetooth-enabled printers, snoopy employees/visitors, employees' Bluetooth devices, organization's Bluetooth devices can be replaced by attacker's spying devices	Bluetooth-enabled printers, snoopy attendees/visitors, attackers may infiltrate into conference	Bluetooth-enabled printers, snoopy users of the Public Network, easy access to the network, bugging devices
What can be stolen?	Social security numbers, bank account information, documents related to business issues, and such Bluetooth devices that have access to some sensitive files	Social security numbers, bank account information, documents related to business issues, and such Bluetooth devices that have access to some sensitive files	Social security numbers, bank account information, documents related to business issues, and such Bluetooth devices that have access to some sensitive files	Social security numbers, unpublished research papers, documents related to business issues, and such Bluetooth devices that have access to some sensitive files	Almost everything, because typically no security is applied!

Table 9. A risk analysis and an evaluation of possible breaches and damage. [Haa08b]

The security procedures (countermeasures) to prevent malicious Bluetooth devices from stealing information from other Bluetooth devices are the same as those discussed above in Sections 6.1-6.11.

7 CONCLUSIONS AND FUTURE WORK

This thesis presents our studies of Bluetooth security in a research laboratory environment. Our research work can be roughly divided into four parts. First, Bluetooth security weaknesses were studied and a Bluetooth security laboratory environment for implementing Bluetooth security attacks in practice was built. Secondly, different types of attacks against Bluetooth security were investigated and the feasibility of some of them were demonstrated in our research laboratory. Countermeasures against each type of attacks were also proposed. Thirdly, new proof-of-concept security analysis tools were implemented, new attacks against Bluetooth security were presented, and countermeasures that render these attacks impractical were proposed. Finally, a comparative analysis of the existing MITM attacks on Bluetooth was provided, a novel system for detecting and preventing intrusions in Bluetooth networks was described, and a further classification of Bluetooth-enabled ad-hoc networks depending on a risk analysis within each classified group was presented.

Overall security in Bluetooth networks is based on the security of the Bluetooth medium, the security of Bluetooth protocols and the security parameters used in Bluetooth communication. There are several weaknesses in the Bluetooth medium, Bluetooth protocols and Bluetooth security parameters which can significantly weaken the overall security of Bluetooth networks. Currently, weaknesses in the Bluetooth security parameters seem to be the biggest problem in Bluetooth security.

The current level of security is insufficient in many Bluetooth devices on the market, as our research work clearly shows. Many kinds of Bluetooth devices have very short, often only four-digit, fixed PIN codes. This is clearly a big security risk. Therefore, Bluetooth device manufacturers should take security issues more seriously. In addition, users' understanding of security issues is very important for protecting sensitive data against eavesdroppers and hackers. Moreover, many users have no idea how to configure their Bluetooth devices' security settings correctly. Application layer key exchange and encryption methods can also be used as an extra security in addition to the Bluetooth built-in security.

Many attacks, such as Reflection attacks and Interception of Packets attacks, are possible because encryption is not used by default in many kinds of Bluetooth devices. We strongly recommend that Bluetooth factory settings should enable encryption by default, because many users do not know about its existence or do not know how to set it up successfully. Another possibility is to set Bluetooth encryption as mandatory. This of course would require minor changes to the Bluetooth specification, and it would mean that older Bluetooth devices would also have to use encryption or communication with new devices would not be possible. On the other hand, if backward-compatibility with old Bluetooth devices must be guaranteed, mandatory encryption is not possible. Many attacks, such as Off-Line PIN Recovery attacks and On-Line PIN Cracking attacks, are also possible because many kinds of Bluetooth devices have very short fixed PIN codes containing only digits. We strongly recommend that the allowed sixteen 8-bit character PIN codes should always be used when possible.

Dozens of attacks are also possible because many kinds of Bluetooth devices have public security level as a fixed factory setting, so they are always discoverable. We strongly recommend that the security level of Bluetooth devices should not be public by default or as the fixed factory setting. The user should have at least the option of changing the factory security level setting somehow. However, in the near future, when techniques for finding hidden Bluetooth devices in an average of one minute are actually ported into the firmware of a standard $30 Bluetooth USB dongle, the private security level will no longer provide significant protection.

Since there are billions of Bluetooth devices in use without SSP's improved security features, malicious security violations are not expected to decrease in the near future. On the contrary, these old Bluetooth devices will be sold for many years to come, thus making security concerns even more alarming.

SSP has gone through a series of reviews by experts, and the released version generally does good work in improving the security of Bluetooth pairing. However, MITM attacks against SPP are also possible, as our research work clearly shows. Therefore, Bluetooth security architecture needs to be further updated to prevent these new threats. The next major security

improvements are roadmapped to the upcoming Bluetooth specification (Seattle), which is expected to be released by the Bluetooth SIG in 2009.

In general, MITM attacks are hard to prevent in wireless networks. By far the best way to stop such attacks is to use SSP's OOB channel. We concur with the designers of SSP on their suggestion to use NFC as the OOB channel.

The problems we want to investigate in our future research work are concerned with the following issues:

1. Bluetooth is a relatively new wireless technology and therefore new attacks against Bluetooth security are likely to be found. We want to further investigate Bluetooth security weaknesses and propose countermeasures against new attacks.

2. Issues related to Bluetooth user experience (ease of use) have become more and more important in recent years. Therefore, we want to investigate how enhanced user experience will affect Bluetooth security in various scenarios, including social aspects and user acceptance/habits in security management. Moreover, we want to devise best practices depending on the risk analysis within each scenario.

3. Since we have already proposed a new efficient Intrusion Detection and Prevention System for Bluetooth networks, we want to implement a working prototype of such a system and also analyse its efficiency.

4. Because cheap Bluetooth devices with an adjustable BD_ADDR are readily available, tools for modifying official firmwares have been released, techniques for finding hidden Bluetooth devices in an average of one minute have been invented, and an open-source Bluetooth sniffer for Linux environments has been released, we want to continue our practical Bluetooth security research under Linux using these new tools. We also want to further develop the existing open-source Bluetooth sniffer to include the BD_ADDR duplication feature and the graphical user interface in order to make it user-friendly.

5. At the time when most of the Bluetooth MITM attacks were introduced, implementing them was not an easy task, as there were no devices with adjustable BD_ADDRs, except sophisticated and expensive protocol analyzers. Now the situation has changed, and we want to make practical implementations of all existing Bluetooth MITM attacks. Moreover, we want to analyse the results of the practical experiments, draw conclusions, and propose practical countermeasures based on our findings.

6. Since there are many new emerging wireless technologies, such as ZigBee [Zig08] and UWB, which are quite similar to Bluetooth technology, it is expected that our work presented in this thesis can be quite easily extended to cover the security of these new technologies. Therefore, we want to investige how various Bluetooth security attacks and their countermeasures can be ported to support ZigBee and UWB technologies, for example.

The research work described in this thesis is only the tip of the iceberg. Bluetooth security intimately depends on general problems of ad-hoc network security, on physical aspects of protecting wireless networks, on cryptographical solutions to key distribution without Trusted Third Party or CA infrastructure, and on application layers. The research in this area combines various skills and techniques, requires cooperation with other researchers, and also requires a certain infrastructure.

REFERENCES

[And01] Anderson R.: *Security Engineering − A Guide to Building Dependable Distributed Systems*. New York, Wiley & Sons, 2001.

[ArK03] Armknecht F. and Krause M.: *Algebraic Attacks on Combiners with Memory*. Proceedings of the 23rd Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'2003), LNCS, Vol. 2729, Springer-Verlag, 2003, pp. 162-175.

[BCK96a] Bellare M., Canetti R., and Krawczyk H.: *Keying Hash Functions for Message Authentication*. Proceedings of the CRYPTO'96, New York, Springer-Verlag, August 1996.

[BCK96b] Bellare M., Canetti R., and Krawczyk H.: *Message Authentication using Hash Functions − The HMAC Construction*. RSA Laboratories' CryptoBytes, Vol. 2, No. 1, Spring 1996.

[Bel08] Belkin International: *Belkin Bluetooth Printer Adapter*. Belkin International, homepage, 2008. http://www.amazon.co.uk/Belkin-Bluetooth-USB-Printer-Adapter/dp/tech-data/B0001Q17NW/ref=de_a_smtd/203-7526837-1808712 (31.10. 2008)

[BGW01] Borisov N., Goldberg I., and Wagner D.: *Intercepting Mobile Communications − The Insecurity on 802.11*. Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, ACM Press, 2001.

[Blu01] Bluetooth SIG: *Bluetooth specification 1.1*. Bluetooth SIG, technical specification, February 2001. https://www.bluetooth.org/apps/content (31.10. 2008)

[Blu03] Bluetooth SIG: *Bluetooth specification 1.2*. Bluetooth SIG, technical specification, November 2003. https://www.bluetooth.org/apps/content (31.10. 2008)

[Blu04a] Bluetooth SIG: *Bluetooth specification 2.0+EDR*. Bluetooth SIG, technical specification, November 2004. https://www.bluetooth.org/apps/content (31.10. 2008)

[Blu04b] Bluetooth SIG: *Bluetooth Special Interest Group Launches Bluetooth Core Specification Version 2.0 + Enhanced Data Rate*. Bluetooth SIG, press release, November 8, 2004. http://www.bluetooth.com/Bluetooth/Press/SIG (31.10. 2008)

[Blu06a] Bluetooth SIG: *Bluetooth Wireless Technology Surpasses One Billion Devices*. Bluetooth SIG, press release, November 13, 2006. http://www.bluetooth.com/ Bluetooth/Press/SIG (31.10.2008)

[Blu06b] Bluetooth SIG: *Bluetooth SIG Selects WiMedia Alliance Ultra-Wideband Technology for High Speed Bluetooth Applications*. Bluetooth SIG, press release, March 28, 2006. http://www.bluetooth.com/Bluetooth/Press/SIG (31.10.2008)

[Blu07a] Bluetooth SIG: *Bluetooth specification 2.1+EDR*. Bluetooth SIG, technical specification, July 2007. http://www.bluetooth.com (31.10.2008)

[Blu07b] Bluetooth SIG: *Overview − About the Bluetooth SIG*. Bluetooth SIG, homepage, 2007. https://programs.bluetooth.org/apps/content/?doc_id=49700 (31.10.2008)

[Blu07c] Bluetooth security & Bluetooth hackers community blog: *Bluetooth Sniffing For Less*. Bluetooth security & Bluetooth hackers community blog, 2007. http:// bluetoothsecurity.wordpress.com/2007/05/12/bluetooth-sniffing-for-less (31.10. 2008)

[Blu07d] Bluediving Project: *Bluediving − Next generation Bluetooth security tool*. Bluediving Project, homepage, 2007. http://bluediving.sourceforge.net (31.10.2008)

[Blu08a] Bluetooth SIG: *2008 Marks Ten Years of Bluetooth Wireless Technology*. Bluetooth SIG, press release, January 7, 2008. http://www.bluetooth.com/Bluetooth/Press/SIG/2008_MARKS_TEN_YEARS_OF_emBLUETOOTHem_WIRELESS_TECHNOLOGY.htm (31.10.2008)

[Blu08b] BlueZ Project: *BlueZ − Official Linux Bluetooth protocol stack*. BlueZ Project, homepage, 2008. http://www.bluez.org (31.10.2008)

[Blu99a] Bluetooth SIG: *Bluetooth specification 1.0A*. Bluetooth SIG, technical specification, July 1999. https://www.bluetooth.org/apps/content (31.10.2008)

[Blu99b] Bluetooth SIG: *Bluetooth specification 1.0B*. Bluetooth SIG, technical specification, December 1999. https://www.bluetooth.org/apps/content (31.10.2008)

[Bon07] Bona Computech: *Mentor Bluetooth Printer Adapter*. Bona Computech, homepage, 2007. http://www.bona.com.tw:8080/product/spec/BT-PR.doc (27.11.2007)

[Bra08] Bragg R.: *The Encrypting File System*. Microsoft Corporation, TechNet Security Homepage, 2008. http://www.microsoft.com/technet/security/topics/cryptographyetc/efs.mspx (31.10.2008)

[Buc01] Buchmann J.: *Introduction to Cryptography*. New York, Springer-Verlag, 2001.

[Che05a] Cheung H.: *SmallNetBuilder − How To: Building a BlueSniper Rifle − Part 1*. SmallNetBuilder, Pudai LLC, 2005. http://www.smallnetbuilder.com/content/view/24256/98 (31.10.2008)

[Che05b] Cheung H.: *SmallNetBuilder − How To: Building a BlueSniper Rifle − Part 2*. SmallNetBuilder, Pudai LLC, 2005. http://www.smallnetbuilder.com/content/view/24228/98 (31.10.2008)

[Con08] Conceptronic: *Conceptronic Bluetooth Printer Adapter*. Conceptronic, homepage, 2007. http://www.conceptronic.net/site/desktopdefault.aspx?tabindex=0&tabid=200&Cat=10&grp=1020&ar=&Prod_ID=451&Prod=CBTPU&subid=559 (31.10.2008)

[Cyb94] Cyberpunks: *Cyberpunks mailing list − RC4 Source Code*. Cyberpunks, 1994. http://www.columbia.edu/~ariel/ssleay/rrc4.html (31.10.2008)

[Dar07] Darkircop: *CSR Sniffer − Firmware assembler and disassembler*. Darkircop, 2007. http://darkircop.org/bt/bt.tgz (31.10.2008)

[DiH76a] Diffie W. and Hellman M.: *New Directions in Cryptography*. Proceedings of the AFIPS National Computer Conference, June 1976.

[DiH76b] Diffie W. and Hellman M.: *Multiuser Cryptographic Techniques*. IEEE Transactions on Information Theory, November 1976.

[Dis08] Distributed.net Project: *Distributed.net − Node Zero*. Distributed.net Project, homepage, 2008. http://www.distributed.net (31.10.2008)

[Ele98] Electronic Frontier Foundation: *Cracking DES − Secrets of Encryption Research, Wiretap Politics, and Chip Design*. Sebastopol, CA, O'Reilly, 1998.

[Epo05] Epox: *BT-DG07A+ − A Class 1 Compatible Bluetooth 2 Dongle*. Epox, homepage, 2005. http://www.epox.nl/products/view.php?product_id=373 (27.11.2007)

[Ett07] Ettus Research: *Universal Software Radio Peripheral*. Ettus Research, homepage, 2007. http://www.ettus.com/downloads/usrp_v4.pdf (31.10.2008)

[Fei73] Feistel H.: *Cryptography and Computer Privacy*. Scientific American, May 1973.

[FlM00] Fluhrer S. and McGrew D.: *Statistical Analysis of the Alleged RC4 Key Stream Generator*. Proceedings of the Fast Software Encryption 2000, LNCS, Vol. 1978, Springer-Verlag, 2000, pp. 19-30.

[FlL01] Fluhrer S. and Lucks S.: *Analysis of the E_0 encryption system*. Proceedings of the 8th Workshop on Selected Areas in Cryptography, LNCS, Vol. 2259, Springer-Verlag, 2001, pp. 38-48.

[Flu02] Fluhrer S.: *Improved key recovery of level 1 of the Bluetooth Encryption System*. Cryptology ePrint Archive, Research Report 2002/068, 2002.

[FMS01] Fluhrer S., Mantin I., and Shamir A.: *Weakness in the Key Scheduling Algorithm of RC4*. Proceedings of the Workshop in Selected Areas of Cryptography, 2001.

[Fro08] Frontline: *FTS4BT Wireless Bluetooth Protocol Analyzer & Packet Sniffer*. Frontline, homepage, 2008. http://www.fte.com/products/FTS4BT-01.asp (31. 10.2008)

[FSe05a] F-Secure Corporation: *F-Secure Virus Descriptions − Skulls.D*. F-Secure Corporation, homepage, 2005. http://www.f-secure.com/v-descs/skulls_d.shtml (31.10.2008)

[FSe05b] F-Secure Corporation: *F-Secure Virus Descriptions − Lasco.A*. F-Secure Corporation, homepage, 2005. http://www.f-secure.com/v-descs/lasco_a.shtml (31.10.2008)

[FSe06] F-Secure Corporation: *F-Secure Virus Descriptions − Cabir*. F-Secure Corporation, homepage, 2006. http://www.f-secure.com/v-descs/cabir.shtml (31. 10.2008)

[GNU08] GNU Radio Project: *GNU Radio − The GNU Software Radio*. GNU Radio Project, homepage, 2008. http://www.gnu.org/software/gnuradio (31.10.2008)

[Gou08] Gough V.: *EncFS − Encrypted Filesystem module for Linux*. EncFS Project, homepage, 2008. http://arg0.net/wiki/encfs (31.10.2008)

[GPS04] Gehrmann C., Persson J., and Smeets B.: *Bluetooth security*. Boston, Artech House, 2004.

[Haa00] Haartsen J.: *The Bluetooth radio system*. IEEE Personal Communications, Vol. 7, No. 1, February 2000, pp. 28-36.

[Haa05a] Haataja K.: *Bluetooth Security Threats and Possible Countermeasures*. Proceedings of the Annual Finnish Data Processing Week at the University of Petrozavodsk (FDPW'2004), Advances in Methods of Modern Information Technology, Vol. 6, Petrozavodsk, 2005, pp. 116-150.

[Haa05b] Haataja K.: *Two Practical Attacks Against Bluetooth Security Using New Enhanced Implementations of Security Analysis Tools*. Proceedings of the IASTED International Conference on Communication, Network and Information Security (CNIS'2005), Phoenix, Arizona, USA, November 14-16, 2005, pp. 13-18.

[Haa06] Haataja K.: *Bluetooth Network Vulnerability to Disclosure, Integrity and Denial-of-Service Attacks*. Proceedings of the Annual Finnish Data Processing Week at the University of Petrozavodsk (FDPW'2005), Advances in Methods of Modern Information Technology, Vol. 7, Petrozavodsk, 2006, pp. 63-103.

[Haa07a] Haataja K.: *Three Practical Bluetooth Security Attacks Using New Efficient Implementations of Security Analysis Tools*. Proceedings of the IASTED International Conference on Communication, Network and Information Security (CNIS'2007), Berkeley, California, USA, September 24-26, 2007, pp. 101-108.

[Haa07b] Haataja K.: *New Practical Attack Against Bluetooth Security Using Efficient Implementations of Security Analysis Tools*. Proceedings of the IASTED International Conference on Communication, Network and Information Security (CNIS'2007), Berkeley, California, USA, September 24-26, 2007, pp. 134-142.

[Haa08a] Haataja K.: *New Efficient Intrusion Detection and Prevention System for Bluetooth Networks*. Proceedings of the ACM International Conference on Mobile, Wireless MiddleWare, Operating Systems, and Applications (Mobilware'2008), Innsbruck, Austria, February 12-15, 2008.

[Haa08b] Haataja K.: *Further Classification of Bluetooth-enabled Ad-hoc Networks Depending on a Risk Analysis Within Each Classified Group*. Proceedings of the IEEE Seventh International Conference on Networking (ICN'2008), Cancun, Mexico, April 13-18, 2008, pp. 232-237.

[HaH08] Haataja K. and Hyppönen K.: *Man-In-The-Middle Attacks on Bluetooth − a Comparative Analysis, a Novel Attack, and Countermeasures*. Proceedings of the IEEE Third International Symposium on Communications, Control and Signal Processing (ISCCSP'2008), St. Julians, Malta, March 12-14, 2008, pp. 1096-1102.

[HeM04] Herfurt M. and Mulliner C.: *BluePrinting − Remote Device Identification based on Bluetooth Fingerprinting Techniques*. Berliner Congress Center, Berlin, Germany, December 27-29, 2004.

[Her04] Herfurt M.: *Detecting and Attacking bluetooth-enabled Cellphones at the Hannover Fairground.* CeBIT 2004, March 30, 2004.

[HHH06] Hassinen M., Hyppönen K., and Haataja K.: *An Open, PKI-Based Mobile Payment System*. Proceedings of the ACM/IEEE/Springer International Conference on Emerging Trends in Information and Communication Security (ETRICS'2006), LNCS, Vol. 3995, Springer-Verlag, June 6-9, 2006, pp. 86-100.

[Hol06] Holtmann M.: *Playing BlueZ on the D-Bus*. Proceedings of the Linux Symposium, Vol. 1, Ottawa, Ontario, Canada, July 19-22, 2006, pp. 421-425.

[HyH07] Hyppönen K. and Haataja K.: *"Niño" Man-In-The-Middle Attack on Bluetooth Secure Simple Pairing*. Proceedings of the IEEE Third International Conference in Central Asia on Internet, The Next Generation of Mobile, Wireless and Optical Communications Networks (ICI'2007), Tashkent, Uzbekistan, September 26-28, 2007.

[IEE07] IEEE Standards Association: *IEEE 802.11 specifications*. IEEE Standards Association, technical specifications, 1999-2007. http://standards.ieee.org/getieee802/802.11.html (31.10.2008)

[IEE08] IEEE Registration Authority: *IEEE Public OUI and Company_id Assignments*. IEEE Registration Authority, 2008. http://standards.ieee.org/regauth/oui/oui.txt (31.10.2008)

[IET00] IETF: *RFC 2828 − Internet Security Glossary*. IETF, May 2000. http://www.rfc-archive.org/getrfc.php?rfc=2828 (31.10.2008)

[IET05a] IETF: *RFC 3931 − Layer Two Tunneling Protocol, Version 3 (L2TPv3)*. IETF, March 2005. http://tools.ietf.org/html/rfc3931 (31.10.2008)

[IET05b] IETF: *RFCs 4301-4309 − a third generation of IPSec RFCs*. IETF, 2005.

[IET06] IETF: *RFC 4346 − The Transport Layer Security (TLS) Protocol Version 3.1*. IETF, April 2006. http://tools.ietf.org/html/rfc4346 (31.10.2008)

[IET07] IETF: *RFC 4835 − Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)*. IETF, April 2007. http://tools.ietf.org/html/rfc4835 (31.10.2008)

[IET99] IETF: *RFC 2661 − Layer Two Tunneling Protocol (L2TP)*. IETF, August 1999. http://tools.ietf.org/html/rfc2661 (31.10.2008)

[Inf03] Infrared Data Association: *IrDA Object Exchange Protocol (OBEX) specifications*. Infrared Data Association, technical specifications, 1997-2003. http://www.irda.org (31.10.2008)

[Inf05] Infrared Data Association: *IrDA specifications*. Infrared Data Association, technical specifications, 1993-2005. http://www.irda.org (31.10.2008)

[IRC08] IRC.org: *Internet Relay Chat − Information about IRC*. IRC.org, homepage, 2008. http://www.irc.org (31.10.2008)

[ITU91] ITU-T: *Recommendation X.800 − Security Architecture for Open Systems Interconnection for CCITT Applications*. ITU-T, Geneva, 1991. http://fag.grm.hia.no/IKT7000/litteratur/paper/x800.pdf (31.10.2008)

[JaW01] Jakobsson M. and Wetzel S.: *Security weaknesses in Bluetooth*. LNCS, Vol. 2020, Springer-Verlag, 2001, pp. 176-191.

[JuM97] Jurisic A. and Menezes A.: *Elliptic Curves and Cryptography*. Dr. Dobb's Journal, April 1997.

[Kli05] Klima V.: *Finding MD5 Collisions − a Toy For a Notebook*. Cryptology ePrint Archive, Research Report 2005/075, March 5, 2005.

[KMP98] Knudsen L., Meier W., Preneel B., Rijmen V., and Verdoolaege S.: *Analysis Method for Alledged RC4*. Proceedings of the ASIACRYPT'98, 1998.

[Kob87] Koblitz N.: *Elliptic curve cryptosystems*. Mathematics of Computation, 1987, pp. 203-209.

[Koh78] Kohnfelder L.: *Towards a Practical Public-Key Cryptosystem*. M.I.T., Bachelor's Thesis, May 1978.

[Küg03] Kügler D.: *Man in the middle attacks on Bluetooth*. Financial Cryptography, LNCS, Vol. 2742, Springer-Verlag, 2003, pp. 149-161.

[LaL04] Laurie A. and Laurie B.: *The Bunker − Serious flaws in Bluetooth security lead to disclosure of personal data*. The Bunker, homepage, 2004. http://www.thebunker.net/resources/bluetooth (31.10.2008)

[LCA04] Levi A., Cetintas E., Aydos M., Koc C., and Caglayan M.: *Relay Attacks on Bluetooth Authentication and Solutions*. Computer and Information Sciences (ISCIS'2004), 19th International Symposium, Kemer-Antalya, Turkey, 2004.

[Lec04] LeCroy − Protocol Solutions Group: *CATC Scripting Language Reference Manual for LeCroy Bluetooth Analyzers − Manual Version 1.21*. LeCroy − Protocol Solutions Group, February 10, 2004. http://www.lecroy.com/tm/library/manuals/ProtocolAnalyzers/PDF/BTCSL_d121.pdf (31.10.2008)

[Lec07] LeCroy − Protocol Solutions Group: *LeCroy BTTracer/Trainer and Merlin II*. LeCroy − Protocol Solutions Group, homepage, 2007. http://www.lecroy.com/tm/products/ProtocolAnalyzers/bluetooth.asp?menuid=60 (31.10.2008)

[LHH04] Laurie A., Holtmann M., and Herfurt M.: *Hacking Bluetooth enabled mobile phones and beyond - Full Disclosure*. 21st Chaos Communication Congress, Berliner Congress Center, Berlin, Germany, December 27-29, 2004.

[LMV05] Lu Y., Meier W., and Vaudenay S.: *The Conditional Correlation Attack – A Practical Attack on Bluetooth Encryption*. Proceedings of the 25th Annual International Cryptology Conference (CRYPTO'2005), LNCS, Vol. 3621, Springer-Verlag, 2005, pp. 97-117.

[LuV04] Lu Y. and Vaudenay S.: *Cryptanalysis of Bluetooth Keystream Generator Two-level E0*. Proceedings of the 10th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'2004), LNCS, Vol. 3329, Springer-Verlag, 2004, pp. 483-499.

[LWW05] Lenstra A., Wang X., and Weger B.: *Colliding X.509 Certificates*. Cryptology ePrint Archive, Research Report 2005/067, March 2005.

[MaS01] Mantin I. and Shamir A.: *A Practical Attack on Broadcast RC4*. Proceedings of the Fast Software Encryption, LNCS, Vol. 2355, Springer-Verlag, 2001, pp. 152-164.

[Mil85] Miller V.: *Use of elliptic curves in cryptography*. Proceedings of the CRYPTO'85, Advances in Cryptology, LNCS, Vol. 218, Springer-Verlag, 1985, pp. 417-426.

[MiT98] Mister S. and Tavares S.: *Cryptanalysis of RC4-Like Ciphers*. Proceedings of the Workshop in Selected Areas of Cryptography (SAC'98), 1998.

[MKK98] Massey J., Khachatrian G., and Kuregian M.: *SAFER+*. Proceedings of the First Advanced Encryption Standard Candidate Conference, NIST, 1998.

[Mor02] Morrow R.: *Bluetooth − Operation and use*. New York, McGraw-Hill, 2002.

[Mos07] Moser M.: *Busting The Bluetooth Myth − Getting RAW Access*. Remote-exploit.org, Research Report, 2007. http://www.remote-exploit.org/research/busting_bluetooth_myth.pdf (31.10.2008)

[Net96] Netscape: *The SSL Protocol Version 3.0*. Netscape, November 18, 1996. http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt (31.10.2008)

[NFC08] NFC Forum: *NFC specifications*. Technical specifications, 2003-2008. http://www.nfc-forum.org (31.10.2008)

[NIS00] NIST: *Digital Signature Standard (DSS) − The Federal Information Processing Standards Publication 186-2*. NIST, January 27, 2000. http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf (31.10.2008)

[NIS01] NIST: *Advanced Encryption Standard (AES) − The Federal Information Processing Standards Publication 197*. NIST, November 26, 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf (31.10.2008)

[NIS02] NIST: *Secure Hash Standard − The Federal Information Processing Standards Publication 180-2*. NIST, August 1, 2002. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf (31.10.2008)

[NIS04] NIST: *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*. NIST, May 2004. http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf (31.10.2008)

[NIS91] NIST: *Proposed Federal Information Processing Standard for Digital Signature Standard (DSS)*. NIST, Federal Register, Vol. 56, No. 169, August 30, 1991, pp. 42980-42982.

[NIS94] NIST: *Digital Signature Standard (DSS) − The Federal Information Processing Standards Publication 186*. NIST, May 19, 1994. http://www.itl.nist.gov/fipspubs/fip186.htm (31.10.2008)

[NIS95] NIST: *Secure Hash Standard − The Federal Information Processing Standards Publication 180-1*. NIST, April 17, 1995. http://www.itl.nist.gov/fipspubs/fip180-1.htm (31.10.2008)

[NIS99] NIST: *Data Encryption Standard (DES) − The Federal Information Processing Standards Publication 46-3*. NIST, October 1999. http://www.cerberussystems.com/INFOSEC/stds/fip46-3.htm (31.10.2008)

[Nok02] Nokia: *User's Guide − Nokia 6310i*. Nokia, homepage, 2002. http://nds1.nokia.com/phones/files/guides/6310i_usersguide_en.pdf (31.10.2008)

[Nok03] Nokia: *User's Guide for the Wireless Headset (HDW-2)*. Nokia, homepage, 2003. http://nds1.nokia.com/phones/files/guides/Wireless_Headset_hdw2_en.pdf (31.10.2008)

[Nok05] Nokia: *Nokia Wireless Headset (HS-26W) User Guide*. Nokia, homepage, 2005. http://nds2.nokia.com/files/support/apac/phones/guides/HS26W_APAC_UG_en.pdf (31.10.2008)

[Nok08] Nokia Corporation: *Wibree*. Nokia Corporation, homepage, 2008. http://www.wibree.com (31.10.2008)

[Obe04] Oberritter A.: *btxml − mobile phone backup tool using Bluetooth*. Saftware, 2004. http://www.saftware.de/bluetooth/btxml.c (31.10.2008)

[Pfl03] Pfleeger C.: *Security in Computing*. 3rd Edition, Upper Saddle River, New Jersey, Prentice Hall, 2003.

[Reu05] Reuters: *Bluetooth Viruses*. Daily Wireless Online, newscopy, February 4, 2005. http://www.dailywireless.org/2005/02/04/bluetooth-viruses (31.10.2008)

[Riv92a] Rivest R.: *The RC4 Encryption Algorithm*. RSA Data Security, March 1992.

[Riv92b] Rivest R.: *RFC 1321 − The MD5 Message-Digest Algorithm*. MIT Laboratory for Computer Science and RSA Data Security, April 1992. http://www.faqs.org/ftp/rfc/pdf/rfc1321.txt.pdf (31.10.2008)

[Riv94] Rivest R.: *The RC5 Encryption Algorithm*. Proceedings of the Second International Workshop on Fast Software Encryption (FSE'1994), Leuven, Belgium, December 14-16, 1994.

[RRS98] Rivest R., Robshaw M., Sidney R., and Yin Y.: *The RC6 Block Cipher*. RSA Security, August 20, 1998. http://theory.lcs.mit.edu/~rivest/rc6.pdf (31.10.2008)

[RSA04] RSA Security: *A Cost-Based Security Analysis of Symmetric and Asymmetric Key* Lengths. RSA Laboratories, 2004. http://www.rsa.com/rsalabs/node.asp?id =2088 (31.10.2008)

[RSA05] RSA Security: *RSA-640 is factored*. RSA Laboratories, November 2, 2005. http://www.rsa.com/rsalabs/node.asp?id=2964 (31.10.2008)

[RSA78] Rivest R., Shamir A., and Adleman L.: *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*. Communications of the ACM, Vol. 21, No. 2, February 1978, pp. 120-126.

[RSA99] RSA Security: *RSA-155 is factored*. RSA Laboratories, August 22, 1999. http://www.rsa.com/rsalabs/node.asp?id=2098 (31.10.2008)

[Sap06] Sapronov K.: *Bluetooth, Bluetooth Security and New Year War-nibbling*. Viruslist.com, Research Report, 2006. http://www.viruslist.com/en/analysis?pubid=181198286 (31.10.2008)

[Sap07] Sapronov K.: *War-nibbling 2007*. Viruslist.com, Research Report, 2007. http://www.viruslist.com/en/analysis?pubid=204791928 (31.10.2008)

[Sch94a] Schneier B.: *Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)*. Proceedings of the Fast Software Encryption 1994, Cambridge Security Workshop, Springer-Verlag, 1994, pp. 191-204.

[Sch94b] Schneier B.: *The Blowfish Encryption Algorithm*. Dr. Dobb's Journal, Vol. 19, No. 4, April 1994, pp. 38-40.

[Sch96] Schneier B.: *Applied Cryptography − Protocols, Algorithms, and Source Code in C*. 2nd Edition, New York, Wiley & Sons, 1996.

[Sha06] Shandle J.: *University research aims at more secure Wi-Fi*. EE Times Online, newscopy, September 1, 2006. http://www.eetimes.com/news/latest/showArticle.jhtml?articleID=192501255 (31.10.2008)

[Sha49] Shannon C.: *Communication Theory of Secrecy Systems*. Bell Systems Technical Journal, No. 4, 1949.

[Shi08] Shishkin E.: *Namesys things*. Namesys, 2008. http://chichkin_i.zelnet.ru/namesys (31.10.2008)

[ShW05] Shaked Y. and Wool A.: *Cracking the Bluetooth PIN*. Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services (MobiSys'2005), Seattle, WA, June 6-8, 2005, pp. 39-50.

[Son05] Sony Ericsson: *Bluetooth Headset HBH-610 User Guide*. Sony Ericsson, homepage, 2005. http://www.sonyericsson.com/cws/download/1/110/552/1192974654/HBH-610_UG_R1a_Multilingual2.pdf (31.10.2008)

[SpB07a] Spill D. and Bittau A.: *BlueSniff − Eve meets Alice and Bluetooth*. Proceedings of the First USENIX Workshop on Offensive Technologies (WOOT'2007), Boston, MA, August 6, 2007. http://www.usenix.org/events/woot07/tech/full_papers/spill/spill.pdf (31.10.2008)

[SpB07b] Spill D. and Bittau A.: *BlueSniff*. University College London, 2007. http://www.cs.ucl.ac.uk/staff/a.bittau/gr-bluetooth.tar.gz (31.10.2008)

[Spe04] Spencer K.: *Taking a peek inside your mobile*. BBC News Online, newscopy, April 21, 2004. http://news.bbc.co.uk/nolpda/ukfs_news/hi/newsid_3642000/3642627.stm (31.10.2008)

[SSH08] SSH Communications Security: *SSH Communications Security Webpage*. SSH Communications Security, homepage, 2008. http://www.ssh.com (31.10.2008)

[Sta03] Stallings W.: *Cryptography and Network Security − Principles and Practice*. 3rd Edition, Upper Saddle River, New Jersey, Prentice Hall, 2003.

[Ste93] Stephenson P.: *Preventive Medicine*. LAN Magazine, November 1993.

[SVA07] Suomalainen J., Valkonen J., and Asokan N.: *Security Associations in Personal Networks − A Comparative Analysis*. Proceedings of the Fourth European Workshop on Security and Privacy in Ad-hoc and Sensor Networks (ESAS'2007), LNCS, vol. 4572, Springer-Verlag, 2007, pp. 43-57.

[Tan06] Tan A.: *Bluetooth gets high-speed boost*. CNET Networks, ZDNet Asia, newscopy, March 9, 2006. http://zdnetasia.com/toolkits/0,39047352,39346568-39094252p,00.htm (31.10.2008)

[Tec07] Tecom: *Tecom Bluetooth Printer Adapter*. Tecom, homepage, 2007. http://www.tecomproduct.com/BT3051.htm (27.11.2007)

[Tri06a] Trifinite.org: *BluePrinting*. Trifinite.org, homepage, 2006. http://trifinite.org/trifinite_stuff_blueprinting.html (31.10.2008)

[Tri06b] Trifinite.org: *BlueBug*. Trifinite.org, homepage, 2006. http://trifinite.org/trifinite_stuff_bluebug.html (31.10.2008)

[Tri06c] Trifinite.org: *Blooover*. Trifinite.org, homepage, 2006. http://trifinite.org/trifinite_stuff_blooover.html (31.10.2008)

[Tri06d] Trifinite.org: *Blooover II*. Trifinite.org, homepage, 2006. http://trifinite.org/trifinite_stuff_bloooverii.html (31.10.2008)

[UKA07] Uzun E., Karvonen K., and Asokan N.: *Usability Analysis of Secure Pairing Methods*. Nokia Research Center Technical Report NRC-TR-2007-002, 2007. http://research.nokia.com/tr/NRC-TR-2007-002.pdf (31.10.2008)

[Vel08] Velasco M.: *Marcos Velasco Security*. Marcos Velasco Security, homepage, 2008. http://www.velasco.com.br (31.10.2008)

[Wal00] Walker J.: *Unsafe at any key size − An analysis of the WEP encapsulation*. IEEE 802.11 Committee, Technical Report 00/362, October 27, 2000. http://www.drizzle.com/~aboba/IEEE/0-362.zip (31.10.2008)

[Wal05] Walko J.: *SIG updates Bluetooth cores roadmap*. EE Times Online, newscopy, November 25, 2005. http://www.eetimes.com/news/latest/showArticle.jhtml?articleID=174401758 (31.10.2008)

[WFL04] Wang X., Feng D., Lai X., and Yu H.: *Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD*. Cryptology ePrint Archive, Research Report 2004/199, August 17, 2004.

[Whi03] Whitehouse O.: *RedFang, Bluetooth Discovery Tool*. SecuriTeam, 2003. http://www.securiteam.com/tools/5JP0I1FAAE.html (31.10.2008)

[Whi04] Whitehouse O.: *@Stake − Where Security & Business Intersect*. CanSecWest/core04, Vancouver, 2004. http://cansecwest.com/csw04archive.html (31.10.2008)

[WYY05] Wang X., Yao A., and Yao F.: *New Collision Search for SHA-1*. CRYPTO'05, the Twenty-Fifth Annual International Cryptology Conference, Rump Session, August 16, 2005.

[Zig08] ZigBee Alliance: *Download ZigBee Technical Documents*. ZigBee Alliance, technical specifications, 2008. http://www.zigbee.org/en/spec_download/zigbee_downloads.asp (31.10.2008)

Kuopio University Publications H. Business and Information technology

H 1. Pasanen, Mika. In Search of Factors Affecting SME Performance: The Case of Eastern Finland.
2003. 338 p. Acad. Diss.

H 2. Leinonen, Paula. Automation of document structure transformations.
2004. 68 p. Acad. Diss.

H 3. Kaikkonen, Virpi. Essays on the entrepreneurial process in rural micro firms.
2005. 130 p. Acad. Diss.

H 4. Honkanen, Risto. Towards Optical Communication in Parallel Computing.
2006. 80 p. Acad. Diss.

H 5. Laukkanen, Tommi. Consumer Value Drivers in Electronic Banking.
2006. 115 p. Acad. Diss.

H 6. Mykkänen, Juha. Specification of reusable integration solutions in health information systems.
2006. 88 p. Acad. Diss.

H 7. Huovinen, Jari. Tapayrittäjyys – tilannetekijät toiminnan taustalla ja yrittäjäkokemuksen merkitys yritystoiminnassa.
2007. 277 p. Acad. Diss.

H 8. Päivinen, Niina. Scale-free Clustering: A Quest for the Hidden Knowledge.
2007. 57 p. Acad. Diss.

H 9. Koponen, Timo. Evaluation of maintenance processes in open source software projects through defect and version management systems.
2007. 92 p. Acad. Diss.

H 10. Hassinen, Marko. Studies in mobile security.
2007. 58 p. Acad. Diss.

H 11. Jäntti, Marko. Difficulties in managing software problems and defects.
2008. 61 p. Acad. Diss.

H 12. Tihula, Sanna. Management teams in managing succession: learning in the context of family-owned SMEs.
2008. 237 p. Acad. Diss.