IST-2000-25350 - SHAMAN

Deliverable Number	D03
Deliverable Title	Interim Report - Security Architecture for Future Mobile Terminals and Applications
Date of delivery	original 29 June 2001 revised 08 November 2001
Document Reference	SHA/DOC/SAG/WP2/D03/2.0
Contractual Delivery Date	31 May 2001
Actual Delivery Date	29 June 2001 revised 08 November 2001
Authors/Editor	Thomas Kuhn, Siemens AG (for list of authors see page 5, below)
Participant(s):	ATEA, Ericsson, Nokia, Siemens AG, T-Nova, Vodafone
Workpackage	WP2
Est. person months	14 MM
Security	Public
Nature	report
Version	Issue 1.0
Total number of pages	120

Abstract:

The document D03 is for the first milestone of the SHAMAN WP2. It provides an interim report on the investigations and conclusions concerning security architecture for future mobile terminals.

Keyword list:

Personal Area Network (PAN), Dynamically Configurable Terminals, Real-world Scenarios, Ad-hoc Security Research Papers, Bluetooth, MExE, IETF, MANET, SLP, ZEROCONF, UPnP, Java, Jini, WAP, i-mode, E-Speak, TESSA, MeT, ARIB MT-TA, Security Requirements, Initial Security Architecture

The work described has been supported in part by the European Commission through the IST Programme under Contract IST-2000-25350.

Executive Summary

The SHAMAN project addresses the protection and security required for users, information and services as the next generation of mobile communications moves into new fields. The project is conducting R&D on the security infrastructures for two major aspects of mobile telecoms following on from Releases 4 and 5 of 3GPP specifications for third generation mobile telecommunications.

The work concerns the provision of security for

global roaming and heterogeneous access networks;
dynamically reconfigurable distributed mobile terminal systems.

We are developing security architectures that will provide specifications of interfaces, protocols and mechanisms that are needed to deliver required levels of protection. We are also developing related supporting technologies based on public key infrastructure (WP3) and security modules based on smart cards (WP4).

This document provides a report on the work of SHAMAN Workpackage 2 (WP2). It is one of a pair of first technical deliverables of the SHAMAN project along with work on networking aspects of security for next generation mobile systems [D02 - Intermediate Report WP1: Results of Review, Requirements and Reference Architecture].

Focus of WP2

WP2 examines the security needs of future multi-function mobile terminals which will consist of dynamically configurable distributed components using mainly local wireless communications among themselves. These terminals will require secure applications environments to support their internal communications and their access to programs and information. In addition, the security requirements of WP2 will also affect the course of further work in WP3.

The main goal of this workpackage is to develop a unified security architecture covering both the future wireless, dynamically configured and distributed terminal and the service and application environment in which it is used. The focus is on security for the application and service layer with standard IP and WAP based services operating in MExE like environments evolving towards provision of *end-to-end* security between the user and the application.

Classification of D03

This document reports on the investigations to date and our interim assessment of the options and possibilities to be pursued; subsequent work will refine the concepts described here leading to project documents that provide detailed definitions and specifications.

The report marks the first milestone (6 months after SHAMAN kick-off) out of four of the SHAMAN workpackage 2. It is an intermediate report on the way to a security architecture for distributed mobile terminal systems. The second milestone (12 months after SHAMAN kick-off) will be an internal milestone where technical research and design resulting in architectural specification and requirements on supporting technologies will be done. This internal milestone is not documented by a report outside the SHAMAN working community. The third milestone (18 months after SHAMAN kick-off) will document a detailed technical specification of distributed mobile terminal system security. The last milestone (24 months after SHAMAN kick-off) will be a final report containing specification of terminal architecture and final technical specifications.

Work done

After giving a scope and overview from the past and present of distributed terminals a first definition of a PAN model is given. This basic definition consist of a terminal model where trusted and untrusted terminals exists. The model is expressed by a simple graphical notation with three basic entities. Furthermore a discussion is added where dynamical configurable terminals are explained. The dynamical configurable terminals can be viewed from serveral ways, one from the "software defined radio" world and one from the "application development" world. Several security risks are related to dynamical configurable terminals like the risk of viruses or the risk of quality losing of the terminal.

However, the advantages of the dynamical configurable terminal is emphasized. The first part of the document end with an enumeration of the current used devices and gives a vision of future distributed terminal components.

The basic definition of a terminal model and the introduced notation is used in the second part of the document to present real-world scenarios. The following scenarios are discussed:

- Using a single component
- Locally connected PAN
- "Local" interactions in public places
- Car scenarios

Each scenario is clearly explained and a threat model is discussed where possible threats of the scenarios are listed.

The third part built up a technical background with several security surveys where the focus is on an overview of research papers of *ad-hoc* networks; service lookup and distribution protocols like E-speak, Jini, SLP and UPnP; the ARIB MT-TA interface distributed terminal techniques; short-range wireless access technologies like Bluetooth; mobile communication techniques like WAP, i-mode and MeT; the dynamic routing techniques like MANET and mobile code execution techniques like MExE and TeSSa.

The definition of a PAN model and the technical background of the surveys are used in the forth part of the document to list the security requirements that the initial security architecture should include. The requirements are derived by first developing a role model for configurable, distributed terminals, and then generating the requirements that each role has. Furthermore the security requirements will be derived by considering the configurability and the dynamic aspects of the terminals together and not separately.

Based on the requirements the challenges and criteria for a future security architecture is given in the last part of the document. One main stress here is the necessary refinement and improvement of the used initial PAN model. This will be used as the foundation for the following stages of the work as outlined above.

Conclusion

This document initiates a broad discussion on the security architecture of future dynamically reconfigurable distributed mobile terminal systems and focuses on the requirements of such a security architecture. As the discussion of the last section "Challenges for the initial security architecture" has showed, our basic definition of a PAN model doesn't fulfil these requirements. Therefore a new, extended PAN reference model has to be developed. This will be the next major work of Workpackage 2.

Authors

- Partner: Vodafone Group (VOD)
  Name: Pubudu Chandrasiri
  Phone: +44 (0)1635 682986
  Fax: +44 (0) 1635 676147
  EMail: Pubudu.Chandrasiri@Vodafone.Com

- Partner: Ericsson Mobile Communications AB (ERIC)
  Name: Christian Gehrmann
  Phone: +46 46 232904
  Fax: +46 46 193455
  Email: Christian.Gehrmann@ecs.ericsson.se

- Partner: Siemens Atea (ATEA)
  Name: Stefan Goeman
  Phone: +32 14 25 30 20
  Fax: +32 14 22 29 94
  EMail: stefan.goeman@siemens.atea.be

- Partner: Vodafone Group (VOD)
  Name: Keith Howker
  Phone: +44 1635 682216
  Fax: +44 1635 234860
  EMail: keith.howker@vf.vodafone.co.uk

- Partner: Siemens AG (SAG)
  Name: Thomas Kuhn
  Phone: +49 89 636 52850
  Fax: +49 89 636 48000
  Email: thomas.kuhn@mchp.siemens.de

- Partner: Ericsson Radio Systems AB (ERIC)
  Name: András Méhes
  Phone: +46 8 585 30275
  EMail: andras.mehes@era.ericsson.se

- Partner: Nokia Research Center (NOKIA)
  Name: Kaisa Nyberg
  Phone: +358 40 7038169
  Fax: +358 9 4376 6850
  EMail: kaisa.nyberg@nokia.com

- Partner: T-Nova Deutsche Telekom Innovationsgesellschaft mbH (TNO)
  Name: Roland Schmitz
  Phone: +49 6151 832033
  Fax: +49 6151 834464
  EMail: roland.schmitz@t-systems.de

- Partner: Vodafone Group (VOD)
  Name: Timothy Wright
  Phone: +44 1635 676456
  Fax: +44 1635 31127
  Email: timothy.wright@vf.vodafone.co.uk

Abbreviations

3GPP	Third Generation Partnership Project
AAP	Address Allocation Protocol
ACMP	Address Assignment & Connection Management Protocol
ACO	Authenticated Ciphering Offset
AODV	Ad hoc On-Demand Distance Vector Routing Protocol
APDU	Application Protocol Data Units
API	Application Programming Interface
ARIB	Association of Radio Industries and Businesses
ARP	Address Resolution Protocol
ASN.1	Abstract Syntax Notation Number 1
AttrRqst	Attribute Request
BD_ADDR	Bluetooth Device Address
B-MAX	Business Mobile Access Exchange
BRP	Bordercast Resolution Protocol
BS	Base Station
BT	Bluetooth
CA	Certification Authority
CD	Compact Disc
CH	Channel
C-HTML	Compact HTML
CLDC	Connected Limited Device Configuration
C-MAX	Contents Mobile Access Exchange
COF	Ciphering Offset number
CRL	Certificate-Revocation List
CSP	Cryptography Service Provider
CUE	Consistent User Experience
DAAdvert	Directory Advertisement Message
DCF	Device Control Function
DCP	Device Control Protocol
DES	Data Encryption Standard
DH	Diffie-Hellman
DHCP	Dynamic Host Configuration Protocol
D-MAX	Database Mobile Access Exchange
DNS	Domain Name Service
DSA	Digital Signature Algorithm
DSR	Dynamic Source Routing Protocol
DVD	Digital Video/Versatile Disc
EC	Elliptic Curve
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve DSA
ESP	Encapsulated Security Payload
ETSI	European Telecommunications Standard Institute

FSR	Fisheye State Routing Protocol
GENA	Generic Event Notification Architecture
GFSK	Gaussian Frequency Shift Keying
GPS	Global Positioning System
GSM	Global System for Mobile communications
HMAC	Hashed Message Authentication Code
HTML	HyperText Markup Language
HTTP	Hypertext Transport Protocol
HTTPS	Secure HTTP
HTTPU	HTTP over UDP
HTTPMU	HTTP Multicast over UDP
IANA	Internet Assigned Numbers Authority
IARP	Intrazone Routing Protocol
IDEA	International Data Encryption Algorithm
IEEE	Institute of Electrical and Electronics Engineers
IERP	Interzone Routing Protocol
IETF	Internet Engineering Task Force
IID	Interface Identifiers
I-MAX	Interface Mobile Access Exchange
IMEP	Internet MANET Encapsulation Protocol
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
IPv4	IP Version 4
IPv6	IP Version 6
IrDA	Infrared Data Association
ISP	Information Service Provider
J2ME	Java 2 Micro Edition
JAAS	Java Authentication and Authorisation
JAIN	Java API for Integrated Networks
JAR	Java Archive
JCA	Java Cryptography Architecture
JDK	Java Developer's Kit
JNI	Java Native Interface
JVM	Java Virtual Machine
LAN	Local Area Network
LANMAR	Landmark Routing Protocol
LDAP	Lightweight Directory Access Protocol
LMP	Link Manager Protocol
MAC	Message Authentication Code
MADCAP	Multicast Address Dynamic Client Allocation Protocol
MANET	Mobile Ad hoc Network
MD	Mini Disc
MD5	Message-Digest Algorithm 5
ME	Mobile Equipment

MeT	Mobile electronic Transactions
MExE	Mobile Station Application Execution Environment
MIDP	Mobile Information Device Profile
M-MAX	Mail Mobile Access Exchange
MMI	Man Machine Interface
MP3	Moving Picture Experts Group Layer-3 Audio
M-PGW	Mobile Packet GateWay
MS	Mobile Station
M-SCP	Mobile Service Control Point
MT	Mobile Terminal
MT	Mobile Termination
MTF	Mobile Termination Function
N-MAX	Name Mobile Access Exchange
NID	Node Identifier
NSP	Network Service Provider
OEM	Original Equipment Manufacturer
OLSR	Optimized Link State Routing Protocol
OMF	Operation & Maintenance Function
OS	Operating System
OSI	Open System Interconnection
RMI	Remote Method Invocation
PAN	Personal Area Network
PC	Personal Computer
PCP	Packet Switched Connection Management Protocol
PDA	Personal Digital Assistant
PDC	Personal Digital Cellular
PDC-P	Packet switched PDC
PGP	Pretty-Good Privacy
PGW	Packet GateWay
PIN	Bluetooth passkey
PIN	Personal Identification Number
PKI	Public Key Infrastructure
PLMN	Public Land Mobile Network
PPM	Packet Processing Module
PTD	Personal Trusted Device
RAND	RANDom (or pseudorandom) number
RFC	Request for Comments
SAAdvert	Service Agent Advertisement Message
SCSI	Small Computer System Interface
SE	Security Element
SHA	Secure Hash Algorithm
SHAMAN	Security for Heterogeneous Access in Mobile Applications and Networks
SIM	Subscriber Identity Module
SLP	Service Location Protocol

SLPv2	Service Location Protocol Version 2
SM	Subscription Module
SMS	Short Message Service
SOAP	Simple Object Access Protocol
SPI	Security Parameter Index
SPKI	Simple Public Key Infrastructure
SRES	Response value
SrvAck	Service Acknowledgement
SrvDeReg	Service Deregister
SrvReg	Service Register
SrvRply	Service Reply
SvrRqst	Service Request
SrvTypeRply	Service Type Reply
SSDP	Simple Service Discovery Protocol
SSL	Secure Sockets Layer
SVRLOC	Service Location Protocol Group
SWIM	SIM/WIM
TA	Terminal Adapter
TAF	Terminal Adaptation Function
TBRPF	Topology Broadcast Based on Reverse-Path Forwarding
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TE	Terminal Equipment
TeSSA	Telecommunications Software Security Architecture
TFTP	Trivial File Transfer Protocol
TLP	Terminal Link Protocol
TLS	Transport Layer Security
TMF	Terminal Management Function
TORA	Temporally-Ordered Routing Algorithm
TV	Television
UDP	User Datagram Protocol
UE	User Equipment
UIM	User Identity Module
UIMF	User Identity Module Function
U-MAX	User Mobile Access Exchange
UPnP	Universal Plug and Play
URL	Universal Resource Locator
USB	Universal Serial Bus
USIM	Universal SIM
VCR	Videocassette Recorder
WAE	Wireless Application Environment
WAP	Wireless Application Protocol
WIM	Wireless Identity Module
WLAN	Wireless Local Area Network

W-MAX	Web Mobile Access Exchange
WML	Wireless Markup Language
WSP	Wireless Session Protocol
WTA	Wireless Telephony Applications
WTLS	Wireless Transport Layer Security
WTP	Wireless Transaction Protocol
WPKI	Wireless Public Key Infrastructure
WWW	World Wide Web
XML	Extented Markup Language
ZEROCONF	Zero Configuration Networking

Table of Contents

Executive Summary 3
Conclusion 4
Authors 5
Abbreviations 6
Table of Contents 11
1 Introduction 14
2 Scope and overview 15
2.1 Background 15
2.2 Personal Area Networking (PAN) 17
2.2.1 Definition 17
2.2.2 A simplified terminal model 18
2.3 Dynamically configurable terminals 19
2.4 Current and future devices 20
2.4.1 Current devices 20
2.4.2 A vision of future distributed terminal components 21
2.5 References 21
3 Real world-scenarios 23
3.1.1 Single component scenario (smart components) 23
3.1.2 Locally connected PAN 24
3.1.3 ”Local” interactions in public places 25
3.1.4 Car scenarios 26
4 Survey study 29
4.1 Ad hoc security research papers 29
4.1.1 Introduction 29
4.1.2 Security threats 30
4.1.3 Trust models 31
4.1.4 Key agreement 33
4.1.5 Future trends 34
4.1.6 References 35
4.2 Bluetooth 35
4.2.1 Description of the system 35
4.2.2 Technological environment 35
4.2.2 Security services 36
4.2.3 Technical security features 37
4.2.4 Bootstrapping security 38
4.2.5 Trust model 39
4.2.6 Strengths and weaknesses 40
4.2.7 Future trends 40
4.2.8 References 40
4.3 MExE 41
4.3.1 Description of the system 41
4.3.2 Technological environment 41
4.3.3 Security services 43
4.3.4 Technical security features 44
4.3.5 Bootstrapping security 47
4.3.6 Trust model 47
4.3.7 Strengths and weaknesses 48
4.3.8 Future trends 48
4.3.9 References 48
4.4 IETF Technologies (MANET, SLP, ZEROCONF) 48

4.4.1 Mobile Ad-hoc Networks (MANET) 48
4.4.2 Service Location Protocol (SVRLOC) 53
4.4.3 Zero Configuration Networking (ZEROCONF) 58
4.4.4 References 61
4.5 UPnP 62
4.5.1 Description of the system 62
4.5.2 Technological environment 64
4.5.3 Security services 66
4.5.4 Trust model 66
4.5.5 Strengths and weaknesses 66
4.5.6 Future trends 66
4.5.7 References 66
4.6 Java and Jini 66
4.6.1 Description of the system 66
4.6.2 Technological environment 70
4.6.3 Security services 70
4.6.4 Strengths and weaknesses 72
4.6.5 Future trends 73
4.6.6 References 74
4.7 WAP 74
4.7.1 Description of the system 74
4.7.2 Technological environment 75
4.7.3 Security services 76
4.7.4 Technical security features 77
4.7.5 Bootstrapping security 81
4.7.6 Trust model 81
4.7.7 Strengths and weaknesses 82
4.7.8 Future trends 82
4.7.9 References 82
4.8 I-mode 83
4.8.1 Description of the system 83
4.8.2 Technological Environment 83
4.8.3 Security services 85
4.8.4 Technical security features 85
4.8.5 Bootstrapping security 86
4.8.6 Trust model 86
4.8.7 Strengths and weaknesses 86
4.8.8 Future trends 87
4.8.9 References 87
4.9 E-speak 87
4.9.1 Description of the system 87
4.9.2 Technological Environment 87
4.9.3 Security Services 89
4.9.4 Technical Security Features 90
4.9.5 Strengths and Weaknesses 92
4.10 TeSSA 92
4.10.1 Description 92
4.10.2 Issues 94
4.10.3 References 94
4.11 MeT 94
4.11.1 Description of the system 94
4.11.2 Technological environment 97
4.11.3 Security services 98
4.11.4 Technical security features 99
4.11.5 Bootstrapping security 100

4.11.6 Trust model 101
4.11.7 Strengths and weaknesses 101
4.11.8 Future trends....101
4.11.9 References....101
4.12 ARIB MT-TA interface....101
4.12.1 System description....101
4.12.2 Technological environment....102
4.12.3 Security services....104
4.12.4 Threat analysis....104
4.12.5 Future trends....105
4.12.6 References....105
5 Security Requirements 106
5.1 Role model....106
5.1.1 PAN component user 106
5.1.2 PAN component owner 107
5.1.3 Application service provider (ASP) 107
5.1.4 Communications service provider (CSP) 107
5.1.5 Authorisation authority....107
5.1.6 PAN component manufacturer....107
5.1.7 PAN component administrator 108
5.1.8 PAN Manager 108
5.1.9 Intruder....108
5.2 Security requirements....108
5.2.1 User....108
5.2.2 PAN component owner 109
5.2.3 Application service provider....109
5.2.4 PAN Component Manufacturer 110
5.2.5 Communication Service Provider 110
5.2.6 Authorisation authority....110
5.2.7 PAN Manager 111
6 Challenges for the initial security architecture 112
6.1 Conclusions from survey study....112
6.1.1 Internal communications specifications and research 113
6.1.2 Specifications for external network and applications....113
6.1.3 Programming languages and general purpose service discovery applications....113
6.2 PAN reference model....114
6.3 Control and automation of dynamic configuration of distributed terminals....114
6.4 Internal communication security for PANs 115
6.4.1 Internal security requirements....116
6.4.2 Initialising security context for internal communication....117
6.5 Secure execution environment in a distributed terminal 118
6.6 Access Control....119

1 Introduction

The mission of Shaman workpackage 2 is to develop a new unified security architecture covering both the future wireless, dynamically configured and distributed terminal and the service and application environment in which it is used. The focus is on security for the application and service layer with standard IP and WAP based services operating in MExE like environments evolving towards provision of *end-to-end* security between the user and the application.

The work within workpackage 2 is organised in three phases.

The first phase includes the development of a functional model for the investigated terminal and service architecture and a corresponding high-level security requirement specification. One key fact to consider is that a single terminal identity will no longer exist when the terminal is a local distributed set of components which may be dynamically configured.

The security requirements will be used as the starting point of four parallel activities constituting the second phase. The areas are:

Control and automation of dynamic configuration of distributed terminals. This is related to control of *ad hoc* networks but differs in that all parts of a distributed terminal are under one owner's control. The goal is to find automatic or if this is not possible, semi-automatic control and configuration systems that are very easy to use.

Internal communications security for a distributed and dynamically configured terminal. The terminal internal requirements for authentication of different parts and protection of the "local" radio network will be investigated. The expected result includes a key management scheme and security protocols and mechanisms for authentication and protection of the radio communication.

Requirements for a secure execution environment in a distributed terminal. The secure execution environment should support secure download of applications as well as secure execution of standard client server applications. The requirement specification will be built on standard components. Development of application protocols, particularly WAP, to provide comprehensive control of actions on behalf of the mobile user in the terminal and in networked resources and services.

The third and final phase comprises a review and update of the requirements defined in phase one and the integration of the results from the second phase into a unified security architecture.

The report is structured in several sections. The second section deals with the lessons from the past and present and is entitled by 'Scope and Overview'. In this section we give a basic definition of a Personal Area Network(PAN), a simplified terminal model is described and dynamical configurable terminals are discussed. At the end of this section a state of the art view of current and future devices is given.

The definition of a PAN is used to construct several 'Real-world scenarios' in the third section. Each scenario has an added discussion about the threat model where possible threats are listed.

The next section (four) gives an overview of several security solutions to existing network technologies and protocols where an overview on research papers which are available in the ad-hoc field is given first. Other topics of this section are service lookup and distribution protocols like E-speak, Jini, SLP and UPnP; the ARIB MT-TA interface distributed terminals technique; the Bluetooth wireless communication technique; the mobile communication techniques like WAP, i-mode and MeT; the dynamical routing techniques MANET; and the mobile code execution techniques: MExE and TeSSa.

All of the above mentioned sections are used as a foundation for the definition of the security requirements for the security architecture which is the main scope of the fifth section.

The report ends with the last section 'Challenges for the initial security architecture' which gives a problem definition and an overview in order to construct an initial security architecture.

2 Scope and overview

In this section we define the overall scope for the work on "Security Architecture for Future Terminals and Applications". First, we give background information. Next, in Section 2.2, we define a Personal Area Network (PAN), and the terminal model we are working with. Section 2.3 treats dynamically configurable terminals. In Section 2.4 we describe current devices and give our view on what we expect from future devices.

2.1 Background

Portable computing devices with networking capabilities are becoming more and more important. Many people are dependent on a laptop for their everyday work and on having connectivity from the laptop to the "home" LAN at all locations. Personal Digital Assistants (PDAs) are used to organize people's work and they can also be used to perform networking duties like sending emails and connecting to the Web. Also, mobile phones are becoming more complex and can perform rather advanced computing and networking tasks. Laptops and PDAs are often dependent on an external unit, i.e., a mobile phone, in order to connect to the mobile network. This means that the laptop or PDA must be connected to the mobile phone using a cable or a local wireless communication link. Cable connections are cumbersome to use. Short-range infrared communication, IrDA [1], is less awkward, but the line-of-sight requirement limits its usefulness. Other wireless techniques, like Bluetooth [2], are about to replace the cables between personal communications devices. In the future we expect *short-range* wireless communication like Bluetooth to be widely used and available in most devices. We give a survey of the Bluetooth technology in section 4.2. This will make it possible to *divide* computing and networking functionality as well as applications between different terminal parts. People will use several communication devices. Several devices will be used to perform different tasks on behalf of the user and interaction between the devices will be necessary to perform a complete task. This we call a distributed terminal system.

Another important technical aspect of mobile terminals is that the user has the possibility to configure the terminal in several different ways. This requires dynamic software upgrades to the terminals. Currently it is possible for the user to download his favourite applications for personal computing devices like laptops and PDAs and this is soon to be possible also for mobile terminals. Also the manufacturer and mobile operators would like to have the possibility to upgrade and reconfigure the terminals. Hence, we foresee a situation where future terminals will be *dynamically reconfigurable* and *distributed*.

Personal computing devices interconnected with short-range wireless communication form a Personal Area Network (PAN). We view the PAN as the logical entity forming the terminal unit in a distributed terminal system. The PAN might consist of personal as well as non-personal devices. Hence, we do not restrict the PAN concept to only personal computing and communication devices. There are several different wireless techniques that can be used for local wireless communication. As examples, we mention Bluetooth [2], IEEE 802.15 [3](based on Bluetooth) and IEEE 802.11 [4](wireless LAN). Our communication model is independent of the particular wireless technology. The security requirements are also independent of the wireless technology. But, the built-in security is different for different technologies.

Today portable devices are mostly used separately, that is, their applications do not interact. However, personal devices can be configured to connect to each other for certain applications, i.e. a PAN can be created. When several different devices can interact in this manner and it is possible to automatically add or remove a device from the PAN, we call it a wireless *ad hoc* network. We give a survey of the current wireless ad hoc network security research in section 4.1. Furthermore, in section 4.4 a survey of ad hoc networking work within IETF is given. An ad hoc network is often described in terms of the ability to use multiple radio hops from source to destination. However, here we only consider small ad hoc networks where most devices are under one owner's control, or where a small number of devices under different people's control form the PAN.

In order to create efficient ad hoc networks, it must be possible for a certain personal device to connect to a particular other device (personal or non-personal), or to form a PAN whenever an application asks for it. Hence, applications must be able to find other devices and the "services" provided by other devices. For example, if a user with a PDA enters a coffee shop, he might be able to utilize communication services provided by the shop. These services might include printing documents, connecting to the Internet, or accessing local information services. How to spontaneously find and connect to services is an important part of the wireless ad hoc networking concept. We call this *service discovery*. Several different service discovery solutions exist and the security architecture needs to be adapted to the service discovery principles. In section 4 we give a survey of several different service discovery techniques. We treat UPnP (section 4.5), Java/Jini (section 4.6) and SVRLOC(section 4.4.2).

A distributed terminal can be "split" in many different ways. That is, even if we use a distributed terminal system not *all* functions in all devices are available over the PAN. The security requirement on the PAN will very much depend on which functions in the personal devices can be reached over the PAN. In order to increase user flexibility, we consider a distributed terminal concept where many core functions in mobile terminals are accessible over the PAN. This may include core terminal functions like access to the SIM/USIM card and connection management functions. As an example, we consider the Association of Radio Industries and Business (ARIB) interface description [5]. A terminal "split" in this way gives high security requirement. We give a survey of the MT-TA interface in section 4.12.

Dynamically connectable and reconfigurable terminal components are good from a usability point of view. However, this might be at the cost of decreased security. In order to put security requirements and provide security solutions, we need a way to measure the amount of trust we can put into a device or an application running on a device. The trust levels can be explicit or a part of the assumption in a trust model. The ability to access core mobile terminal functions like SIM cards and connection management functions requires a rather high level of trust on the device or application accessing these functions. We will develop a trust model for a distributed terminal system. In this work we will consider the work done in the Mobile electronic Transaction (MeT) initiative [6]. MeT introduced the concepts of "Personal Computing Device" and "Personal Trusted Device" applicable to the PAN concept. We give a survey of the MeT in section 4.11.

The ability to reconfigure and software upgrade a device implies a high security risk. Malicious software is a serious threat to the whole system. On the other hand, dynamic software upgrading gives higher flexibility and better user convenience. A security architecture for safe software downloading is defined within the 3GPP MExE work [7]. We will carefully analyse the MExE work when we study dynamic configuration for distributed terminals. Shortcomings and possible improvements of the MExE framework will be identified. Our final security architecture will contain solutions for secure software downloading in distributed terminal systems.

To summarise, our work will result in security architecture(s) for a distributed terminal system with the following main components:

1. Control and automation of dynamic configuration of distributed terminals, that is, configuration of the terminal element within a distributed terminal. Access to different functions in the terminal components will be restricted by security policies. Security mechanisms that implement the policy will be developed.
2. Internal communication security for PANs. The terminal internal requirements for authentication of different parts and protection of the "local" radio network as well as communication between personal devices and subscription modules will be investigated. We will provide solutions including a key management scheme and security protocols, and mechanisms for authentication and protection of the local radio communication.
3. Requirements for a secure execution environment in a distributed terminal. The secure execution environment should support secure download of applications as well as secure execution of

standard client server applications. A complete security architecture for secure execution in distributed terminal system including PKI and secure module usage will be developed.

2.2 Personal Area Networking (PAN)

Next we give a detailed description of the PAN concept. We define what we refer to as a PAN and define a simple terminal model. Using this model we describe different PAN configurations in section 3.

2.2.1 Definition

A PAN consists of a limited number of components, which have the ability to form networks and exchange information. The different components can be under one user's control, i.e. a personal computing component, or be controlled by different users or organizations. We are only considering short-range wireless networks and only relatively small PANs (on the order of 10 components). A component taking part in a PAN should be able to perform some of the following functions:

- Network discovery
- Network formation
- Address allocation
- Address resolution
- Name Resolution
- Bridging/routing

How these functions are provided determine how to establish and maintain a PAN. These functions might affect the security of the PAN. We will focus on networking security mechanisms and look for solutions as independent as possible of the other basic functions of the PAN.

One can think of many various PAN configurations and scenarios. We use a PAN model where all components forming the PAN have the same short-range radio interface. Some components in the PAN might also have a global network interface as illustrated in Figure 1, PANs can be formed by components with or without a global network interface.

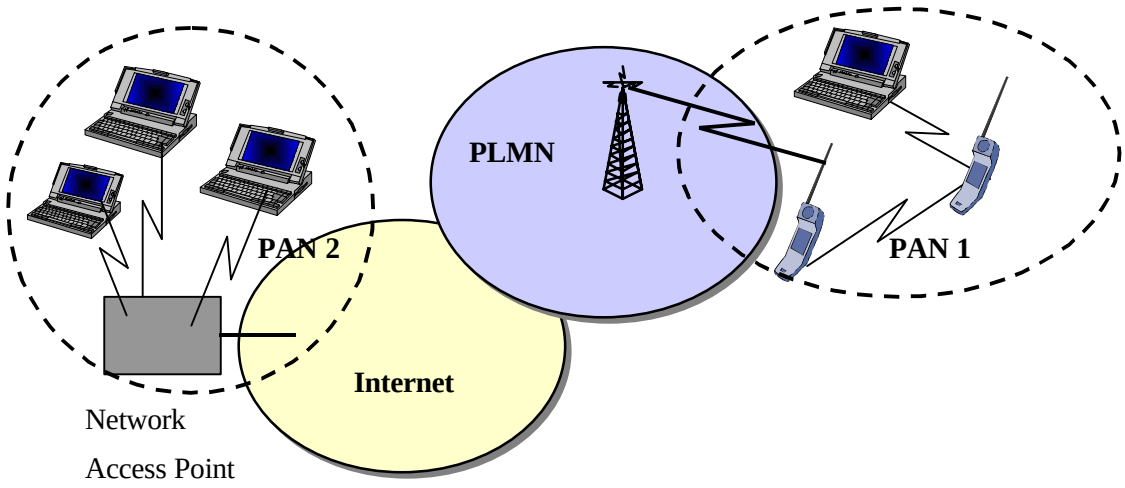


Figure 1. PAN examples

The network access point in Figure 1 acts as a bridge, proxy, or a router between a network (10baseT, GSM, etc) and the rest of the PAN.

We are only considering wireless communication in a licence-free spectrum. One example is the Bluetooth wireless technology. We do not assume that a supporting infrastructure (for key management or administration) is available to the PAN. However, we consider the usage scenario where at least one of the PAN components has a global network connection through a "global" or "local" interface.

2.2.2 A simplified terminal model

The security requirements will very much depend on who is the “owner” of the components in the PAN. One simplified model (from a security point of view) would be to only consider components that are controlled by the same owner all the time. However, in many practical situations an *unknown* component (i.e. components controlled by other users) may *temporarily* be used and connected to the PAN. Hence, we suggest including also unknown components in our scenarios.

In order to describe different PAN scenarios we introduce a simplified terminal model with three basic entities:

1. A component that supports one or more local communications bearers. This component might be used by the *controller* of the component or by a temporary user.
2. A logical entity called Subscription Module (SM), i.e. an entity that might be a part of another component or directly connected to a component, which contains unique information, needed for non-local network connections.
3. A component with a global network interface, that is a component that supports bearers that allow the component to communicate with other components beyond its immediate geographic locality.

In order to illustrate our scenarios, we use the symbols in Figure 2 for the entities in the list.

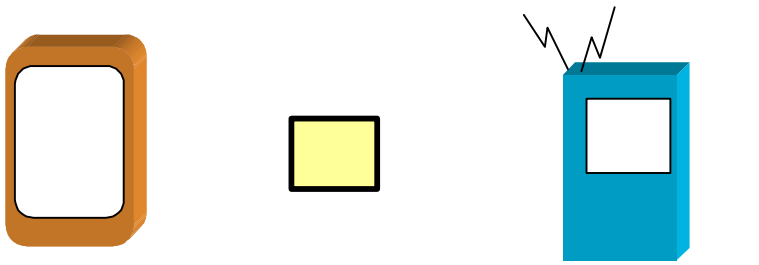


Figure 2. Simplified terminal model

We allow combination of these three basic entities. To indicate who is the *controller* of an entity we mark the components with capital letters. For example, a component currently controlled by user A is marked with the letter A. The Subscription Module can be (physically) attached to either of the other entities. Even if user A is the controller of a component, another user might temporarily use that component. There might not be one sole controller of the component, but different parties control different functions in the entity. In our scenario descriptions, we use a simplified model with only *one* possible controller per entity. Entities with multiple controllers can be illustrated by using more than one drawing where each drawing illustrates the situation for one set of functions.

The controller of a component is normally the *owner* of the component and the one that manages the settings. Some functions might not be accessible by the normal controller. For example a user might be the controller of a subscription module, but anyway she is not able to access all information or functions in this module. There may be an *additional* controller of the module, for example, the operator. At this stage, we do not consider whether an entity can be considered trusted for certain usage or not. But we introduce a more fine grain PAN role model in section 5.

Using this simplified model we can describe an arbitrary number of different PAN configurations. Some examples representing important real-world scenarios are given in Section 3.

2.3 Dynamically configurable terminals

The term *dynamically configurable terminals* can be understood in a number of ways. Those from the "software defined radio" world might understand it to mean that all but the baseband RF functionality of the terminal is flexible. This would mean that entire air interface handling software could be download to the phone. A slightly less radical understanding in this community might be that, though the spectrum band and modulation method might be constant, improvements to encoding and error checking could be downloaded to the terminal. Those from the application development world would understand *dynamically configurable terminals* as meaning terminals to which various high level applications could be download with all the basic bearer and transport layer functionality of the terminal remaining constant. Examples of applications that might be downloaded are diaries, call handling applications, or, very simply, a new ringtone. This workpackage will develop a security architecture that can be used across the entire range of downloadable functionality, from both baseband upgrades to high level applications.

Security has always been a big issue in discussion and development of configurable terminals. The risk of viruses that could propagate themselves over many terminals and thereby impede terminal behaviour or launch distributed denial of service attacks on the network, is often mentioned. Clearly, with the enormous cost that viruses have caused in the PC world, this risk is not to be underestimated. However, it should be noted that in many cases, viruses in the PC world do not spread because PC functionality is configurable as such, but because there are common applications for PC's that have errors that allow the operation of self-propagating programs. Such viruses could be said to be using the terminal functionality "illegitimately". Dynamically configurable terminals however, hold out the risk of applications that use terminal functionality legitimately and yet cause massive damage. Adequate authentication of the source of downloaded software and appropriate authorisation and access control for such software are seen as vital to meeting this threat.

A second risk with dynamically configurable terminals relates to the quality of terminal behaviour. At present, terminals undergo a considerable amount of quality assurance (QA) before they are sold to terminal distributors. Terminals with bugs are generally not released. Dynamically configurable terminals have the risk that new software will not have been as rigorously checked as the original terminal and may cause the terminal to malfunction. This risk is further increased if parties other than the original terminal manufacturer can download software affecting functionality further "down" into the terminal behaviour than that of high level applications. Again, there is a need for authentication and authorisation to meet this threat.

This brings us to another issue, that of the possibility of discontinuity between authorisation and responsibility. One party may, by virtue of the configuration of a terminal, be able to legitimately download software to a user's terminal. However, if that software causes the terminal to malfunction, the user may well go to a well-known body such as their manufacturer or operator instead of to the source of the software. One response to this might to only allow parties such as the operator or manufacturer to download software, but this might cut out many innovative and proficient third party application writers.

In some cases, such as the present MExE specification, the user is brought in to resolve such clashes of commercial interest. In some cases this may be appropriate, but in other cases it is asking the user to

make a decision they may not qualified to make. The history of mobile communications tells us that security that relies on the user is not good security.

However, in spite of all these security issues, there are clearly many advantages in having dynamically configurable terminals. Increased capability of terminals should mean increased terminal use and value for the benefit of operators and other service providers. The user benefits by being able to extend their initial investment in a terminal with further increases or changes in the terminal functionality.

Therefore within the term "security" we see the following issues for SHAMAN to tackle

- Authentication of software sources
- Flexible but meaningful authorisation structures
- Achieving openness without losing accountability
- Giving those parties most at risk most control
- Minimising user involvement whilst not cutting the user out

2.4 Current and future devices

2.4.1 Current devices

Nowadays, people use (and carry) a large variety of portable digital devices, which tend to be fairly sophisticated but mostly standalone and often incompatible. A non-exhaustive list includes mobile phones, laptops, PDAs, game consoles, GPS devices, MP3/MD/DVD/CD players/recorders, TVs, VCRs, still and video cameras, printers, scanners, calculators, books, diaries, translators, and smart watches. Most of these gadgets share some basic building blocks; most notably, a power supply, a computing core, and a user-interface of some complexity. Of course, the more specialized units also have their function-specific components.

Generally, the separation of high-level functionality entails a large redundancy of elementary blocks, such as processors, batteries, displays, buttons/keyboards, speakers, and so on. Due to the lack of interoperability, however, most of this redundancy is relatively useless. Although one might carry five batteries and three displays in total, if the laptop's screen gets damaged or the mobile phone's battery runs out there may be no way to download and view the latest movie trailers or an important company presentation in the middle of the woods. On the other hand, when operational each device can be used on its own without support from other devices.

The majority of these devices cannot interact and/or share data with each other directly, but most of them have standard interfaces to connect to a personal computer (PC). Connectivity and data transfer is facilitated mainly by cables (e.g. serial/parallel port, SCSI, USB, FireWire, Ethernet) or storage media (flash memory, floppy/zip disk, tape), but infrared and wireless technologies are increasingly available. In any case, current "personal networks" are usually wired and often centred around the least specialized and (thus?) most versatile of digital devices: the computer. Although, configuration of such networks can be cumbersome, once connected the topology is easy to trace and visualize, communication partners can be immediately identified, and a certain level of link security is afforded by the physical security of the connections.

Since most devices, except mainly computers and smart phones, are quite specialized, have relatively static internal configuration, and require user intervention to initiate connections/data transfer, their execution environments may be assumed secure. On the other hand, since generally one of the parties to any communication in these networks is a computer, we cannot assume that the distributed terminal does not possess an insecure operating system .

To summarize, current personal networks consist of highly autonomous and mostly static devices with a large number of redundant components and limited inter-device communication taking place mainly through a central unit; the [re]configuration of the network and the individual devices is inflexible (if

at all possible), but the topology is evident and both the connections and the execution environment may be regarded secure as long as the central unit (PC) can be trusted.

2.4.2 A vision of future distributed terminal components

In contrast to the current situation, future personal networks (or distributed terminals) are expected to be wireless, dynamic, easily [re]configurable, with extensive inter-device communication, and with component redundancy reduced to the mandatory support of the local network interface, configuration functionality, and some level of security in every device. Whether power distribution could and/or should also be solved in a wireless fashion or each device must be equipped with an independent (but modular) power source remains an interesting question. All devices are expected to be modular and easily [re]configurable. Today's laptops for instance may be split into several separate components, such as display, keyboard, speakers, microphone, and various storage options, leaving a PDA-like computing core accessing previously integrated functionality through the PAN.

At least two important classes of component devices are foreseen: specialized "dumb" units, which provide only one (or very few) functions to the distributed terminal, and versatile "smart" units, which form the distributed brains of the PAN. Examples of the first class are mainly input/output devices, i.e. user-interface components, storage options, printers/scanners. Representatives of the second class would include smart phones and various general-purpose computing units (PDA, car/home computer). Smart units are expected to control and configure the PAN, define its "identity", and communicate with the outside world. Downloaded applications would also effectively execute in smart units under tight process control with access rights to other devices based on their trust level/security credentials. To enable this, all units (dumb or smart) are expected to support a unified configuration and [access] control interface both at the per-device level and at the PAN level. For dumb devices this is in addition to their specialized functionality, and at a minimum it is likely to include elementary key management capability, software upgrade, and service advertisement. Depending on the configuration of the PAN and the intelligence of the units involved, some dumb devices may also be able to perform service discovery on their own and even request services from other devices unassisted.

An important assumption underlying the PAN model is that the local cloud of distributed terminal components communicates using a shared broadcast medium, possibly accessible to hostile devices. In this setting, cryptography is absolutely necessary to provide the usual security functions such as confidentiality, integrity, authentication and access control, but also secure configuration. Whether such configuration and control functionality is implemented using symmetric or public-key methods, per-service client lists, authorization certificates, or other means, will depend on the sophistication of the devices in question and their importance/sensitivity from the perspective of total system security. Irrespective of the implementation details, key-management (generation and distribution) becomes central in this model. In particular, secure storage of secret keys and secure key-distribution must be ensured. Since the main focus is a relatively small network of personal devices, initial keying by physical contact may be acceptable but this will be analysed within the project. Untrusted or foreign devices may be allowed to temporarily join the PAN and access selected services by establishing (and later removing) temporary keys for them. Smart units will need to have a secure operating system (execution environment) with mandatory access control and trusted path mechanisms. Similar measures may also be appropriate for dumb units.

2.5 References

[1] Infrared Data Association (IrDA), http://www.irda.org/

[2] Bluetooth Special Interest Group (SIG), http://www.bluetooth.com/

[3] IEEE 802.15 WPAN task Group 1, http://www.ieee802.org/15/pub/TG1.html

[4] IEEE Std. 802.11, 1999 Edition 8ISO/IEC 8802-5-1998], Standards for Local and Metropolitan Area Networks—Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications.

[5] "MT-TA Interface Description", Association of Radio Industries and Businesses (ARIB) , http://www.arib.or.jp/IMT-2000/V130Jan01/T12/0_T12coverV130.html.

[6] Mobile electronic Transaction (MeT) initiative, http://www.mobiletransaction.org/.

[7] 3GPP TS 23.057 "Mobile Station Application Execution Environment (MExE)".

3 Real world-scenarios

We investigate several basic scenarios that are within the scope of this work. This investigation is designed to reveal likely threats against distributed and configurable terminals. Security requirements to counter the threats will be given in section 5.

3.1.1 Single component scenario (smart components)

Although a single component is not strictly a distributed terminal, this use case is still valid for consideration, say, at the purchase of a "first component", as a minimal PAN, or as a failure mode of a proper distributed terminal. Other than the familiar requirements for standalone use, a smart PAN component should provide appropriate functionality for initiating secure communications with peer or slave components. For a single component to provide basic terminal functionality, it should be equipped with a global network interface and a subscription module. This simplest of cases is illustrated in Figure 3.

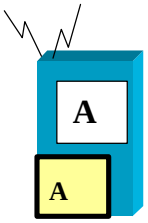


Figure 3. Travelling lightly

3.1.1.1 Threat Model for Figure 3

As the component has network access it is at risk from anything that it can access via its network connection. Hence, if the component has an execution environment, there is the threat of downloaded malicious executable content. This malicious content may affect the operation of this component alone, or, if the content can access the network connectivity functionality, the content could propagate itself to other components with network connectivity (i.e. the content is a "virus"). This malicious content may be downloaded at the instigation of the user, or may have been "pushed" to the component if push functionality is supported.

Malicious content can be downloaded even if the component does not have a standardised execution environment as such. If the component OS has known weaknesses, these can be exploited. Alternatively, the component may have bugs and be known to "lock up" or malfunction if certain content (e.g. certain characters or characters sets) is sent to the component.

This malicious content may have been written to be malicious, or may be legitimate content that has been tampered with in transit. However, code tampering in real time seems less likely than the code being malicious from the start, and being downloaded because the user has visited an uncontrolled site or because the code generator masquerades as a legitimate site.

There are further threats from components not seen in the scenario which do not take an active role in the communication the component has with the network, that is, the threat of passive eavesdropping on the component's network communications.

The user might physically control all parts of his terminal, but he is not allowed to tamper with some core network connection functions or subscription module functions. These functions must then be protected from the manufacturer and operator point of view against tampering by the user A.

3.1.2 Locally connected PAN

At times global connectivity is not necessary or it might not even be desirable (e.g. for personal privacy). Although, there may be components in the PAN with a global network interface and/or a subscription module, it should be possible to deactivate these capabilities without affecting local connectivity and functionality of the PAN. Figure 4 shows a schematic representation of a PAN with only local communication.

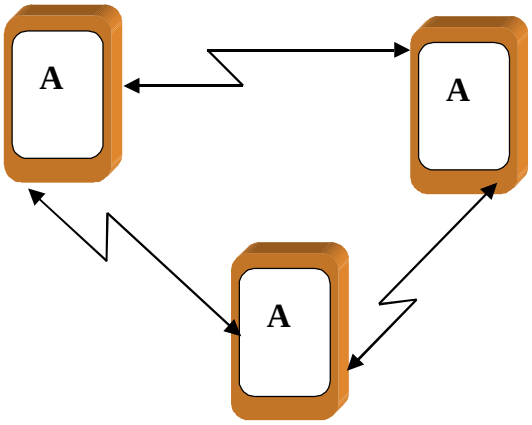


Figure 4. PAN game at home

3.1.2.1 Threat Model for Figure 4

In this scenario apparently only trusted components are involved since the same user controls them all. In such a scenario, where only apparently trusted components are involved, the users of the components, and the components themselves, may assign wide access privileges and high authorisation to the components in the network. Even if the same user controls a unit it might be infected by virus or malicious software and this will be a big potential threat against the rest of the components that A controls. If A can be sure that the component she controls are trustworthy, there are still threats relating to a malicious component masquerading as one of the components controlled by A. The malicious component may then perpetrate a number of attacks against the other members of the network.

The range of attacks will be determined by the level of access control and authorisation given to the different components that A controls. For instance, if one component can read any data on another component without requiring user permission, this functionality will be open to the malicious component with no possibility of user prevention. Similarly if being trusted means that one device will accept any content from another component and assign high access control to that content and/or without scanning this content for viruses, the malicious component can spread malicious content through the network.

There is also the threat of passive eavesdropping on the inter-component communications.

3.1.3 "Local" interactions in public places

PANs using the same short-range wireless technology (such as Bluetooth) for local communication can interact with each other and/or public services when within range. This does not require a global network interface, but may need an identity module for other purposes. Figure 5 shows two locally communicating PANs, whereas Figure 6 depicts a service access scenario. In both cases, a single smart component is assumed to be the link between the PAN and the outside world.

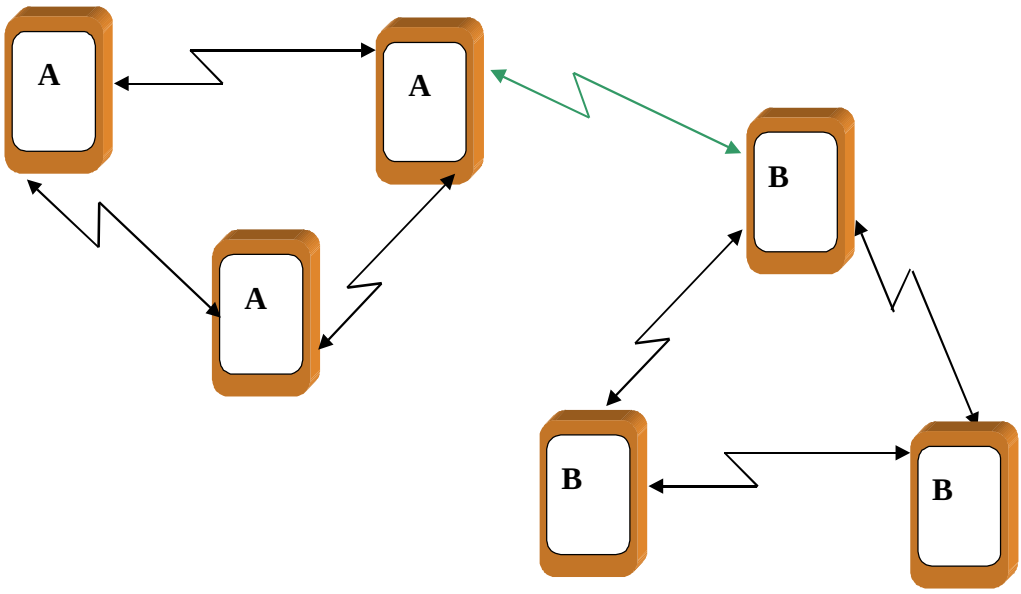


Figure 5. Friends sharing music (user A and B)

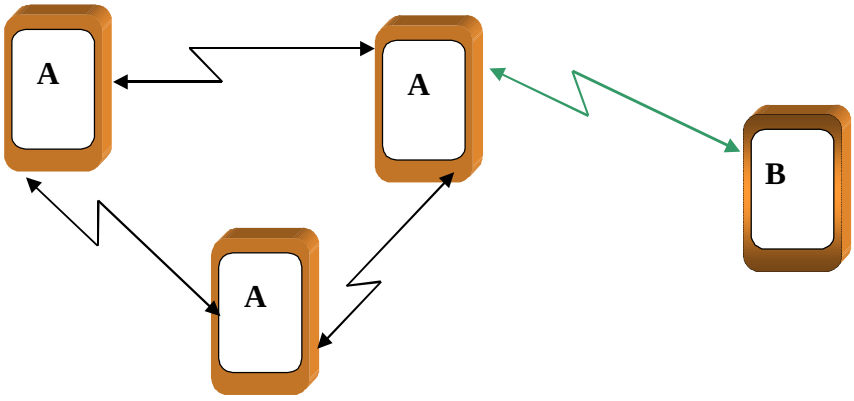


Figure 6. A using a public printer ran by service provider B

3.1.3.1 Threat Model for Figure 5 and Figure 6

These scenarios are very similar to a trusted local LAN connection to an untrusted network or the Internet. A user uses one of his components to connect to a friend or external network in order to gain

access to some service. It might not be possible for the user to be sure that what she receives over the network or the other party is trusted and not malicious content. If the component connecting to the external network is compromised or fails to filter out malicious content obtained from the external network, it is likely that the other components under the control of the same user will trust the filtering of the link (smart) component and so also be "infected". This is especially likely if the components are not all peers, but rather the link (smart) component assumes the role of a "master" and the other components are slaves. There could therefore be a spread of these viruses between the components controlled by one user. These viruses in turn could cause a number of undesirable events including:

- monitoring of the communications in the trusted network,
- damaging or modifying data shared between the entities in the trusted network (security within the trusted network will tend to be relaxed since all the components are trusted).

A potential threat is that the other party might deny service. The other party might also impersonate or modify content that is transmitted through the entity that she is controlling.

3.1.4 Car scenarios

These scenarios assume that the car is equipped with a powerful global network interface, which is preferred over other (non-local) network interfaces that may also be present in the PAN. At the same time, perhaps as a safeguard, the car's built-in network interface does not have its own subscription module; thus, it can only be operated if another unit with a subscription module is available.

Figure 7 shows the owner of the car using the built-in interface with a subscription module in another unit of her PAN. (Possible global network interfaces of other components in the PAN are not shown.)

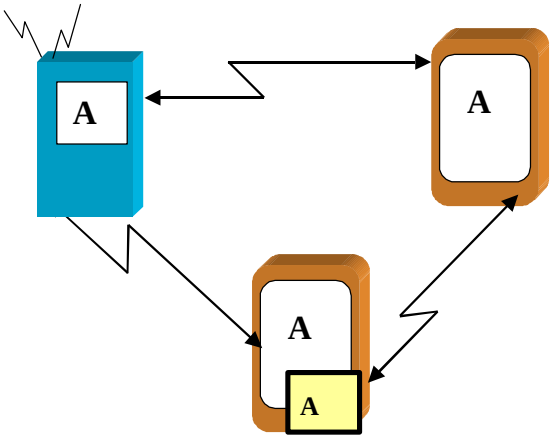


Figure 7. The car-owner connecting through the built-in network interface

Another scenario involving several components controlled by different users concerns a passenger accessing the car's global network interface using his own subscription. Of course, this should only be allowed with the owner's permission, which does imply a certain level of trust between A and B. Figure 8 illustrates the situation.

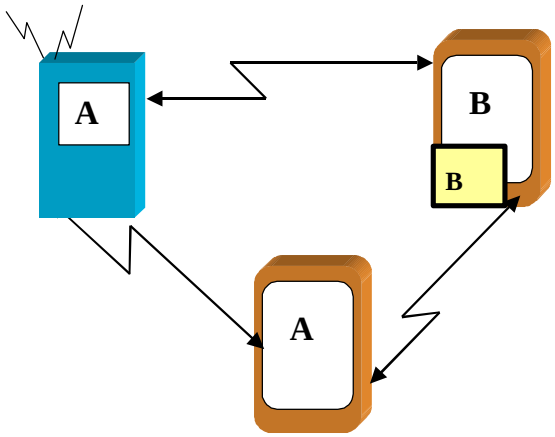


Figure 8. A passenger using the car's built-in interface

3.1.4.1 Threat Model for Figure 7

Possible attacks that could originate within the local network have already been identified in the threat analysis for Figure 4. Also the potential threats associated with a global network connection as identified under Figure 3 will also be applicable.

The presence of the global network interface will lead to additional threats compared to the scenarios in Figure 3 and Figure 4. In order to get access to the network the component with the network interface needs to have access to a subscription module. This module now resides in *another* physical component and the communication with the subscription module must take place over an open interface. This situation means that if no proper authentication and encryption is performed between the *modem* unit and the subscription module, the subscription module itself as well as the information transmitted from the module would be vulnerable to attacks. The situation where core functions in the subscription module are available from an external interface makes the subscription module much more vulnerable. If a trusted component with network interfaces is compromised or stolen also the subscription module will be threaten.

3.1.4.2 Threat Model for Figure 8

The threat situation is the same as for Figure 7 both regarding the internal network and the external network interface.

Since the subscription module now needs to be accessed by a unit not controlled by B this situation is a bit more complex than the one in Figure 7. In order to give access to the subscription module to an unknown or only partly known unit, the controller of the subscription module must be sure that she can control exactly how long time and to what extend the unknown component should be able to access the subscription module. If access from the component with the network interface is unrestricted there is a risk that the subscription module will be used by user A or hostile third parties without the control of user B.

In this model all content downloaded from the global network by component B will be via the component controlled by user A. This will increase the risks associated from malicious code been transmitted from component A to component B with the original content. All unencrypted network connections are insecure from user B's point of view. If the connection is not encrypted it is possible for the locally untrusted component to send malicious applications to the network via the established connection. . It is also possible for component A with the network interface to forward a copy of the data stream received over the global network interface to a hostile party without the user B ever

noticing about it. A hostile third party might use that for analysis or even possible for a replay attack. Denial of service attacks against user B will be easy to perform by A.

4 Survey study

The purpose of this section is to present a survey of existing work on various aspects of personal area networks, ad hoc networks, distributed terminals and configurable terminals. The work covered by this survey includes technical solutions of different nature, and also theoretical work and implementation tools. For each topic, the main focus is on security related aspects: trust model, security services and technical security solutions, and an attempt has been made to analyse main strengths and weaknesses from the security point of view.

The variety of previous work discussed by this survey can be divided roughly in the following three categories:

- Internal communications security specifications and research
- Specifications for external network and applications security requirements
- Programming languages and general purpose service discovery applications

In the first category, various technologies for ad hoc networking and distributed terminals are presented. Bluetooth radio technology, discussed in section 4.2, provides a comprehensive solution for short range wireless connectivity and link layer security. The Internet specifications for ad hoc networking: MANET and ZEROCONF, presented in section 4.4, are designed to run on standard Internet protocols, and also, rely mainly on existing Internet security solutions. The ARIB MT-TA (section 4.12) defines an interface for external terminal equipment. Previous research work in ad hoc security is mainly concentrated in trust models and ad hoc key agreement methods and is discussed in section 4.1.

The second category contains applications provided by an external network that are intended for use by an distributed or configurable terminal. The usage of WAP protocol in section 4.7 is not limited to distributed terminals, but is included in this survey as an example of security requirements such an application may have. The specific requirements due to distributed terminals become more clearly exposed within the framework of Personal Environment as defined in MeT in section 4.11. The Japanese I-mode application is discussed in section 4.8.

The second category also includes the special applications for downloading executables and configuration data to remote terminals. They are discussed in sections 4.3 (MExE) and 4.10 (TeSSA).

In the third category, the security aspects of Java are considered from the point of view of ad hoc networking in section 4.6. This section also includes Jini which is a service application built on top of Java. The remaining three general purpose service discovery applications are E-speak in section 4.9, the Internet based SVRLOC, discussed in section 4.4.2 together with the other Internet solutions, and the proprietary Microsoft solution for ad hoc networking, the UPnP protocol, presented in section 4.5.

4.1 Ad hoc security research papers

4.1.1 Introduction

In this section, we give a survey of research on *wireless ad hoc* networks (from now called ad hoc networks). An ad hoc network is a network formed without any central administration. The ad hoc network consists of independent mobile nodes that each use their own wireless interface to send data. Hence, in an ad hoc network there is no fixed infrastructure like base stations or switching nodes. A PAN can be considered a personal ad hoc network. Our PAN scenarios only include a small number of ad hoc nodes and in such network routing of data packets are not so hard to achieve. Most ad hoc network research is target towards routing problems in ad hoc networks. Except the security problems related directly to the path finding and routing signalling, the security problems one faces in ad hoc networks are to a large extend independent of the size of the network and if advanced routing mechanisms are needed or not.

Ad hoc security is not a large research field. One can even question whether this is a research field of its own or the security problems related to ad hoc networks can be solved by state-of-the-art security solutions. However, ad hoc networking is certainly a research field and there are some security aspects of ad hoc network that are different from other communication networks. Some of these issues are treated in a few research papers. We summarize the main results from these papers in this survey. In our summary, we concentrate on the aspects applicable to our PAN model. Three different aspects of security in ad hoc networks are covered: security threats, trust models, and key agreement.

4.1.2 Security threats

Expected security properties in ad hoc networks are discussed by Stajano and Anderson in [1] and also by Zhou and Haas in [2]. Stajano and Anderson are considering three different security properties: *confidentiality*, *integrity (authenticity)* and *availability*. Zhou and Haas use a similar list but treat authenticity separately from integrity and add *non-repudiation* to the list. In both papers threats against ad hoc networks are discussed in connection to the security properties. We reuse that structure here and analyse different threats related to the expected security properties.

4.1.2.1 Availability

Availability means that the services expected from the network or the network nodes are available independent of denial of service attacks. It should not be impossible for an adversary to disable a node or a service given by a node in the ad hoc network.

Denial of service attacks on wireless ad hoc networks can be of many different types. Jamming attacks can be performed against the radio channel. If ad hoc routing is performed, a false node participating in the network can easily disrupt the routing. An adversary node can launch a battery exhaustion attack or some other type of node failure attack.

In general, it is hard to protect an ad hoc node against all possible types of denial of service attacks. Attacks targeted at the radio interface or at basic functions like the power system (i.e. energy exhaustion attack) are not possible to avoid by cryptographic techniques. Introducing priority rules on the task performed by an ad hoc node gives some level of protection against battery exhaustion [1].

Denial of service attacks on routing protocols can be avoided by proper integrity protection of the routing information [2] and by using diversity coding [3],i.e., using multiple paths for the routing messages.

4.1.2.2 Authenticity

Authenticity assures the identity of the communicating nodes in the ad hoc network. Without authenticity it is difficult to provide availability, integrity and confidentiality.

The main difference regarding authenticity in ad hoc networks compared to other wireless networks, is the *absence of an online server*. Authenticity is often tightly coupled to key agreement and this is why key agreement is very hard in ad hoc networks (see section 4.1.4). If authenticity not is provided, an adversary could masquerade as a legitimate node and gain unauthorised access to resources or confidential information. Furthermore, secure key agreement is not possible, and integrity and confidentiality cannot be provided.

4.1.2.3 Integrity

Integrity ensures that information provided by a node is never corrupted or false. Two aspects are relevant for ad hoc networks: message corruption during transfer, and tampering of nodes. A node that is left unattended for a long time might have been messed with internally, and the information provided by that node is not correct anymore.

Protection during message transfer is provided by message authentication codes, and if the key agreement problem is solved this can be done in exactly the same way as for traditional networks. Tampering can, to some extent, be avoided by making the node tamper-proof. However, this is hard or even almost impossible to achieve in some circumstances. For example, in [1] a thermometer is given

as an ad hoc node example. In this case, an adversary might not only tamper with the thermometer itself, but might attack the sensor of the thermometer, making it give false temperature information to the surrounding ad hoc nodes.

4.1.2.4 Confidentiality

Confidentiality ensures that no information is disclosed to unauthorised entities. Messages travelling between ad hoc nodes must be protected. If the key agreement problem is solved, confidentiality during transmission can be provided in exactly the same way as for traditional networks. Confidentiality of stored information on network nodes is harder to solve. In the end, this becomes a question of tamper resistance of the node.

4.1.2.5 Non-repudiation

Non-repudiation ensures that the origin of a message cannot deny having sent the message. This might be an issue for some ad hoc routing protocols. Non-repudiation for ad hoc networks is mentioned in [2]. However, we do not see any non-repudiation threats that are unique or especially hard to solve for ad hoc networks compared to other wireless or wired networks.

4.1.3 Trust models

The trust model is fundamental for the security of a communication system. The trust model is the foundation on which key agreement, authentication and access control mechanisms are built. The dynamic nature of an ad hoc network makes these hard to model. Here we give a summary of the trust model used in a couple of research papers.

4.1.3.1 Robustness using threshold cryptography

In [2], it is noted that in ad hoc networks there is no central node that can be considered trusted, and that the trust relationship among nodes changes. Hence, they propose a key management scheme where distribution of trust is accomplished using threshold cryptography [5]. In this model, at least t+1 of n servers must be compromised in order for an adversary to succeed in providing a correct false signature. The idea is that a set of servers replaces the Certificate Authority role for key management. However, introducing threshold cryptography into ad hoc network key management only makes the system more *robust* against inside attacks against the key management, it does not describe which nodes should take the server roles and why they should be considered trusted or trusted with high probability.

4.1.3.2 The resurrecting duckling

A trust model for ad hoc networks called the "the resurrecting duckling" is introduced in [1]. Stajano further develops this model in [4]. Below we give a short overview of the "resurrecting duckling" model.

In the "resurrecting duckling" model, it is assumed that the ad hoc nodes consist of common user equipment that is able to communicate with each other. Typical such equipment is mobile phones, PDAs, laptops, DVD players, hi-fis, and televisions. It may also be units controlling other devices in the home like the heating system, lights and curtains, locks and burglar alarms. For these devices it is important that it is possible to set up secure associations between the devices. For example, this can be between a controller and the controlled unit. But, these associations must also be transient. User equipment can be resold or be given away, and then it must be possible to delete the old security associations and create new ones.

The "resurrecting duckling" is a metaphor from biology; it refers to the well-known *imprinting* phenomena of some species. The model is that, at manufacture, an ad hoc device is assumed to be "newborn" and to have no relation to other device. The device is associated (imprinted) with the first other ad hoc device that gives it a secret key. This device will then be the only one that has a secure relation with this device, except devices that the new device will make associations with on its own (i.e., other "newborn" devices). The security relation will last as long as the original controlling device wants. If needed (when the device changes owner or is resold) the controlling device can put the

controlled device back into the "newborn" state. As a security backup, it is possible for a manufacturer to have a master password that can be used to regain control over a device that does not anymore have any contact with the legitimate controlling unit.

It was noted in [4], that the original "resurrecting duckling" trusts model is limited in the sense that it only takes into account master-slave relationships between devices. Hence, Stajano extended the model to one that is a better fit for peer relationships. In his extended model, it is assumed that the controlling device at the "imprinting" stage not only transfers a secret key to the "newborn" device, but together with the key a complete *security policy* that tells the "newborn" device which trust relationships it shall have to other ad hoc nodes. This policy might include that the device can even be controlled to some or large extent by devices other than the original controlling device. The policy can be changed at any time to reflect a new communication situation. The policy tells which credentials a principal should exhibit in order to persuade a unit to perform particular actions.

The "resurrecting duckling" model is a useful trust model for many ad hoc network situations. However, the original trust model is too limited due to the master-slave relationship constraint. This constraint is removed in the extended model. In the extended model, trust relations are determined by policy rules. However, it will be hard to write policy rules that will fit into real ad hoc networking scenarios. In a situation where a new ad hoc relation between "imprinted" devices needs to be set up and the policies of the nodes do not allow this, interaction with the controlling units is necessary in order to change the policies. But, the controlling unit or units might not be within communication range.

4.1.3.3 Simplified trust model and trust distribution

In [6], trust in an ad hoc setting is discussed. The background is ad hoc Jini services, but the analysis is made for ad hoc networks in general. The authors conclude that trust can be modelled in several different ways and that trust is in general non-symmetric, non-transitive (i.e. if Alice trust Bob and Bob trusts Carol this does not mean that Alice trusts Carol) and that there are often different levels of trust. If trust in an ad hoc setting is considered solely from the point of view of creating initial security context between the involved ad hoc nodes, a simplified trust model can be used. In the simplified model the authors assume that trust relations among ad hoc nodes are symmetric and transitive. Furthermore, it is assumed that only *one* level of trust exists, i.e., an ad hoc node is trusted or non-trusted (i.e. trusted for setting up secure connections).

A public key can represent an entity or a relation, and in PKIs the public keys might represent trust levels and relations. In [6], trust based on public keys is considered. The simplified trust model implies that a public key is either trusted or non-trusted. Given the simplified trust model and the assumption of trust represented by public keys, the authors show that trust can easily be distributed within an ad hoc network. Trust distribution makes it possible to reduce the number of manual interactions (compared with the "resurrecting duckling" model) needed to set up secure ad hoc connections. The problem considered is a group of ad hoc nodes that gathers and would like to set up secure connection among themselves. This can for example be a meeting room scenario where each person in the meeting room carries a communication unit, for example a laptop or a PDA. The communication units compose the nodes in the ad hoc network, and in order to communicate securely among all the participants, they need to set up a secure connection among all the nodes. The trust distribution is a basic two step process:

- Taken an arbitrary group of ad hoc nodes, first determine the equivalence classes created by existing trust relationships. We call these equivalence classes trust groups. Each of these trust groups are fully connected, i.e., all nodes within a trust group trust all nodes in the group.
- If there are more than one-trust groups, manually create a new trust connection between arbitrary nodes in distinct groups. Each time such a new manual trust relationship is created, the groups merge due to our assumption about the nature of trust relationships.

Consider Figure 9 below. Here we illustrate an ad hoc network with about a dozen of nodes. The initial trust relations are represented with dashed lines. Thus, for example, the node L in the figure

does not have any trust relation with the other nodes in the network. Furthermore, the nodes H and M form an isolated "island" without any trust relations to the rest of the nodes in the group.

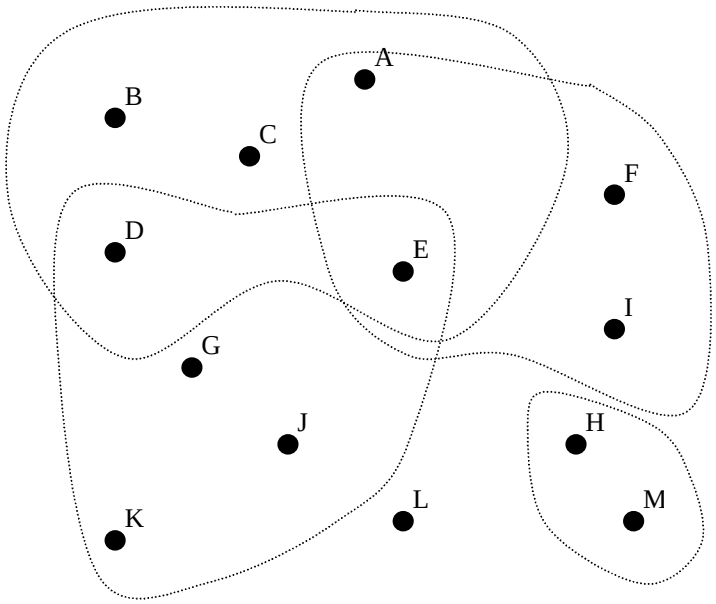


Figure 9: Ad-hoc networks

If the trust distribution principle is applied to this network, first a trust relation is created between the node L and some other node except H or M. This can be done by manually (if accepted). Next, in the same way, a trust relation is created manually between node H (or M) and one of the other nodes. Now, all nodes are fully connected and it is possible to create a trust relation that comprises all nodes in the ad hoc network. The trust distribution can be done in a fairly straightforward manner using public key signatures.

4.1.4 Key agreement

Trust in a communication system means that a party that possesses a shared secret or a private key (corresponding to a particular public key) is trusted to set up a secure connection, access some information, etc. If a particular key really should be trusted, it must be transferred in a way that does not compromise the security of the system. Key agreement between trusted parties based on strong shared secrets or public keys is not hard to solve. However, in an ad hoc networking scenario, with a lack of a supporting online server, it might not be possible to assume that the trusting parties share a common strong secret or share a common trusted third party certificate authority.

According to the resurrecting duckling model [1], the "imprinting" phase is equal to a shared key agreement. Key agreement needs only be done when activating the ad hoc device and that is for personal devices not too often. The authors suggest two key agreement procedures:

- Physical contact
- Diffie-Hellman protocol with check-values

Physical contact here means that with an electrical contact the bits constituting the "imprinting" are transferred in plaintext. According to the second proposal, a Diffie-Hellman exchange is performed and after that the hash value of the result is calculated in both devices. These values are displayed to both users. The drawback of this approach is that it might be cumbersome to verify the hash value manually and not all devices will have a display.

In [7], Asokan and Ginzboorg treat key agreement in ad hoc networks. They consider a small group of people coming together for an ad hoc meeting. The goal is to share information securely so that no one outside the room can eavesdrop on their session. No shared public key infrastructure or strong pre-shared secret is assumed. However, they find it feasible to assume that a short password is possible to securely distribute among the participant. Hence, what they are investigating is password based key agreement schemes for ad hoc groups. Password based key-agreement was introduced by Bellovin and Merrit in [8]. This is the basis for the protocols suggested in [7] where an introduction to password-based key exchange is given, and it is shown how it can be generalised to groups. A general protocol for password-based key exchange in ad hoc groups is given. Due to some security problems with their general proposal they then continue with describing a protocol based on a Diffie-Hellman key exchange. This one we present here. In the following, denote by $M_1, M_2, ..., M_n$ the n different players who all share a password P. At the end of the protocol, all share a session key $K = g^{S_1,S_2,...,S_n}$ where S_i is the random share contributed by M_i. The group Diffie-Hellman protocol works as follows:

1. $M_i \rightarrow M_{i+1}$: $g^{S_1,S_2,...,S_i}$, i = 1, ...,n-2, in sequence
2. M_{n-1} ® ALL: $\pi = g^{S_1,S_2,...,S_{n-1}}$, broadcast
3. $M_i \rightarrow M_n$: $P(c_i)$, i = 1, ..., n - 1, in parallel, where $c_i = p^{\hat{S}_i / S_i}$ and $\hat{S}_i$ is a blinding factor that is randomly chosen by M_i.
4. $M_n \rightarrow M_i$: $(c_i)^{S_n}$, i = 1, ..., n -1, in parallel
5. M_i ® ALL: M_i, $K(M_i,H(M_1, M_2, ..., M_n))$, for some i, broadcast

Stage 1. Player M_1 computes g^{S_1} and sends it to M_2. In step i, player M_i raises the quantity received in the previous step to S_i, and sends it to M_{i+1}.

Stage 2. M_{n-1} broadcast π to everyone.

Stage 3. Each M_i (i = 1,2, ..., n-1) removes its contribution from π but inserts a random blinding factor. The resulting quantity is encrypted using P. Each M_i (i = 1,2, ..., n-1) sends in parallel the encryption to M_n.

Stage 4. M_n decrypts the received messages and extracts c_i. It then raises each c_i to S_n and returns the result (in parallel) to each M_i. M_n can now compute the session key as $K = p^{S_n}$. Each M_i (i = 1,2, ..., n-1) computes the session key as $K = (c_i^{S_n})^{S_i / \hat{S}_i} = p^{S_n}$.

Stage 5. At least two different players broadcast a key confirmation message that allows each player to verify that at least one other player has decided on the same key K.

In [7], a fault tolerant group password based Diffie-Hellman key exchange on so-called d-cubes is also given. It does only work as long as attackers cannot remove or modify messages sent by honest players.

The main problem with password-based ad hoc key agreement is the distribution of the password. Passwords can only be distributed securely if humans control the units and that might be cumbersome for the users to handle.

4.1.5 Future trends

The research activities in the ad hoc networking security area are not large. The papers summarised here treat ad hoc networking from slightly different angles. Security problems closely related to ad hoc routing are somewhat different from the general ad hoc security problem treated in [1]. Trust models and tools to handle trust in ad hoc networks are areas that certainly need more work and we will probably see lot more contributions in the future.

4.1.6 References

[1] F. Stajano and R. Anderson, "The Resurrecting Duckling: Security for Ad-hoc Wireless Networks", in B. Christianson, B. Crispo, and M. Roe (Eds.), *Security Protocols, 7th International Workshop Proceedings*, LNCS, vol. 1796, Springer-Verlag 1999.

[2] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks": *IEEE Network Magazine*, vol. 13, no. 6 November/December 1999.

[3] E. Ayanoglu, C.-L. I, D. Gitlin and J. E. Mazo. "Diversity coding for transparent self-healing and fault-tolerant communication networks", *IEEE Transaction on Communications*, vol. 41, no. 11, November 1993.

[4] F. Stajano. "The resurrecting duckling -- what next?". B. Christianson *et al*. (Eds.), *Security Protocol, 8th International Workshop Proceedings*, LNCS ??, Springer-Verlag , April 2000.

[5] Y. Desmedt and Y. Frankel. "Threshold cryptosystems", *Advances in Cryptology - Crypto '89, LNCS 435*, Springer-Verlag, August 1990.

[6] P. Eronen, C. Gehrmann and P. Nikander, "Securing ad hoc Jini Services", *Proceedings of NORDSEC 2000*, Reykavijk, October 2000.

[7] N. Asokan and P. Grinzboorg, "Key Agreement in Ad-Hoc Networks", To appear *in Computer Communication Review*, 2000, http://www.semper.org/sirene/people/asokan/research/index.html

[8] S. Bellovin and M. Merrit, "Encrypted key exchange, Password based protocols secure against dictionary attacks", *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 1992.

4.2 Bluetooth

4.2.1 Description of the system

Bluetooth™ products and systems provide short range, wireless connectivity between a variety of common devices. Different applications can be built based on these spontaneous, ad-hoc networks.

Bluetooth can be used to connect almost any device to another Bluetooth enabled device. A typical example is to link a Personal Digital Assistant (PDA) or a laptop to a mobile phone. In this way one can easily set up connections with PDA or laptop without messing around with the cables. Bluetooth can also been used to form ad hoc networks, piconets, of several (up to eight) devices. This can be useful for example in a meeting, where participants with Bluetooth enabled laptops want to share their files with each other. PAN networking (see Sec. 2.2) offers natural applications for Bluetooth technology.

The security requirements for Bluetooth applications will vary, based on the sensitivity of the information involved, the market, and the needs of the user. There are some applications which do not require any security and others which require extremely high levels of security. Risk analysis and trade studies need to be conducted prior to implementing new applications using Bluetooth technology.

Within the Bluetooth SIG, applications are implemented as profiles. Working groups within the Bluetooth SIG define these profiles. The Security Expert Group (BSEG) provides the Bluetooth SIG and associated working groups with expertise regarding all aspects of Bluetooth security.

1.1.2 Technological environment

Bluetooth devices are categorized into three different classes by their transmission power. A class 3 device uses 1mW power and has a transmission range of 0.1 to 10 meters. A class 2 device uses power of 1 to 2.5 mW and a 10 meter range. A class 1 device uses power up to 100 mW and a range up to 100 meters.

The architecture of Bluetooth is composed of a radio, the base frequency (Baseband) and the Link Manager. In Europe, Bluetooth operates on radio frequency 2.45 GHz. The theoretical maximum bandwith is 1 Mb/s, which is somewhat reduced by Forward Error Correction. Bluetooth designates the frequency hopping to be implemented with Gaussian Frequency Shift Keying (GFSK).

The base frequency part of the Bluetooth architecture uses a combination of circuit and packet switching technologies. Bluetooth can support either one asynchronous data channel and up to three simultaneous synchronous speech channels, or one channel that transfers asynchronous data and synchronous speech simultaneously.

The Link Manager is an essential part of the Bluetooth architecture. It uses Link Manager Protocol (LMP) to configure, authenticate and handle the connections between Bluetooth devices. LMP also operates the power management scheme, which is divided into three modes: sniff, hold and park.

The Bluetooth Generic Access Profile (GAP) is the first of the Bluetooth profiles defined in the specification [3]. The GAP describes how devices are to behave in standby and connecting states in order to guarantee that links and channels always can be established between Bluetooth devices, and that multi-profile operation is possible. Special focus is put on discovery, link establishment and security procedures.

The modes of operation defined by GAP are not service or profile specific, but are generic and can be used by any other Bluetooth profile and also devices implementing multiple profiles. The following four types of modes of operation are defined by the GAP:

1. Discoverability modes
2. Connectability modes
3. Pairing modes
4. Security modes

Service Discovery Protocol in Bluetooth is used to inquire about services and service attributes. The procedures for discovery of supported services and capabilities using the Service Discovery Propocol are described in the Service Discovery Application Profile [3].

In piconets, one of the Bluetooth devices acts as a master and the others as slaves. The master sets the frequency-hopping behaviour of the piconet. It is also possible to connect up to 10 piconets to each other to form a scatternet.

4.2.2 Security services

The current Bluetooth specification [1] define security at the link layer. The security services are authentication and confidentiality. Also certain data integrity methods are specified but they are not sufficiently strong to offer protection against active disruption measures.

Application level security is not specified, allowing application developers the flexibility to select the most appropriate security mechanisms for their particular application. Hence Bluetooth does not define message authentication, end-to-end confidentiality, nor identification of the originating application.

A Bluetooth SIG white paper *Security Architecture* [2] defines the protocols and functionality required to implement the security services within a specific application category. The rules that determine the access rights to different resources on the devices is called the *access policy*. The access policy together with the description of the usage of basic security mechanisms, authentication and encryption, make up the *security policy*. The security policy is part of the security architecture for a Bluetooth application profile.

Entity authentication provides authenticated connections between Bluetooth devices. It is possible to make use of a hardware method for initial identification of Bluetooth devices during which a PIN value is inserted to the devices.

The link confidentiality is provided by an encryption method, which is a proprietary stream cipher algorithm.

Bluetooth link layer security measures shall be appropriate for a peer environment. This means that in each Bluetooth unit, the authentication and encryption routines are implemented in the same way.

4.2.3 Technical security features

4.2.3.1 Security data items

Four different data items are used for maintaining security at the link layer: a public address which is unique for each user, two secret keys, and a random number which is different for each new transaction. The four entities and their sizes as used in Bluetooth are summarized in the following table.

Data item	Size
BD_ADDR	48 bits
Private user key, authentication	128 bits
Private user key, encryption (byte-wise configurable length)	8-128 bits
RAND	128 bits

Table 1: Data items used in authentication and encryption processes

The Bluetooth device address (BD_ADDR) is the 48-bit IEEE address which is unique for each Bluetooth unit. The Bluetooth addresses are publicly known, and can be obtained via MMI interactions, or automatically, via an inquiry routine by a Bluetooth unit. It should be noted that the BD_ADDR is not a secured identity.

The secret keys are derived during initialization and are further never disclosed. Normally, the encryption key is derived from the authentication key during the authentication process. For the authentication algorithm, the size of the key used is always 128 bits.

For the encryption algorithm, the key size may vary between 1 and 16 octets. A method for truncating the encryption key has been specified for reducing the entropy of the encryption key. It is possible that such methods have been implemented also in hardware to permanently reduce the encryption key length in some Bluetooth modules.

Prior to encryption, the devices negotiate the encryption key length, and it must be agreed upon by both devices.

The authentication key is more static in its nature than the encryption key – once established, the particular application running on the Bluetooth device decides when, or if, to change it. To underline the fundamental importance of the authentication key to a specific Bluetooth link, it will often be referred to as the link key.

A link key can be one of the following three types:

- Master key
- Unit key
- Combination key

The RAND is a random number which is created using a random or pseudo-random process in a Bluetooth unit. This is not a static parameter, it will change frequently, and used for several different security-related purposes.

4.2.3.2 Security algorithms

Bluetooth link level security measures make use of the following security-related algorithms:

- Random or pseudo-random number generator
- Block cipher algorithm A_r which is identical to SAFER+. From this a modification A'_r is derived, which makes the algorithm non-invertible.
- E an expansion of an L octet word to a 128-bit word
- A hash function *Hash* consists of A_r, A'_r, E, offset, xor, and addition mod 256

  inputs: two 128-bit strings, $8\times L$ -bit string, integer L = 6 or 12,

  output 128 bits
- Authentication algorithm E_1 , which is equal to *Hash* with L = 6

  inputs: RAND 128 bits; K_{link} 128-bits; BT_ADDRESS 48-bit

  outputs: SRES 32 bits; ACO 96 bits
- Key derivation function E_2 uses A'_r and has two modes E_{21} and E_{22}
  - E_{21} is used for generation of combination keys and unit keys

    inputs: RAND 128 bits, BD_ADDR 48 bits

    output: KEY 128 bit
  - E_{22} is used to derive initialization keys and master keys

    inputs: PIN' ($8L'$ bits) which is the PIN *augmented* with BD_ADDR, L' (= 6,7,…,16)
    length of PIN' : $L' = \min\{16, L+6\}$, where L is the length of the PIN; and RAND 128 bits

    output: KEY 128 bit
- E_3 is the key generation function for encryption. It is equal to *Hash*, with L = 12.

  inputs: K_{link} 128-bit; EN_RAND 128-bit; COF = BD_ADDR||BD_ADDR, if the link key is master key (of a master with address BD_ADDR), else COF = ACO

  outputs: encryption key K_C 128 bits
- E_0 is stream cipher algorithm, which a generalization of the Massey-Rueppel combination generator. It is re-synchronized for every payload.

4.2.4 Bootstrapping security

The security context creation between two BT devices is established by creation of a link key. The link keys are generated and distributed among the Bluetooth units during the initialization process which consists of the following five steps:

- Generation of the initialization key
- Generation of the link key
- Link key exchange
- Authentication
- Generating an encryption key.

The security context is based on the current link key between two devices. The **link key** can be established using one of the following ways

- It is given in both devices by an application
- It is established as a unit key supplied by one device
- It is established as a combination key by both devices
- It is established as a master key supplied by the master of the piconet

The link keys obtained using methods 1, 2 and 3 are *semi-permanent*, the master key is typically *temporary*, and used for broadcasting messages from the master to the slaves.

To establish a link key based on a unit key or a combination key, a shared **initialization key** is created in both devices first. For this purpose, algorithm E_{22} is used. The initiator device generates a random number RAND and supplies a PIN of certain length. The third input to E_{22} algorithm is the responder's address.

Unit key is generated by a device when it is in the operation for the first time using algorithm E_{21}. The inputs are a RAND generated by the device and its address. Unit key is stored in a non-volatile memory and (almost) never changed. The unit key is transported to the other device masked by the initialization key.

Both devices contribute in an identical way to the generation and exchange of a **combination key**. A device generates a random number and computes its key part as the output of algorithm E_{21} using its own address. Then it sends the random number to the other device masked by the initialization key. The other device retrieves the random number and computes the sender's key part. The combination key is obtained as the xor sum of the two key parts.

A new combination key can be generated and exchanged also if a previous link key exists between two devices. Then the current link key is used in the place of the initialization key.

A **master key** is derived by the master from two 128-bit random values using algorithm E_{22}. It is transported to a slave masked by an overlay computed from a random number and the current link key using algorithm E_{22}. The random number used in this computation is also sent to the slave, who then can compute the overlay, and subsequently, recover the master key.

4.2.5 Trust model

The purpose of Bluetooth Baseband level security is to create and maintain a trusted connection between two Bluetooth devices using a wireless radio link. The trust model is the one derived from the wireline connection using a cable. The following trust-related procedures are needed:

- Maintenance of a list (access list) of devices with which pairing is allowed (authorization)
- Initial identification of the other device (initial authentication)
- Verification of the identity of the device at the other end of the link to see if it is the same as initially (authentication)
- A method to prevent eavesdropping of the link (encryption).

A solution for the first procedure for the Bluetooth PAN Profile is being developed (not yet available). One possibility is to use a specific entity for managing access rights and the device data base as described in the white paper on Bluetooth Security Architecture [2].

For initial identification different methods are used in different scenarios:

- Identification based on a priori keys using higher (application) layer authentication mechanisms.
- Identification of the device using physical means by a user (or users) , and/or identification of the user of the device (meeting scenario)

In Bluetooth, if the initial identification is successful, it is acknowledged by inserting a PIN, or by inserting a link key to the device.

After the initialization, the connection between the two devices can be protected using link level security mechanisms or higher level security mechanisms.

4.2.6 Strengths and weaknesses

No significant weaknesses have been reported in the cryptographic algorithms and protocols provided by the Bluetooth baseband security specifications. If properly initialized, Bluetooth baseband security can be considered sufficiently strong for transferring (session) key material or other secret information for higher layer security applications.

If a PIN is used it must be sufficiently long and secret. Two types of attacks can be distinguished

- Passive eavesdropping
- Active man-in-the middle

Both attacks use the possibility to perform exhaustive search of the PIN. In the passive attack, after recording the traffic, the eavesdropper has unlimited time to perform exhaustive search. In the active attack the man-in-the-middle initiates the Bluetooth authentication protocol, and can test the used PIN value against the received response. This exhaustive search must be successful before the other party expects the response from the man-in-the-middle.

In the light of the identified attacks it is essential to develop practical methods exchanging PINs that are sufficiently, preferably full 128 bits, long.

Another, not cryptography related problem, is concerned with anonymity and privacy protection of the Bluetooth device users. When in discoverable mode, a Bluetooth device is broadcasting its unique device address, thus making it possible to locate the device, and further using this facility, to track the whereabouts of the user of the Bluetooth device. This problem was discussed in [4]. The user is responsible for turning the discoverable mode off, and must be instructed not to keep the device in discoverable mode unless necessary.

4.2.7 Future trends

No essential changes are expected in Bluetooth Core specification. Further development is active in the Bluetooth working groups where the Bluetooth profile specification work is done. There the question, shall the profile rely on Bluetooth baseband security, or not, must be decided. The benefits offered by the baseband authentication are not always fully understood, which may lead to solutions, where security is based on higher (application) layer security mechanisms.

There is also some activities going on for developing solutions to support anonymity, by migrating temporary pseudonyms to be used instead of fixed public Bluetooth addresses.

4.2.8 References

[1] Bluetooth Special Interest Group, Baseband Specification, Specification of the Bluetooth System, Core, Version 1.1, December 1, 2000, http://www.bluetooth.com/

[2] Bluetooth Special Interest Group, *Bluetooth Security Architecture*, Version 1.0, 15 July 1999.

[3] Bluetooth Special Interest Group, Specification of the Bluetooth System, Volume 2, Profiles v 1.1 (2001)

[4] M. Jakobsson and S. Wetzel, Security Weaknesses in Bluetooth, RSA Conference 2001, April 2001

4.3 MExE

4.3.1 Description of the system

The intention of the MExE environment (Mobile Execution Environment) is to provide an environment for securely downloading and executing software, from different parties and with different security constraints, on mobile devices. The MExE specification was first created by ETSI and is now maintained and developed by the 3GPP. More information of MExE can also be found at http://www.MExEforum.org .

Considering the wide and diverse range of current and future technology and devices that (will) use wireless communication and provide services based thereon, a one-size-fits-all approach is unrealistic. Instead the mobile devices are categorised in three classmarks, depending on their capabilities.

In order to manage the MExE and prevent attack from unfriendly sources or transferred applications unintentionally damaging the MExE device, a security framework is defined and consists of three different security domains, i.e. the Operator Domain, the Manufacturer Domain and the Third Party Domain.

4.3.2 Technological environment

The following architectural model shows an example of how standardised transport mechanisms (WAP, HTTP, etc.) are used to transfer MExE services between the Mobile Terminal and the MExE service environment, or to support interaction between two Mobile Terminals executing a MExE service.

The MExE service environment can consist of several service nodes, each providing MExE services that may be transferred to the Mobile Terminals using mechanisms such as fixed/mobile/cordless network protocols, Bluetooth, infrared, serial links, wireless optimised protocols (e.g. WAP), standard internet protocols. These service nodes may exist in the circuit switched domain, packet switched domain, multimedia subsystem or in the Internet space (e.g. SMS servers, multimedia messaging servers, internet servers etc.). The MExE service environment may also include a proxy server (e.g. WAP gateway) to translate content defined in standard Internet protocols into their wireless optimised derivatives.

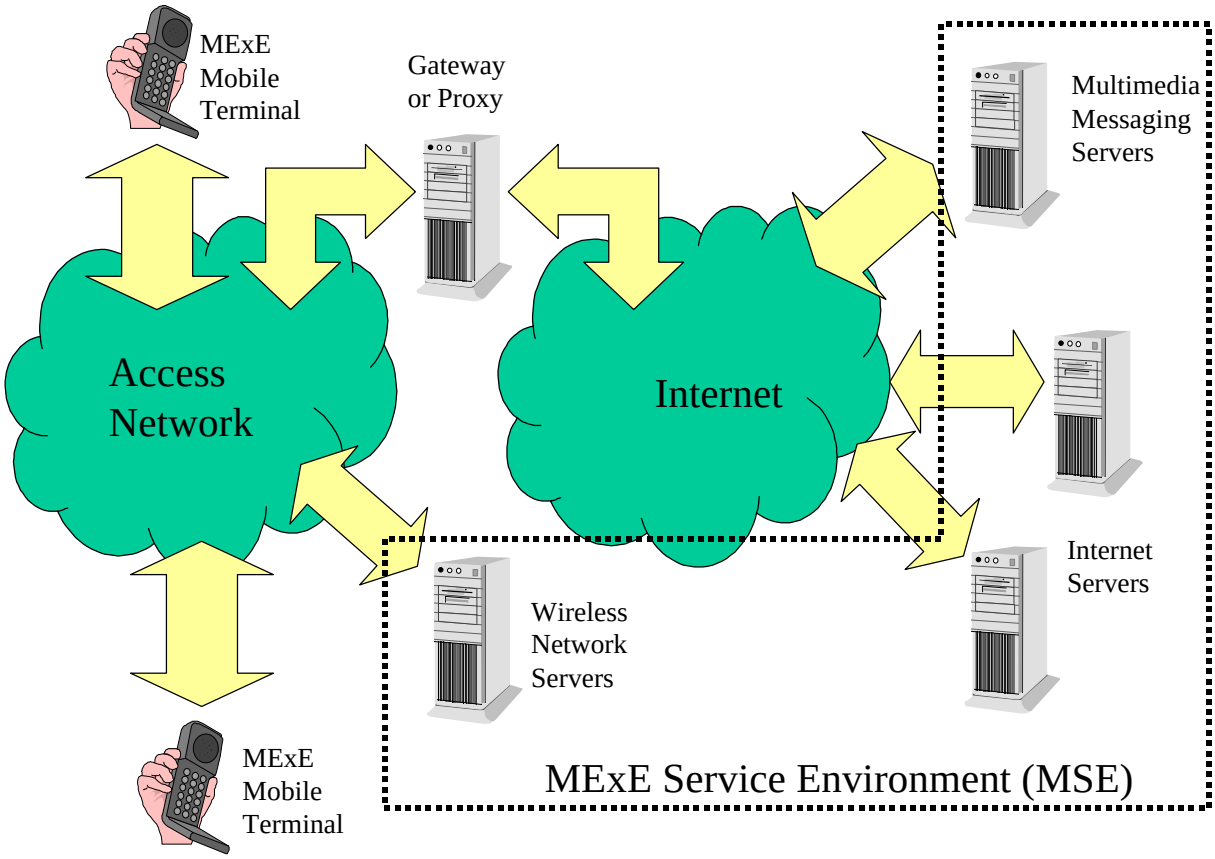


Figure 10: The MExE Service Environment

Mobile devices will be categorised in three classmarks, depending on the capabilities of the device. In order to be MExE compliant, support of at least one MExE classmark in mandatory. A MExE UE may also include optional support for applications form any other MExE classmark. The following MExE classmarks are defined:

- MExE classmark 1: Based on WAP (Wireless Application Protocol) – requires limited input and output facilities on the client side, and is designed to provide quick and cheap information access even over slow data connections.
- MExE classmark 2: Based on PersonalJava with the addition of the JavaPhone API – provides and utilises a run-time system requiring more processing, storage, display and network resources, but supports more powerful applications and more flexible MMIs (Man-Machine Interface). The PersonalJava application environment is a Java platform for building network-connectable applications for consumer devices for home, office and mobile use. It is comprised of the Java Virtual Machine (VM) and an optimised version of the Java class library. In addition, the PersonalJava API includes specific features required by consumer applications in resource-limited environments. JavaPhone is vertical extension to the PersonalJava platform that defines APIs for telephony control, messaging, address book and calendar information, etc.

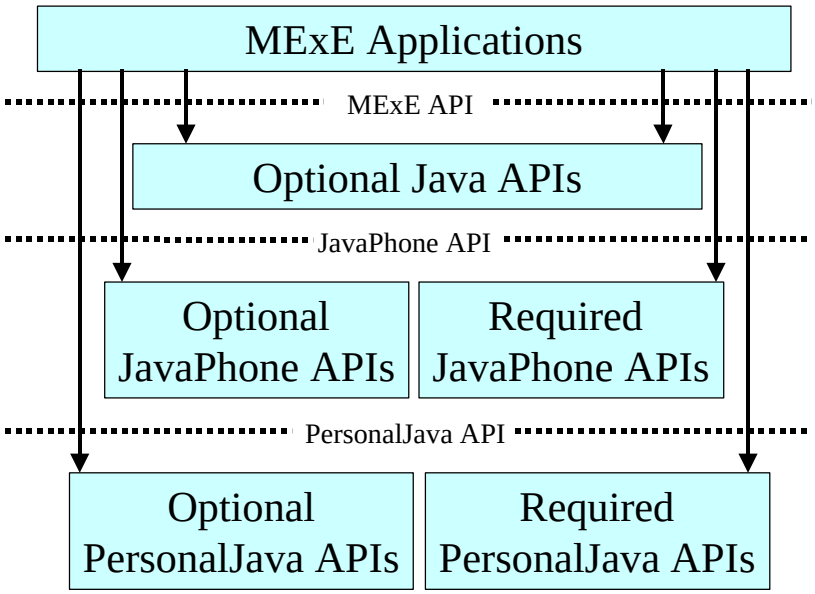


Figure 11: MExE Classmark 2 high level architecture

- MExE classmark 3: Based on Java 2ME (Java 2 Micro Edition) CLDC (Connected Limited Device Configuration) and MIDP (Mobile Information Device Profile) environment – supports Java applications running on resource constrained devices. J2ME specifically addresses the vast consumer space, which covers the range of extremely tiny commodities such as smart cards and pagers all the way up to the set-top box, an appliance almost as powerful as a computer. In order to fit on the various types of devices and also to support extensibility, J2ME introduces the concept of *Configuration and Profiles*. A *Configuration* defines a minimum platform with a virtual machine and libraries that are available on all devices of a similar class. In a *Profile* J2ME addresses the specific demand of a certain category of devices, by including additional APIs. A *Profile* is implemented on top of a *Configuration*.

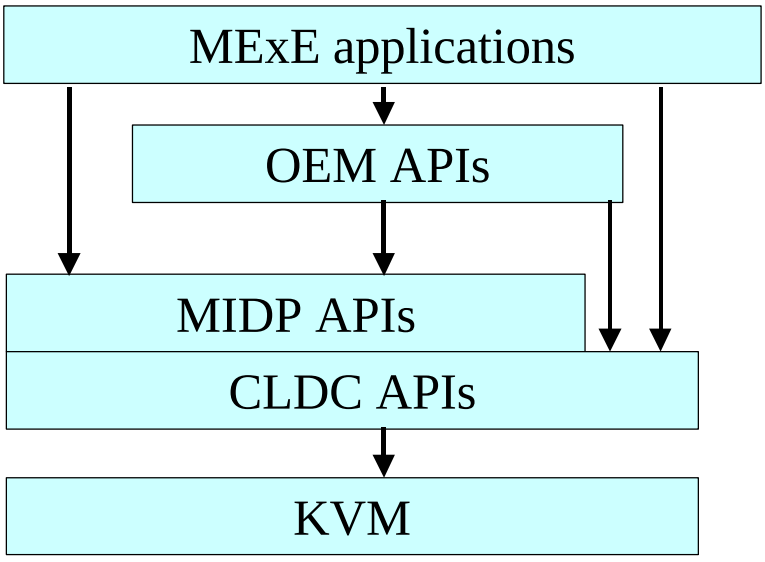


Figure 12: MExE Classmark 3 high level architecture

4.3.3 Security services

In order to manage the MExE and prevent attack from unfriendly sources or transferred applications unintentionally damaging the MExE device, a security system is required.

The basis MExE security is:

- A framework, which defines the permissions that, transferred MExE executables have within the MExE MT.
- The secure storage of these permissions.
- Conditions within the execution environment that ensure that MExE executables can only perform actions for which they have permission.

In other words, authentication (i.e. digital signatures) is required by applications to access certain functionality. Once they have been authenticated, the user gives the applications explicit permission for that functionality to be accessed.

To this end several security services are defined within the MExE environment:

- ***MExE executable authentication***: The MExE permission framework consists of 3 security domains, i.e. the Operator Domain (MExE executables authorised by the network operator), the Manufacturer Domain (MExE executables authorised by the terminal manufacturer), the Third Party Domain (trusted MExE executables authorised by trusted third parties). Untrusted MExE executables require no authentication and are not permitted to execute in a security domain. Instead, they execute in the Untrused Area. They have very reduced privileges and are not permitted to access any actions that access the phone functionality. They are capable of making phone calls with user permission.
- ***User permissions***: The user will be prompted for user permissions whenever an application wishes to perform an action that may be seen as a security risk from the user's standpoint. The following types of permissions are available in the MExE environment: Blanket Permission, Session Permission, and Single-Action Permission
- ***MExE Certification and Authorisation***: In order to enforce the MExE security framework, the MExE capable Mobile Terminal is required to operate an authentication mechanism for verifying downloaded MExE executables. A successful authentication will result in the MExE executable being trusted; and able to be executed in a security domain – Manufacturer, Operator or Third Party. As the MExE Mobile Terminal may want to authenticate content from many sources, a public key based solution is implemented.

4.3.4 Technical security features

4.3.4.1 MExE executable permissions

4.3.4.1.1 MExE executable permissions for operator, manufacturer and third party security domains

The following table specifies the permission of operator, manufacturer and third party security domains in the order of restriction. If an action can be categorised into a more restrictive group near the top of the table, then it shall not be again categorised into another, less restrictive, group further down in the table.

	MExE Security Domains		
Actions	**Operator**	**Manufacturer**	**Third Party**
Device core function access: includes functions which are an essential part of the phone functionality	No		

Support of Core Software Download: e.g. update UE software	No	Yes	No
SIM smart card low level access: includes functions which allow communications at the transport service access point (send and receive application protocol data unit)	No		
Network Security access: includes all functionalities which relate to CHV, CHV2, UNBLOCK CHV and UNBLOCK CHV2 (verification, management, reading or modifying), GSM authentication, GSM ciphering.	No		
Network property access: includes functions which enable the management of operator-related data parameters and network settings.	Yes	No	
Network services access: includes all functionalities which result in or need interaction via the operator's network	Yes		Yes
User private data access: includes all functionalities which relate to management, reading or modifying of data that the user has stored in the MS including user preferences.	Yes		
MExE security function access: includes all functionalities which relate to certificate handling in the MS; end to end encryption, signed content, hashing, access to public, private, secret keys stored in the MS or in a smart card.	Yes		
Application access: includes the functionalities which relate to launch provisioned functionality, MExE executables, external executables (SIM toolkit application, …) usage.	Yes		
Lifecycle management: includes the functionalities which are needed for installing or removing MExE executables in the MS.	Yes		
Terminal data access: includes the functions which relate to accessing terminal data, i.e. not user data.	Yes		
Peripheral access: includes the functionalities related to peripherals other than user interface peripheral usage through a high level software application interface.	Yes		
Input/output User interface access: includes the functionalities related to the user interface and user notification means usage.	Yes		

Table 2: Security domains and actions

4.3.4.1.2 MExE executable permissions for untrusted MExE executables

When the security domains are not supported, then all executables are untrusted and they will execute in the untrusted area for which very restrictive permissions are defined with respect to:

- access the user interface

- file persistent data and file access
- initiating a voice/data connection
- generating a DTMF
- adding a phonebook entry
- executable interaction

The untrusted MExE executables permitted to use these facilities shall be MExE executables the user has downloaded himself and not MExE executables that have been pushed to the user. Untrusted MExE executables shall not be permitted access to any other functions.

4.3.4.2 User permission types

The term “user permission” is defined to mean that the user can give permission for a specific action in one of the ways as described below:

- Blanket Permission – The user gives blanket permission to the MExE executable for the specified action, and the MExE executable subsequently uses the user’s original permission for subsequent actions whenever the MExE executable is running. Typically such permission would be given at MExE executable configuration or run time. The user may revoke the blanket permission at any time. The user permission no longer applies once the MExE executable has been removed.
- Session Permission – The user gives permission to the MExE executable for the specified action during a specific run time session of a MExE executable. The MExE executable subsequently uses the user’s permission for subsequent actions whilst the MExE executable session is still running. Typically such permission would be given at MExE executable run time. The user may revoke the session permission at any time. The user permission no longer applies once the MExE executable run time session has terminated.
- Single-Action Permission – The user gives a single permission to the MExE executable for the specified action. If the MExE executable subsequently wishes to repeat the action, it must again request the user’s permission for the identified subsequent action. Typically such permission would be given at MExE executable run time. The user permission no longer applies once the action has terminated.

4.3.4.3 Certification and authorisation architecture

In order to enforce the MExE security framework a MExE capable MS is required to operate an authentication mechanism for verifying downloaded MExE executables. A successful authentication will result in the MExE executable being trusted, and able to be executed in a security domain (as determined by the root public key of its certification tree).

As the MExE MS may want to authenticate content from many sources, a public key based solution is mandatory. Before trusting MExE executables, the MExE MS will check that the MExE executable was signed with a private key, for which the MExE MS has the corresponding public key. The corresponding public key held in the MS must either be a root public key or a signed public key provided in a certificate. The MExE MS must be able to verify certificates. Support for certificate chains is therefore mandatory. However, it is not necessary for the intermediate public keys to reside on the terminal, they can be downloaded onto the terminal such that a valid certification path is established to one of the root public keys on the terminal.

4.3.4.3.1 Certificate management

The manufacturer may load initial third party certificates on the device. Downloaded certificates shall be verified by an existing trusted certificate and placed in the domain defined by the root public key at the top of the verification chain for the downloaded certificate.

The administrator root certificate shall be provided on the SIM if support for certificate storage on the SIM exists. For SIMs not having certificate storage, the administrator root may be downloaded.

The actions that may be performed for a given certificate are:

- addition,
- deletion,
- mark un-trusted (un-trusted certificates cannot be used to verify applications or other certificates. This process may be preferred to certificate deletion as there is a change that the certificate may become trusted again in the near future),
- mark trusted (marking as trusted is the process of allowing an untrusted certificate to come into use again),
- modify fine grain access permissions (proposed as a future enhancement).

The ability to perform these actions depends on the certificate type being modified as well as the access level of the entity performing the operation. Users may add a third party certificate as long as it is certified by an existing trusted certificate.

Using a provisioned functionality, users may delete Third Party certificates.

4.3.4.3.2 Certificate extension for removal of network access

MExE defines the certificate extension (attribute) “access-Restriction”. If the access-Restriction extension is present in a certificate used to verify the signature on a trusted application or in any certificate in the certificate chain used to verify that signature, then the application shall not be permitted network service access capabilities.

The extension prevents the trusted applications of developers who do not need network service access from writing applications that can perform network service access.

The support of this extension in the operator domain is mandatory. The support of this extension in the manufacturer and third party domain is optional.

The extension is defined for X.509v3 only. Support for WTLS, X9.68 certificate formats is for further study.

4.3.5 Bootstrapping security

The execution of MExE applications is based on the ability to validate the certificate, belonging the particular application. Therefore, a MExE MS cannot verify certified MExE executables, and run the executables, of a particular domain unless it has a root public key for that particular domain.

Root public keys shall be securely installed in the MExE MS, say at manufacture time.

It is also recommended that a “disaster recovery” root public key be securely installed on the terminal, to be used to install root public keys when all other root public keys on the terminal are invalid.

Third Party Domain root public keys will typically be installed along with and integrated into the MExE ME browser, as is done for PC-based browsers.

A MExE mobile station shall support at least one level of certificate under operator, manufacturer or Third Party root public keys.

4.3.6 Trust model

The MExE trust model is based on the two types of Java security i.e. sandbox and fine grain.

The sandbox model has just one domain; there is no concept of a partly trusted domain. The essence of the sandbox model is that local code is trusted to have full access to vital system resources while downloaded remote code is not trusted and can access only the limited resources provided inside the sandbox.

Using the sandbox system, each MExE security domain shall be implemented as running in a sandbox, configured with different privileges corresponding to those of the domain. If the security domains are

not supported then the Java sandbox security model shall be supported and it shall be configured for untrusted MExE executable support only. Using the fine grain Java security system, each security domain will be a set of constraints within which a Java fine grain security domain can be configured. However, the current MExE specification does not support fine grain security in Classmarks 1 and 3.

4.3.7 Strengths and weaknesses

Although not a weakness in itself, there is a limitation in the MExE security model that should be considered. Namely, the current MExE specification relies on every application to chain up only to the root public key that controls the domain which it was targeted. However, there are instances when there are more than one certification path that could terminate in two different root public keys. If such a scenario occurs, there will be some ambiguity on the targeted execution domain.

4.3.8 Future trends

There are many discussions within the 3GPP MExE Group as well as many proposals to the group, concerning the future development of MExE. MExE is an evolving standard that incorporates many different technologies. As these technologies evolve, so will MExE. Similarly, as new technologies emerge, they may be incorporated into the MExE specification.

Already there are proposals from companies like Microsoft to include Common Language Infrastructure (CLI) and C# into the MExE specification.

MExE has already received much interest and will continue to gain further support as 3G becomes more a reality. MExE is set to become the standard of the future.

4.3.9 References

[1] The official Web-site can be found at http://www.MExEforum.org .

[2] MExE White Paper to be found at http://www.MExEforum.org .

[3] 3GPP TS23.057, “Mobile Station Application Execution Environment (MExE)”.

[4] J2SE security http://www.javasoft.com/j2se/1.3/docs/guide/security/index.html .

[5] Java 2 security http://www.javasoft.com/products/jdk/1.2/docs/guide/security/index.html .

[6] Java security tutorial http://java.sun.com/docs/books/tutorial/security1.2/overview/index.html .

4.4 IETF Technologies (MANET, SLP, ZEROCONF)

The Internet Engineering Task Force IETF is an open international community which is concerned with the development of internet standards. Internet standards are published as Request for Comments (RFC) which are available at http://www.ietf.org/rfc and various mirror sites. Working documents called Internet drafts are available at http://www.ietf.org/1id-abstracts.html. Unrevised Internet drafts have a maximum life time of six months. After that time, they must be updated, or they will be deleted. After a document becomes an RFC, it will be replaced in the Internet-Drafts Directories with an announcement to that effect. The standardization process of the IETF is explained in section 6 of RFC2026 [1].

The actual technical work of the IETF is done in its working groups, which are organized by topic into several areas. In the following sections, the working groups concerned with topics which are considered to be relevant for SHAMAN are described.

4.4.1 Mobile Ad-hoc Networks (MANET)

4.4.1.1 Description of the system

A Mobile Ad hoc NETwork (MANET) consists of mobile platforms, e.g. a router with multiple hosts and wireless communication devices, which are designated as nodes. These nodes are free to move about arbitrarily. The nodes may be located in or on satellites, airplanes, ships, trucks, cars, perhaps

even on people or very small devices, and there may be multiple hosts per router. A MANET is an autonomous system of mobile nodes. The system may operate in isolation, or may have gateways to a fixed network.

A MANET is an autonomous system of mobile routers (and associated hosts) connected by wireless links. The routers are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Such a network can operate in isolation or be connected to the larger Internet via exterior routing functionality. In this case, it is envisioned that MANETs will operate as *stub* networks, i.e. that traffic carried by MANET nodes either originates from a node or is destined for a node that is part of the MANET.

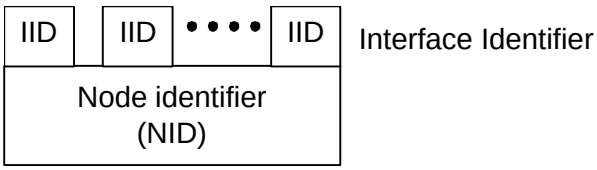


Figure 13: Generic MANET Router Structure

Each node in a mobile ad-hoc network logically consists of a router and possibly multiple hosts and wireless communication devices which are connected through interfaces [Figure 13]. The MANET node is identified by its node identifier (NID). A MANET node can own several interfaces with associated interface identifiers (IID) by which attached hosts and other MANET nodes are connected. A node may be a single physical device, or it may consist of several physically separate devices. The dominant property of a MANET network is its dynamic routing topology which is caused by the fact that nodes can move arbitrarily. The size of a MANET network can range from small-scale as sensor networks over multi-hop WLAN extensions to networks consisting of several hundreds of nodes.

The primary focus of the MANET working group is to develop and evolve routing and interface definition standards that support self-organizing, mobile networking within the Internet protocol suite. This means that – contrary to MobileIP where individual hosts are moving – the nodes of the routing infrastructure itself are moving. The MANET working group develops routing protocols for such networks. The charter of the MANET working group and published RFCs and Internet drafts can be found at http://www.ietf.org/html.charters/manet-charter.html. At the time of this writing, the RFC2501 [2] “Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations” is available which describes the scope of MANET and criteria for the evaluation of MANET routing protocols. Furthermore, twelve Internet drafts describing different MANET routing protocols or variants and extensions thereof are available.

4.4.1.2 Technological environment

The MANETs are characterized by their dynamic topology, bandwidth-constrained, variable-capacity links due to the wireless links, and energy-constrained operation of battery powered nodes. The nodes are equipped with wireless transmitters and receivers and exchange routing information via wireless links which are – without some form of network-level or link-layer security – vulnerable to many forms of attacks as snooping network traffic, replay transmissions, manipulation of packet headers, and redirection of routing messages. At a given point in time, depending on the nodes’ positions and their receiver and transceiver coverage patterns, transmission power levels and co-channel interference levels, a wireless connectivity in the form of a random, multihop graph or “ad hoc” network exists between the nodes. This ad hoc topology may change with time as the nodes move or adjust their transmission and reception parameters. It is important to note that the routing in a MANET is intended to occur at the IP layer.

4.4.1.3 Security services

The MANET working group concentrates on the development of different routing protocols for ad-hoc networks. Currently there does not exist a common MANET technology, only a set of different proposed routing protocols (see Table 3 below on the Internet drafts available at the time of this

writing). Currently only one MANET RFC “Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations” exists which describes characteristics of MANETs and discusses the effects these have on the design and evaluation of network control protocols (routing protocols). While the emphasis is on routing performance evaluation considerations, RFC2501 shortly addresses security issues and only mentions that the decentralized nature of a manet network provides robustness against single points of failures. Without some form of network-level or link-layer security, a MANET routing protocol is vulnerable to many forms where the most pressing issue is one of inter-router authentication prior to the exchange of network control information. It may be relatively simple to snoop network traffic, replay transmissions, manipulate packet headers, and redirect routing messages. While these concerns exist within wired infrastructures and routing protocols as well, maintaining the “physical” security of the transmission media is harder in practice with MANETs. The application of IPsec is mentioned as one possibility to prohibit disruption or modification of protocol operation. It is expected that several levels of authentication will be explored by the group ranging from no security and simple shared-key approaches to full public-key infrastructure-based authentication mechanisms. However, RFC2501 [2] does not state more specific security requirements or possible solutions.

Some of the MANET Internet drafts proposing different routing protocols contain also some security information. Table 3 summarizes this information.

Table 3: Security Considerations of MANET Internet drafts

Internet Draft	**Security Considerations**
draft-ietf-manet-aodv-08.txt: Ad Hoc On-Demand Distance Vector (AODV) routing protocol	Currently, AODV does not specify any special security measures. Route protocols, however, are prime targets for impersonation attacks, and must be protected by use of authentication techniques involving generation of unforgeable and cryptographically strong message digests or digital signatures. It is expected that, in environments where security is an issue, that IPSec authentication headers will be deployed along with the necessary key management to distribute keys to the members of the ad hoc network using AODV.
draft-ietf-manet-dsr-05.txt: Dynamic Source Routing Protocol	Security concerns are not addressed specifically. It is assumed that all nodes participating in the DSR protocol do so in good faith and without malicious intent to corrupt the routing ability of the network. In mission-oriented environments where all the nodes participating in the DSR protocol share a common goal that motivates their participation in the protocol, the communications between the nodes can be encrypted at the physical channel or link layer to prevent attack by outsiders.
draft-ietf-manet-dsrflow-00.txt: Flow State in the Dynamic Source Routing Protocol	All security considerations outlined in the base DSR protocol apply as well (see above).
draft-ietf-manet-simple-mbcast-00.txt: Simple Protocol for Multicast and Broadcast	Security issues relevant to DSR apply as well (see above).
draft-ietf-manet-fsr-00.txt: Fisheye State Routing Protocol (FSR)	Security issues are not addressed.

draft-ietf-manet-lanmar-00.txt: Landmark Routing Protocol (LANMAR)	It is stated, that LANMAR provides some kind of security. When a node broadcasts routing update message, only entries of in-scope nodes and landmarks are included. This will prevent other remote nodes from being heard. Whilst the statement is true, it is unclear how it relates to security.
draft-ietf-manet-olsr-04.txt: Optimized Link State Routing Protocol	The OLSR routing protocol does not provide security itself. However, it is mentioned that upon receiving a HELLO message – which are broadcasted by each node to check links – , the node should update the neighbor information corresponding to the sender node address. A node may – e.g. for security reasons – wish to restrict updating the neighbor-table, i.e. ignoring HELLO messages from some nodes.
draft-ietf-manet-tbrpf-01.txt: Topology Broadcast Based on Reverse-Path Forwarding (TBRPF)	Authentication issues exist for allowing "foreign" nodes to join a MANET via TBRPF neighbor discovery. Additionally, privacy issues exist for any networking protocols run over unencrypted wireless data links such as IEEE 802.11. Finally, denial-of-service attacks are possible if rogue nodes join a TBRPF MANET and offer to provide a multi-hop relay service, but then fail to perform the service when it is required. It is believed that IPSec may be useful in addressing some or all of these issues. When UDP/IPv4 is used to transmit TBRPF packets, Internet security mechanisms with policy-based address filtering can be used.
draft-ietf-manet-tora-spec-03.txt: Temporally-Ordered Routing Algorithm (TORA)	TORA has been designed to work on top of lower layer mechanisms or protocols that provide – beside other services – authentication.
draft-ietf-manet-zone-brp-00.txt: The Bordercast Resolution Protocol (BRP)	Does not contain any information on security.
draft-ietf-manet-zone-iarp-00.txt: The Intrazone Routing Protocol (IARP)	It is assumed that security issues are adequately addressed by the underlying protocols, e.g. by IPsec.
draft-ietf-manet-zone-ierp-00.txt: The Interzone Routing Protocol (IERP)	It is assumed that security issues are adequately addressed by the underlying protocols, e.g. by IPsec.

To summarize, the different MANET Internet drafts deal with security in the following ways:

- Ignore security aspects completely.
- Refer to security-mechanisms at the link-layer or using IPsec [23] to authenticate network control messages.
- One Internet draft mentions policy aspects shortly that a node may decide whether to accept routing information dependent on the sending node.

4.4.1.4 Technical security features

The routing protocols proposed at the time of this writing do not introduce any security mechanisms. Instead they either do not address security or they simply refer to IPsec or link-layer security mechanisms. Therefore no technical security features specific to MANET are to be considered.

An outdated Internet draft from S. Jacobs and M. S. Corson (February 25, 1999) has been available that specified authentication parameter to be included in a IMEP protocol entity (Internet MANET Encapsulation Protocol, which seems to be outdated, too). This document mentions different authentication options (secret key keyed-MD5 with prefix-postfix-method [19]; although HMAC seems to be preferable, RSA signature on MD5 hash with key-length of 512, 768, 1024, 2048 bit; EC signature with 80, 120, 160 bit, DSA with 512 bit key length).

When keyed-MD5 is used, the MSA Manet Security Associations (cryptographic mechanism and key) of each MANET node that a node will interact with have to be configured. For PKI solution, self-signed certificates forming a PGP-style web-of-trust and certificates issued by CAs are considered. When both nodes share the same CA (presumably meaning that their end-entity certificate has been issued by the same CA without an intermediate CA) or when self-signed certificates are used, the receiver uses the CA's or senders public key to verify the signature. When the sender and the receiver do not share a common CA, then it is tried to build a trust-hierarchy path between the receiver's CA and the sender's CA using the certificates included in the message. A trust hierarchy path between the sender certificate, the CA that issued the sender certificate, and the CA of the validating system has to be constructed. This proposal is unclear and seems to be unsecure. (no trusted root CA is mentioned, the properties are not precisely defined that the chain has to satisfy – it is only required that the subject and issuer name the chain correctly). Furthermore, the certificates have to be embedded in each message and each message is signed (high computational requirements and communication overhead). CRLs (Certificate-Revocation List) are mentioned, but it is left open how a current CRL can be obtained within an ad-hoc network..

4.4.1.5 Bootstrapping security

As current MANET Internet drafts currently do not specify any security techniques, no specific bootstrapping aspects are dealt with, too. However, when authentication of MANET nodes is involved, cryptographic data has to be available to authenticate the MANET node sending routing information, and policies have to be administratively defined that specify how to deal with routing information depending on its sender. The MANET Internet drafts referring to IPsec or relying on the security mechanisms of the underlying link layer, do not deal with necessary prerequisites as key management or policy definition.

4.4.1.6 Trust model

The nodes participating in an ad-hoc network are assumed to be well-behaving. By authenticating the sender of a routing message or by using address filtering which is an unreliable method mentioned in draft-ietf-manet-tbrpf-01.txt, a node can restrict the nodes that can be part of an ad-hoc network. Information received from a trusted node can (and usually will) be based on information it has received from third nodes. So trust relationship is in general transitive, i.e. each node has to trust all nodes participating in the ad-hoc network.

4.4.1.7 Strengths and weaknesses

Besides the functional strenghts of MANETs like forming dynamic routing topologies which occurs at the IP layer there are weaknesses like:

- While the robustness concerning the failure of a node or link is an intrinsic property of MANETs, routing protocols will in general not be robust against individual nodes providing intentionally rogue information to launch denial-of-service attacks or to ensure that – possibly unprotected – network traffic is routed across nodes controlled by the attacker.
- Proposals for routing protocols do either not deal with security issues in any way or suggest using IPsec or to rely on link-level security.

However there are several weaknesses of the MANET which are due to the fact that the focus is primarily on performance of routing protocols and therefore specific issues of ad-hoc networks are not addressed:

- key establishment is not considered.
- Simple trust model where all nodes of an ad-hoc network are expected to be trustworthy (e.g. how to deal with routing dependent on origin or different trustworthy nodes like some trusted ad-hoc nodes connected with untrusted nodes),
- does not deal with usage of networking infrastructure like who may use ad-hoc network and in which way, who may join it or which way is the access to offered services or gateways to fixed internet. How can infrastructure functions as e.g. naming service be accessed securely?

Replay protection or sending captured messages to other nodes is not considered and this effect depends on specific properties of the considered routing protocols.

4.4.1.8 Future trends

The MANET working group appears to concentrate on routing protocols of ad-hoc networks which are described in the above mentioned twelve internet drafts. In the moment of writing it is not clear which of this internet draft will be revised or supported in the future. While the working group charter identifies security issues of ad-hoc networks as a topic, it does not seem to be a main focus of former or current activities.

4.4.2 Service Location Protocol (SVRLOC)

4.4.2.1 Description of the system

The Service Location Protocol (SLP) provides a flexible and scalable framework for providing hosts with access to information about the existence, location, and configuration of networked services. Traditionally, users have had to find services by knowing the name of a network host (a human readable text string) which is an alias for a network address. The SLP eliminates the need for a user to know the name of a network host supporting a service. Rather, the user supplies the desired type of service and a set of attributes which describe the service. Based on that description, the Service Location Protocol resolves the network address of the service for the user.

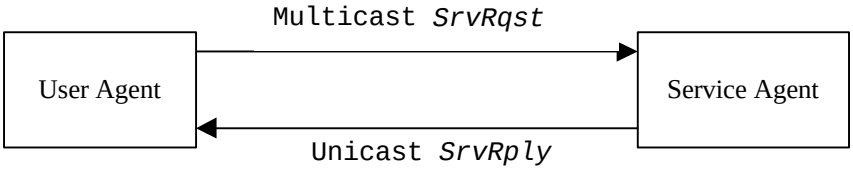
The SLP provides a dynamic configuration mechanism for applications in local area networks. The applications are modeled as clients that need to find servers attached to any of the available networks within an enterprise. For cases where there are many different clients and/or services available, the protocol is adapted to make use of nearby Directory Agents that offer a centralized repository for advertised services.

The Service Location Protocol is a decentralized, lightweight, scalable and extensible protocol for service discovery within a site. It allows but does not require centralized administration. Even when security, administrative policies or convenience require centralization (say in large enterprise deployments) the protocol requires very little administration. The protocol limits its use of multicast and broadcast as much as possible to conserve network bandwidth. Moreover, the protocol is extensible to arbitrary service advertisement and discovery and supports multiple languages and character set encodings.

The Service Location Protocol supports a framework by which client applications are modeled as 'User Agents' and services are advertised by 'Service Agents.' A third entity, called a 'Directory Agent' provides scalability to the protocol.

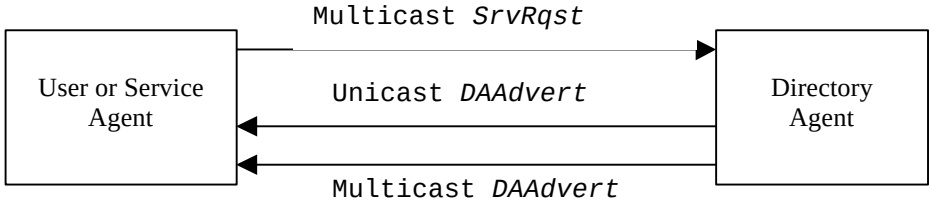
The User Agent issues a 'Service Request' (*SrvRqst*) on behalf of the client application, specifying the characteristics of the service which the client requires. The User Agent will receive a Service Reply (*SrvRply*) specifying the location of all services in the network which satisfy the request.

The Service Location Protocol framework allows the User Agent to directly issue requests to Service Agents. In this case the request is multicast. Service Agents receiving a request for a service which they advertise unicast a reply containing the service's location.



In larger networks, one or more Directory Agents are used. The Directory Agent functions as a cache. Service Agents send register messages (*SrvReg*) containing all the services they advertise to Directory Agents and receive acknowledgements in reply (*SrvAck*). These advertisements must be refreshed with the Directory Agent or they expire. User Agents unicast requests to Directory Agents instead of Service Agents if any Directory Agents are known.

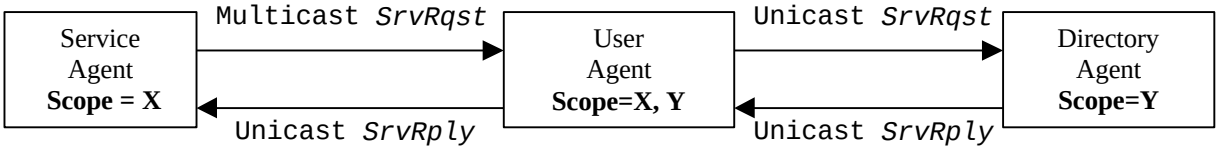
The User and Service Agents discover Directory Agents two ways. First, they issue a multicast Service Request for the 'Directory Agent' service when they start up. Second, the Directory Agent sends an unsolicited advertisement infrequently, which the User and Service Agents listen for. In either case the Agents receive a DA Advertisement (*DAAdvert*).



It is also possible for a SLP Agent to configure the Directory Agent to use statically or to use DHCP [7][17].

The Services are grouped together using 'scopes'. These are strings which identify services which are administratively identified. A scope could indicate a location, administrative grouping, proximity in a network topology or some other category. Service Agents and Directory Agents are always assigned a scope string which can be configured statically or using DHCP. When no scope string is configured, the default scope ‘DEFAULT’ is used.

A User Agent is normally assigned a scope string (in which case the User Agent will only be able to discover that particular grouping of services). This allows a network administrator to 'provision' services to users. Alternatively, the User Agent may be configured with no scope at all. In that case, it will discover all available scopes and allow the client application to issue requests for any service available on the network.



In the above illustration, the User Agent is configured with scopes X and Y. If a service is sought in scope X, the request is multicast. If it is sought in scope Y, the request is unicast to the DA. Finally, if the request is to be made in both scopes, the request must be both unicast and multicast.

A Service Advertisement contains a service URL which indicates the type of the advertised service and its location. The service URL syntax and its semantics are defined in [8]. A service URL is of the form “`service:<servicetype>://<address specification>`”. An example illustrates service URLs:

```
service:tftp://host37.company.de:5000
```

This service URL refers to a TFTP service (trivial file transfer protocol) running on the host with the name “host37.company.de” on port 5000.

The service type should by defined by a “Service Template” which provides expected attributes, values and protocol behavior. Furthermore there exist abstract service types which are of the form “`service:<abstract type>:<concrete type>`”, e.g.

“`service:printer:lpr://host37.company.de`”.

These refer to service types which can be used with a variety of different protocols.

When a service URL is registered, it is accompanied by a set of attributes which define the service. These attributes are used by User Agents in service requests to select appropriate services. The allowable attributes which may be used are typically specified by a Service Template for a particular Service Type that is readable by people and machines [8]. These attributes convey configuration information to client software, or service characteristics meaningful to end users.

The Service Template contains a human-readable description of the service, the syntax of the service type-specific URL part, and definitions for the attributes describing the service. At the Internet Assigned Numbers Authority IANA there exists a repository for service templates which can be accessed at the URL http://www.isi.edu/in-notes/iana/assignments/svrloc-templates/.

The SVRLOC working group has already defined RFCs. The second revision of the Service Location Protocol SLPv2 is specified in RFC2608 [7]. The RFC2609 [8] defines service templates which are a means to describe services and their describing attributes that can be announced and looked for using SLP. The charter of the SVRLOC working group, published RFCs and internet drafts can be found at http://www.ietf.org/html.charters/svrloc-charter.html.

4.4.2.2 Technological environment

The Service Location Protocol is intended to function within networks under cooperative administrative control. Such networks permit a policy to be implemented regarding security, multicast routing and organization of services and clients into groups which are not be feasible on the scale of the Internet as a whole. The SLP has been designed to serve enterprise networks with shared services, and it may not necessarily scale for wide-area service discovery throughout the global Internet, or in networks where there are hundreds of thousands of clients or tens of thousands of services. The SLP uses standard internet protocols (UDP or TCP) and does not address topics specific to mobile or wireless environments. It makes use of unicast and multicast service. When multicast is not available, also broadcast may be used. The SLP introduces some network base load as service agents have to announce their services regularly to directory agents.

4.4.2.3 Security services

The Service Location Protocol addresses security by providing means to authenticate service URLs and service attributes. This provides User Agents and Directory Agents with knowledge of the integrity of service URLs and attributes included in SLP messages.

The Service Agents and User Agents may verify digital signatures provided with DAAdverts. User Agents and Directory Agents may verify service information registered by Service Agents. The keying material to use to verify digital signatures is identified using a SLP Security Parameter Index, or SLP SPI. Directory Agents do not sign service registrations or attribute lists. They simply cache services registered by Service Agents. However, DAs must not accept registrations including authentication blocks for SLP SPIs which it is not configured with.

The following types of content can be protected by a digital signature:

- URL entry
- Attribute List

- Directory Advertisement Message (DAAdvert)
- Service Agent Advertisement Message (SAAdvert)

While in the case of DAAdvert and SAAdvert a whole message is protected, in all other cases only selected fields are protected, an individual URL entry and an individual Attribute List.

In the case of a URL Entry, the digital signature is computed over the following information:

- SPI String
- URL
- Timestamp (validity of authentication information)

In the case of an Attribute List, the digital signature is computed over the following information:

- SPI String
- Attribute List
- Timestamp (validity of authentication information)

The Service Location Protocol uses the following messages:

- A service request *SrvRqst* message contains the type of requested service (as e.g. “service:tftp”), a scope list, and a predicate string describing the intended service. Furthermore a list of previous responders is included to prohibit them responding, and a SLP SPI describing the authentication block the response has to contain.
- A service reply *SrvRply* message contains a list of URL entries. Each URL entry can contain an authentication block.
- A service registration message *SrvReg* is sent by a Service Agent to Directory Agents. It contains an URL entry which can contain several authentication blocks, the service type that defines the URL’s service type regardless of the scheme of the URL, a list of scopes, and a list of attributes describing the service and associated attribute authentication blocks.
- A service acknowledgement *SrvAck* is sent to a Service Agent after it has registered itself with a *SrvReg* message (and also to acknowledge a deregistration, cf. *SrvDeReg* message below).
- A Directory Agent Advertisement *DAAdvert* announces the availability and properties of a Directory Agent. It contains the DA’s URL, and it can contain a scope list that designates the supported scopes, an attribute list that contains further information describing the DA as e.g. the minimum refresh interval of reregistrations, and a list of SPIs that a DA is capable of verifying. Service Agents must not register services with authentication blocks for SPIs that are not on the list. DAs will reject service registrations which they cannot verify. It is possible to sign the complete *DAAdvert* message.
- Similarly, when no Directory Agent is available and when a User Agent has not been configured with a scope list, a User Agent can solicit Service Agent Advertisements to discover the scopes that Service Agents support. The *SAAdvert* message contains the SA’s URL, can contain a list of supported scopes and a list of supported attributes. The complete *SAAdvert* message can be signed.
- A Service Type Request *SrvTypeRequest* message can be used by User Agents to discover all types of services on a network. It contains a list of previous responders, a scope list, and a naming authority of the registered service types (service types with a naming authority other than IANA have their name appended to the service type string, separated by a ‘.’ character).
- The corresponding response Service Type Reply *SrvTypeRply* contains a list of service types.

- An Attribute Request message *AttrRqst* allows a User Agent to discover all attributes of a given service or for an entire service type. It contains a list of previous responders, an URL, a scope list, a tag list of all attribute tags to be returned, and a SPI indicating the SPI of the attribute authentication blocks to be contained in the response. When a SPI is included, the list of tags must be omitted and a full URL has to be included.
- The corresponding response *AttrRply* contains a list of attributes, and if an SPI was contained in the request, the corresponding attribute authentication blocks are included.
- Using a Service Deregistration message *SrvDeReg*, a Service Agent can indicate to a Directory Agent that it wants to remove its registration before its lifetime expires. It contains a scope list, an URL entry. When no authentication blocks are used, it is also possible to deregister singular attributes. When a service that has been registered with an authentication block, a corresponding reregistration message or deregistration message must contain authentication blocks for each of the SPIs it has been previously registered.

Usually confidentiality is not provided by SLP. However, RFC2608 [7] recommends to use ESP (Encapsulated Security Payload) [12] in this case.

4.4.2.4 Technical security features

The Service Location Protocol Version 2 uses DSA signatures as specified in [15] to sign information fields. All SLP agents must implement DSA. The Service Agents must register services with DSA authentication. The signature format conforms to that in the X.509 v3 certificates, is ASN.1 encoded and has the following structure:

1. The signature algorithm identifier (an OID of id-das-with-sha1)
2. The signature value
3. The certificate path.

A Service Agent configured with the ability to sign service registrations must sign each of the URLs and Attribute Lists using each of the keys it is configured to use, and the Directory Agent it is registering with accepts. The Service Agent has to acquire DAAdverts for all Directory Agents it will register with to obtain the Directory Agent's SLP SPI list and attributes.

The incremental registration (changing some attributes) is not allowed for services registered with authentication blocks.

4.4.2.5 Bootstrapping security

When authentication is enabled, cryptographic keys and certificates are required. Service agents require a private key to sign their service advertisements and must have access to the certificates certifying their public key. Every host configured to generate a digital signature – be it a Service Agent ot a Directory Agent – includes the SLP SPI used to verify it in the Authentication Block it transmits. Every host, be it Directory Agent, a Service Agent, or a User Agent – which can verify a digital signature must be configured with keying material and other parameters corresponding with the SLP SPI such that it can perform verifying calculations. Furthermore the policies have to be defined for all entities – Service Agents, Directory Agents, and User Agents – that define their administratively set behavior, i.e. which SAs have to be signed and by whom. RFC2608 [7] provides mechanisms to authenticate service advertisements, but the prerequisites the involved security mechanisms require and the policy definition are out of scope.

The only systems which can generate digital signatures are those which have been configured by administrators in advance. Agents which verify signed data may assume its 'trustworthy' in as much as administrators have ensured the cryptographic keying of Service Agents and Directory Agents reflects 'trustworthiness'.

4.4.2.6 Trust model

The standard provides security mechanisms that allow to authenticate the origin and integrity of service URLs, attribute lists, and advertisement messages. It does not mandate the policy for the involved entities, i.e. which service advertisements to sign using which keys, and which one to accept dependent on signer and possibly the signed content.

4.4.2.7 Strengths and weaknesses

The precise security objectives are not made explicit. So it cannot be analyzed whether the proposed security mechanisms are in fact suitable to accomplish these objectives. So it depends on the point-of-view whether the following observations are actually security weaknesses or simply properties of the proposed mechanisms:

- Authentication is optional. Without using authentication, an attacker can easily provide fake information and launch denial-of-service attacks and routing users to an attacker's server (e.g. to capture sensitive data). An adversary might easily use SLP to advertise services on servers controlled by the adversary and thereby gain access to users private information.
- Although when the data contained in the service advertisement is protected, the access to the actual network service is not necessarily secured and in this case it is not ensured that the connected service is in fact the announced one. However, RFC2609 [8] defines service templates as a standardized format to register service templates with IANA.
- SLP provides information on the services available in a network. This information can be valuable for an attacker. Especially selective denial-of-service attacks are possible.

Only URLs and service attributes are authenticated, not the actual communication with service agents and directory agents. An attacker can easily deploy his own Directory Agent (when announced using DHCP, which is unsecured, or by impersonating the Directory Agent). The communication itself between User Agent and Directory Agent is not protected, so an attacker can operate a Directory Agent successfully without having access to any cryptographic keys and thereby control which User Agents find which services.

4.4.2.8 Future trends

The Service Location Protocol is available in its second version which corrected SLPv1 as specified in RFC2165 [21] by correcting protocol errors, adding some enhancements and removing some requirements. The appendix A of RFC2608 [7] summarizes the changes. Authentication has been reworked to provide more flexibility and protection, especially for DA advertisements. The SLPv2 appears to be a stable standard. However, it is unclear whether SLP has been deployed in operational environments.

It is assumed that evolution will concentrate on adding further services to be supported by SLP. This is done by the definition of Service Templates [8]. From this, service specific security issues could arise.

4.4.3 Zero Configuration Networking (ZEROCONF)

4.4.3.1 Description of the system

The goal of the ZERO CONFiguration Networking (ZEROCONF) Working Group is to enable networking in the absence of configuration and administration. Zero configuration networking is required for environments where administration is impractical or impossible, such as in the home or small office, embedded systems 'plugged together' as in an automobile, or to allow impromptu networks as between the devices of strangers on a train. The ZEROCONF protocols are intended to operate correctly without any configuration by a user or infrastructure services as e.g. DHCP [4] or DNS [13][14] servers. When configuration information is available it may be used, but it shall not be relied upon to be available.

The following functions which are a subset of problems to solve for networking in an environment without preexisting configuration are defined by the ZEROCONF working group:

- IP interface configuration:

  For each interface an IP address, network prefix, gateway router are required.

- Translation of name to address:

  Mapping of host names to IP address without DNS servers.

- Service discovery:

  The service discovery must allow a service to be discovered; discover via service identifier and/or service type; discover services without use of a service specific protocol and complete in a timely manner. Furthermore it should discover via service characteristics.

- Automatic allocation of mulitcast addresses:

  The multicast address must have a given scope, lease time, not allocated more than once within the scope of the address and become available for use again after the address expires or becomes deallocated.

It is intended to achieve a security level so that ZEROCONF networks are not less secure than networks that do not use ZEROCONF protocols. Only specific network topologies are considered, either a single network segment where all hosts are reachable by link-layer broadcast or multicast messages, or a set of network segments interconnected by a single router.

The ZEROCONF protocols intends to make networking as easy as possible. However, in some cases other considerations may dominate ease of use. For example, network security requires some configuration.

The charter of the ZEROCONF working group and published RFCs and Internet drafts can be found at http://www.ietf.org/html.charters/zeroconf-charter.html. At the time of this writting, no RFC is available, but three Internet drafts that describe requirements (draft-ietf-zeroconf-reqts-07.txt) and proposals for the dynamic configuration of an IPv4 address for a local link (draft-ietf-zeroconf-ipv4-linklocal-02.txt) and for the allocation of multicast addresses (draft-ietf-zeroconf-zmaap-00.txt). Furthermore, the RFC2462 [20] “IPv6 Stateless Address Autoconfiguration” which is related to the Internet draft “draft-ietf-zeroconf-ipv4-linklocal-02.txt” exists, but it has been developed before the ZEROCONFIG working group started.

4.4.3.2 Technological environment

Many common TCP/IP protocols such as DHCP [4], DNS [13][14], MADCAP [9], and LDAP [22] must be configured and maintained by an administrative staff. This is unacceptable for emerging networks such as home networks, automobile networks, airplane networks, or ad hoc networks at conferences, emergency relief stations, and many others. Such networks may be nothing more than two isolated laptop PCs connected via a wireless LAN or link-local-only devices which can be very simple devices of the kind that currently use USB or Firewire. For all these networks, an administrative staff will not exist and the users of these networks neither have the time nor inclination to learn network administration skills. Instead, these networks need protocols that require zero user configuration and administration, and can be used jointly with configured information e.g. use both link-local IP address and routable IP address on the same interface.

4.4.3.3 Security services

The Internet draft stating the requirements on ZEROCONF protocols contains also a section dealing with security requirements. The overall goal is to achieve a security level which is not less secure than when common Internet protocols would be used. This means especially when the commonly used alternatives that require configuration are already insecure, it is not required that ZEROCONF protocols are any more secure.

The security requirements of the functions considered in ZEROCONF are the following:

- IP Interface Configuration:

The IP communication relies on lower level address resolution protocols, which are ARP [18] for IPv4 and Neighbor Discovery [16] for IPv6. In the case of IPv4 it is acceptable – though not desirable – to use no security for IP interface configuration as the existing IPv4 address resolution mechanisms (ARP and variants) are already insecure. In IPv6 however, the Neighbor Discovery Protocol can be secured using IP Security mechanisms and therefore in this case it must be possible to secure IP interface configuration.

Another aspect is that conforming implementations must not route packets containing link-local addresses. Without some form of gateway functionality, which would have to provide adequate security, devices which have only a link-local IP address can assume that they communicate only with other local devices. This assumption on the local environment can have security implications, e.g. a mouse or a keyboard will usually not authenticate and encrypt mouse movements or keyboard strokes, respectively.

- Name to Address Resolution:

  Usually, DNS is used for mapping host names to IP addresses. As DNS is unsecured, it can be subverted in many ways, e.g. by pointing a client to a wrong DNS server or by sending fake DNS replies. However, DNSsec [5][6] allows a client to verify whether received data has been signed by a source it has been configured to accept. Therefore ZEROCONF requires that a ZEROCONF name to address resolution protocol must be compatible with DNSsec which means that it must be also possible to use DNSsec for authenticated name resolution when a host or its administrator chooses to do so.

- Service Discovery:

  The standard service discovery protocol ZEROCONF refers to is the Service Location Protocol, Version 2 (SLPv2) [7]. SLPv2 messages that contain answers can contain authorization blocks that allow a client to check the integrity and source. Therefore also a ZEROCONF service discovery protocol must allow a client to verify that a service advertisement sent by a server was created by an authorized source.

- Multicast Address Allocation:

  The threat of using an insecure Multicast Address Allocation protocol is that an active attacker could get all or a huge number of multicast addresses. This would prevent other hosts from effectively participating in the Multicast Address Allocation protocol. The correspondent protocols in a configured environment AAP (draft-ietf-malloc-aap-04.txt) and MADCAP [9] do not include security mechanisms themselves. However they suggest to use IPsec to ensure message integrity and end-point authentication. Therefore ZEROCONF requires that a ZEROCONF protocol for multicast address resolution must either be compatible with IP Authentication Header or another authentication mechanism.

However, the ZEROCONF protocols of the proposed Internet drafts for IPv4 link-local IP address and multicast address allocation do not provide security services.

4.4.3.4 Technical security features

In the current development status, no technical security features specific to ZEROCONF networking exist.

4.4.3.5 Bootstrapping security

In the absence of security services not applicable.

4.4.3.6 Trust model

There are no security mechanisms within ZEROCONF. The scenario where ZEROCONF plays a major role is in restricted environments. Restricted environments have only a network where all participating hosts are directly reachable or only one single router exists. The trust model of ZEROCONF assumes that all participating hosts are well-behaving.

4.4.3.7 Strengths and weaknesses

Some weaknesses of ZEROCONF are:

- The overall goal is to achieve a security level which is not less secure than when common Internet protocols would be used. This means especially when the commonly used alternatives that require configuration are already insecure, it is not required that ZEROCONF protocols are any more secure.
- IP local link address: When conflicts occur (two hosts with same IP local link address), a host detecting this must configure to a new link-local IP-address. An attacker can force another host to relieve its IP address. However, when some security context based on IP address is associated with this IP address (e.g. IP address is associated by a server with a user after user has authenticated himself by password entry), the attacker can masquerade as the user.
- A denial of service by requesting all IP addresses; seems to be possible even when all exchanged messages would be protected.

4.4.3.8 Future trends

Though there is no security services available within ZEROCONF the IPv4 local link address is already implemented by WIN98 and Mac OS 8.5 clients and later.

4.4.4 References

[1] S. Bradner: The Internet Standards Process – Revision 3. Internet Request for Comments RFC 2026, Oct. 1996. http://www.ietf.org/rfc/rfc2026.txt

[2] S. Corson, J. Macker: Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. Internet Request for Comments RFC 2501, Jan. 1999. http://www.ietf.org/rfc/rfc2501.txt

[3] M. Scott Corson, Joseph P. Macker, Gregory H. Cirincione: Internet-Based Mobile ad-hoc Networking. Internet Computing 3(4):63–70, IEEE, Jul./Aug. 1999.

[4] R. Droms: “Dynamic Host Configuration Protocol”. Internet Request for Comments RFC 2131, March 1997. http://www.ietf.org/rfc/rfc2131.txt

[5] D. Eastlake: “Domain Name System Security Extensions”. Internet Request for Comments RFC 2535, March 1999. http://www.ietf.org/rfc/rfc2535.txt

[6] D. Eastlake: “DNS Request and Transaction Signatures (SIG(0)s)”. Internet Request for Comments RFC 2931, Sep. 2000. http://www.ietf.org/rfc/rfc2931.txt

[7] E. Guttman, C. Perkins, J. Veizades, M. Day: “Service Location Protocol, Version 2”. Internet Request for Comments RFC 2608, Jun. 1999. http://www.ietf.org/rfc/rfc2608.txt

[8] E. Guttman, C. Perkins, J. Kempf: “Service Templates and Service: Schemes”. Internet Request for Comments RFC 2609, Jun. 1999. http://www.ietf.org/rfc/rfc2609.txt

[9] S. Hanna, B. Patel, M. Shah: “Multicast Address Dynamic Client Allocation Protocol (MADCAP)”. Internet Request for Comments RFC 2730, Dec. 1999. http://www.ietf.org/rfc/rfc2730.txt

[10] S. Jacobs, M. S. Corson: MANET Authentication Architecture. IETF work in progress (expired), Jan. 1999. draft-jacobs-imep-auth-arch-01.txt.

[11] S. Kent, R. Atkinson: “Security Architecture for the Internet Protocol”. Internet Request for Comments RFC 2401, Nov. 1998. http://www.ietf.org/rfc/rfc2401.txt

[12] S. Kent, R. Atkinson: “IP Encapsulating Security Payload (ESP)”. Internet Request for Comments RFC 2406, Nov. 1998. http://www.ietf.org/rfc/rfc2406.txt

[13] P.V. Mockapetris: “Domain names - concepts and facilities”. Internet Request for Comments RFC 1034, Nov. 1987. http://www.ietf.org/rfc/rfc1034.txt

[14] P.V. Mockapetris: “Domain names - implementation and specification”. Internet Request for Comments RFC 1035, Nov. 1987. http://www.ietf.org/rfc/rfc1035.txt

[15] National Institute for Standards and Technology: “Digital Signature Standard”, NIST FIPS PUB 186, May 1994. http://csrc.nist.gov/publications/fips/

[16] T. Narten, E. Nordmark, W. Simpson: “Neighbor Discovery for IP Version 6 (IPv6)”. Internet Request for Comments RFC 2461, Dec. 1998. http://www.ietf.org/rfc/rfc2461.txt.

[17] C. Perkins, E. Guttman: “DHCP Options for Service Location Protocol”. Internet Request for Comments RFC 2610, June 1999. http://www.ietf.org/rfc/rfc2610.txt

[18] D.C. Plummer: “Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware”. Internet Request for Comments RFC 826, Nov. 1982. http://www.ietf.org/rfc/rfc0826.txt

[19] R. Rivest: “ The MD5 Message-Digest Algorithm“. Internet Request for Comments RFC 1321, April 1992. http://www.ietf.org/rfc/rfc1321.txt

[20] S. Thomson, T. Narten: “IPv6 Stateless Address Autoconfiguration”. Internet Request for Comments RFC 2462, Dec. 1998. http://www.ietf.org/rfc/rfc2462.txt

[21] J. Veizades, E. Guttman, C. Perkins, S. Kaplan: “ Service Location Protocol“. Internet Request for Comments RFC 2165, June 1997. http://www.ietf.org/rfc/rfc2165.txt

[22] W. Yeong, T. Howes, S. Kille: “X.500 Lightweight Directory Access Protocol”. Internet Request for Comments RFC 1487, July 1993. http://www.ietf.org/rfc/rfc1487.txt

[23] S. Kent, R. Atkinson: Security Architecture for the Internet Protocol. Internet Request for Comments RFC 2402, November 1998. http://www.ietf.org/rfc/rfc2402.txt

4.5 UPnP

4.5.1 Description of the system

The UPnP (Universal Plug and Play) [1][2] is a further development of PnP (Plug and Play) which provides a user of a PC system with easier setup, configuration, and adding of peripherals. Thereas UPnP extends this simplicity to include the entire network, enabling discovery and control of devices, including networked devices and services, such as network-attached printers, Internet gateways, and consumer electronics equipment with support of zero-configuration. The scope of UPnP encompass scenarios like home automation, printing and imaging, audio/video entertainment, kitchen appliances, automobile networks, and proximity networks in public venues.

The basic building blocks of a UPnP network are devices, services and control points [Figure 14]. The UPnP Device Architecture defines a schema or template for creating device and service descriptions for any device or service type. UPnP does not specify the APIs applications will use.

A UPnP device is a container of services and nested devices. All of this information is captured in an XML device description document that the device must host. Devices may contain multiple services. A pointer (URL) to these service descriptions is contained within the device description document.

The smallest unit of control in a UPnP network is a service. A service exposes actions and models its state with state variables. As in the device case the service information is captured in an XML service description. A service in a UPnP device consists of a state table, a control server and an event server. The state table models the state of the service through state variables and updates them when the state changes. The control server receives action requests (such as set_time), executes them, updates the

state table and returns responses. The event server publishes events to interested subscribers anytime the state of the service changes.

The control point in a UPnP network is a controller capable of discovering and controlling other devices. After discovery, a control point could: Retrieve the device description and get a list of associated services; retrieve service descriptions for interesting services; invoke actions to control the service; or subscribe to the service`s event source e.g. when the service state changes.

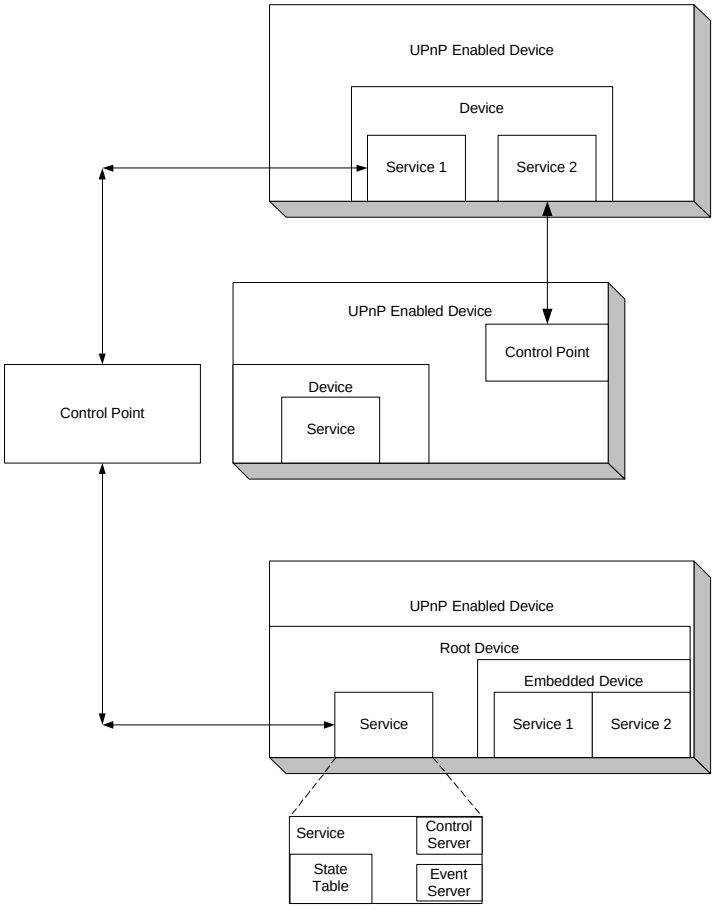


Figure 14: The building blocks of UPnP

As the entities - used in a UPnP network - makes it clear that there are several steps involved:

1. Addressing – The foundation for UPnP is the TCP/IP protocol suite. Each device must have a Dynamic Host Configuration Protocol (DHCP) client and search for a DHCP server when the device is first connected to the network. If no DHCP server is available, the device must use Auto IP to get an address.
2. Discovery – Once devices are attached to the network and addressed appropriately discovery can take place. Discovery is handled by the SSDP. The fundamental exchange in both cases is a discovery message containing a few, essential specifics about the device or one of its services, e.g. type, identifier, and a pointer to its XML device description document.
3. *Description – After a control point has discoverd a device, the control point still knows very little about the device. The control point must retrieve the device's description from the URL provided by the device in the discovery message. The UPnP description for a device is expressed in XML and includes vendor-specific, manufacturer information including the model name and number, serial number, manufacturerer name, URLs to vendor-specific Web sites, and so forth. The*

description also includes a list of any embedded devices or services, as well as URLs for control, eventing, and presentation.

4. Control – To learn more about the service, a control point must retrieve a detailed UPnP description for each service. The description for a service is also expressed in XML and includes a list of the commands, or actions, the service responds to, and parameters or arguments, for each action. The description for a service also includes a list of variables; these variables model the state of the service at run time, and are described in terms of their data type, range, and event characteristics. Control messages are also expressed in XML using SOAP. In response to the control message, the service returns action specific values or fault codes.
5. Eventing – The service publishes updates when variables which models the state of the service at run time change. A control point may subscribe to receive this update information. The service publishes updates by sending event messages. These messages are also expressed in XML and formatted using GENA. A special initial event message is sent when a control point first subscribes; this event message contains the names and values for all evented variables and allows the subscriber to initialize its model of the state of the service.
6. Presentation – The control point can retrieve a page if there is a device URL. The page can be loaded into a browser, and depending on the capabilities of the page, allow a user to control the device and/or view device status.

There is a UPnP Forum which is a group of companies and individuals across the industry that intend to play a leading role in the authoring of specifications for UPnP devices and services. The Forum aims to enable the emergence of easily connected devices and to simplify the implementation of networks in the home and corporate environments. The UPnP Forum up to the writting of this text has 334 company members. The Forums`s web site, http://www.upnp.org, is the central repository for schema that has been developed and standardized by the Forum. In addition, the site includes the device architecture document, templates for device and service descriptions, and guidelines for device and service description design. The web site also distributes information about the Forums`s activities and progress.

4.5.2 Technological environment

UPnP provides support for communication between control points and devices. The network media, the TCP/IP protocol suite and HTTP provide basic network connectivity and addressing needed. On top of these open, standard, Internet based protocols, UPnP defines a set of HTTP servers to handle discovery, description, control, events, and presentation. Figure 15, below, illustrates this interworking of serveral protocols.

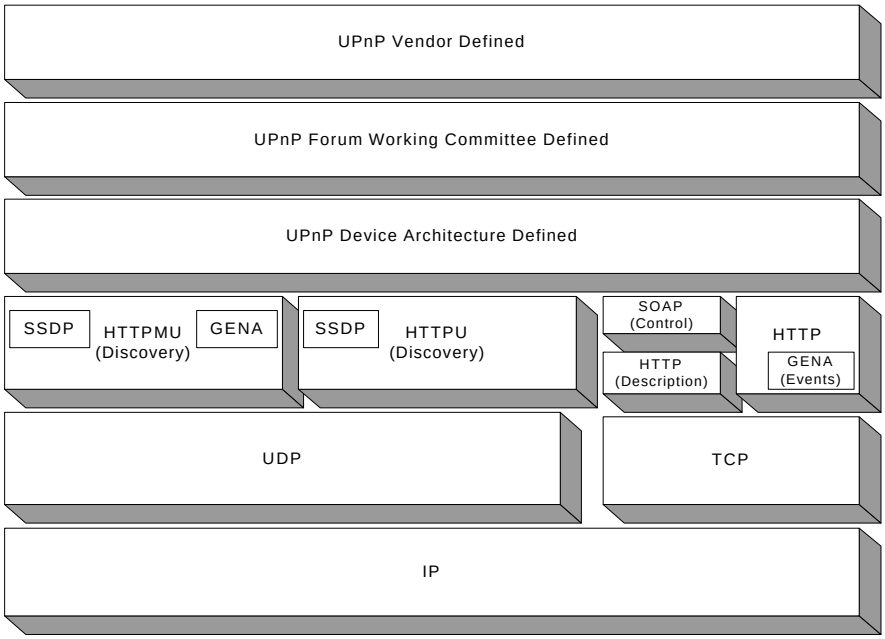


Figure 15: UPnP Protocol Stack

A description of protocols used and their relation to UPnP are given next:

- TCP/IP (Transmission Control Protocol/Internet Protocol) – The TCP/IP networking protocol stack serves as the base on which the rest of UPnP protocols are built.
- HTTP (HyperText Transport Protocol) – all aspects of UPnP are build on top of HTTP or its variants.
- HTTPU (HTTP over UDP) and HTTPMU (HTTP Multicast over UDP) – deliver messages on top of UDP/IP instead of TCP/IP. These protocols are used by SSDP (see below).
- SSDP (Simple Service Discovery Protocol) – defines how network services can be discovered on the network. SSDP is built on HTTPU and HTTPMU and defines methods both for a control point to locate resources of interest on the network, and for devices to announce their availability on the network. A UPnP control point, upon booting up, can send an SSDP search request (over HTTPMU), to discover devices and services that are available on the network. The UPnP device listen to the multicast port. Upon receiving a search request, the device examines the search criteria to determine if they match. If a match is found, a unicast SSDP (over HTTPU) response is sent to the control point. Similarly, a device, upon being plugged into the network, will send out multiple SSDP presence announcements advertising the services it supports.
- GENA (Generic Event Notification Architecture) – define the ability to send and receive notifications using HTTP over TCP/IP and multicast UDP. A control point interested in receiving event notifications will subscribe to an event source by sending a request that includes the service
- of interest, a location to send the events to and a subscription time for the event notification. The subscription must be renewed periodically to continue to receive notifications, and can also be canceled using GENA.
- SOAP (Simple Object Access Protocol) – defines the use of XML and HTTP to execute remote procedure calls. Each UPnP control request is a SOAP message that contains the action to invoke along with a set of parameters. The response is a soap message as well and contains the status, return value and any return parameters.

4.5.3 Security services

The current version 1.0 of UPnP includes no security services at all. It is focused on the home or small office where physical security is the first line of defense. Security services including authentication and authorization are subject to discussion for support in future versions. Apart from this SOAP can also make use of Secure Sockets Layer (SSL) for security and use HTTP's connection management facilities.

4.5.4 Trust model

There are no security mechanisms within UPnP. The scenario where UPnP plays a major role is in restricted environments. Restricted environments have only a network where all participating hosts are directly reachable or only one gateway exists. The trust model of UPnP assumes that all participating devices, services and control points are well-behaving and trust each over.

4.5.5 Strengths and weaknesses

The strengths of UPnP are:

- Because UPnP is a distributed, open network architecture, defined by the protocols used, it is independent of any particular operating system, programming language, or physical medium (just like the Internet).
- The TCP/IP protocol suite and open standard protocols are used. Therefore only limited resources are necessary particularly on the server side.

Beneath the strengths of UPnP there are some weaknesses:

- There exists no mechanism to define predicates like in SLP in order to have a sufficient service search granulary. Therefore scalability is a problem especially in huge networks with much administration.
- As mentioned in the security services section no security aspects are considered or defined in the specification.

4.5.6 Future trends

At the web page of the UPnP Forum there is mentioned that security aspects will be get relevant in the future work. It is not clear when and which extent the security aspects take in the UPnP work.

4.5.7 References

[1] Microsoft Corporation, Universal Plug and Play Device Architecture, 2000, http://www.upnp.org/download/UPnPDA10_20000613.htm

[2] Microsoft Corporation, 2000, Understanding Universal Plug and Play, Whitepaper, http://www.upnp.org/download/UPNP_UnderstandingUPNPTomFRaviRaoEditsFinal.doc

4.6 Java and Jini

4.6.1 Description of the system

4.6.1.1 Java

Java is a modern, object-oriented programming language that has been designed to be platform-independent, i.e. Java code that has been compiled once (the so-called bytecode) should run on virtually any computing platform. This has been achieved by introducing a Java Virtual Machine (JVM) as an additional interface between the machine and the compiler. The platform-independence

and the relative compactness of the Java bytecode makes it especially suited for use in heterogenous environments where bandwith is limited, i.e. the Internet.

By design, the Java programming language is more secure than others, like C++, are. But because it was also, by design, meant to be used in an inherently opened and chaotic environment like the Internet is, it faces new kinds of challenges that were not before addressed by any other programming language.

4.6.1.2 Jini

Jini is based on the idea of federating groups of users and groups of services that the users need and is built on top of Java. The goals of the system as described by the Jini Architecture specification [1] include the following:

- To enable users to use and share services over a network
- To provide location independent access to services available on the network
- To make the task of building, maintaining and changing a network of devices services less problematic.

Jini uses concepts already existing in Java such as mobile code, strongly typed interfaces and the separation of interface and implementation and adds some extra elements like leasing. The five key concepts of Jini are Discovery, Lookup, Leasing, Remote Events and Transactions.

The Jini infrastructure is made up of a number of components that facilitate the concepts of joining and leaving a federation or workgroup. Before describing the components, the concept of a service needs to be defined. According to the Jini Architecture specification “a service may be a computation, storage, a communication channel to another user, a software filter, a hardware device, or another user”. The members of a Jini system federate to share access to services, so the system can be viewed as collection of services that have the potential to enable a member of the federation perform a particular task within a particular period of time.

A service is essentially a Java object (which can be made up of other objects) that has an interface that defines the operations that can be requested of the service. Services make use of the Jini infrastructure to discover each other, to make calls to each other and to announce their presence to other users and services. Examples of possible services are a JavaSpaces [2] service that could be used for simple communication and storage of related groups of objects written in Java or a printing service, which could print from Java applications and legacy applications.

The Jini lookup service is the central component of Jini’s runtime infrastructure. It offers Jini clients a way to find Jini services and service providers a method to advertise their services. There are a number of components required to use the lookup service: the service registrar and *discovery* and *join* protocols (See Figure 16 and Figure 17). The discovery protocol locates an appropriate lookup service and the join protocol is used to join that lookup service. Discovery and join takes place when a device is plugged in. Users of the system can then access the services available from the device (See Figure 18). This process works by matching functionality indicated by interfaces to objects that implement the specified functionality. Once the client has located a service, service objects are loaded into the client. The client is now in a position to invoke the service. Communication between services takes place using Remote Method Invocation (RMI) RMI allows the passing of full objects including code around the network.

The lookup service makes use of the leasing and event interfaces described below. Leasing ensures that services registered are available and events help in detecting problems with discovery and device configuration. Leases can be renewed if the time of required service access exceeds the duration of the lease.

Access to services in the Jini system is based on leases. A lease guarantees access over a fixed period of time. Leasing is part of the service protocol and takes place when a service is requested for a particular period of time. Access to the service may then be granted for that period. Leasing is

implemented by holding onto a reference to a resource for a particular period of time. When the period of time is over the reference is no longer valid.

A transaction is an optional extension to the basic Jini system. It can be defined as a series of operations that result in the system moving from a consistent state to another consistent state. The transaction interface enables Jini applications to co-ordinate state changes. It uses the "*two phase commit*" approach for the group of objects involved in the application. This involves one round of voting on the possible results of the transaction and then a co-ordinator decides if the group should commit. The implementation of the desired semantics of the transaction is left up to the designer of the particular objects involved in the transaction.

The Jini architecture supports distributed events. The event and notification interfaces enable event-based communication between Jini services by extending the standard event model used by JavaBeans. The JavaBeans component event model allows events to be handled by third-party objects and ensures event delivery within a particular period of time.

The Jini model introduces the notion of groups to distinguish and identify virtual networks. Any device (client), service provider and lookup service can claim to belong to one specific, many specific or all possible groups. The belonging to one or more (or all) groups is specified by the owner or administrator of the entity (client, service, lookup service). There is no a priori restriction as to what groups an entity can belong to.

The Jini model distinguishes the following phases:

- Discovery phase: The service provider searches for available lookup services serving the group(s) to which the service provider claims to belong to. All lookup services serving the requested groups reply and state that the service can register with the lookup service.

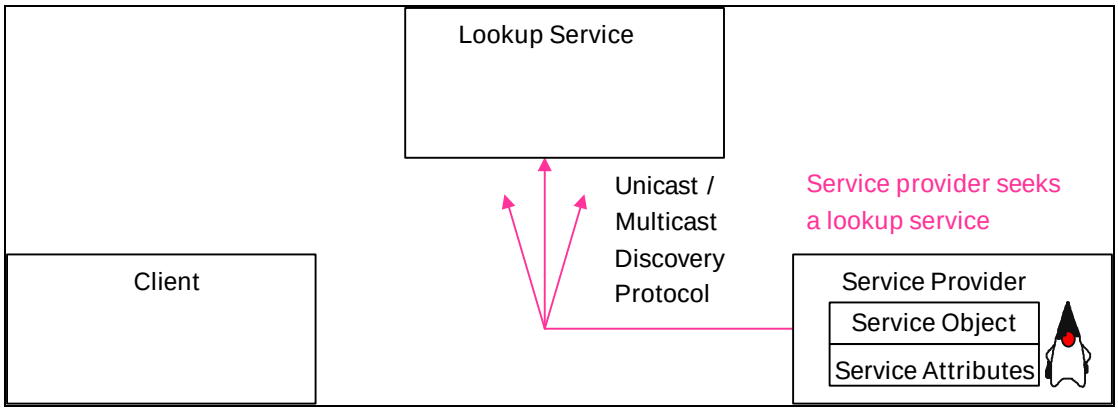


Figure 16: Discover phase.

- Join phase: The service provider registers a service object (proxy) and its attributes at the lookup services of his choice or at all lookup services serving the group(s) to which the service claims to belong to.

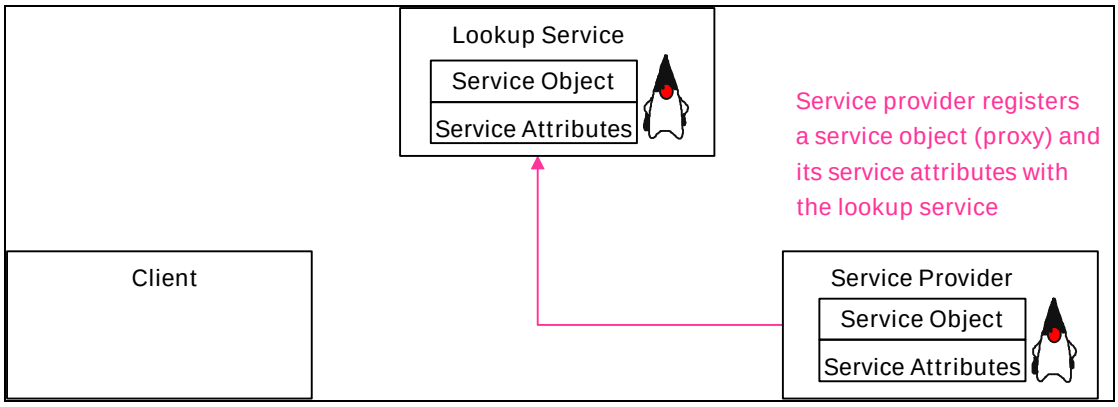


Figure 17: Join phase.

- Lookup phase: The client searches for all lookup services serving the group(s) to which the client claims to belong to. The lookup services reply and inform the client about what services are registered with the lookup services that serve the group(s) to which the client belongs. The client chooses the service and receives the service object (proxy) and perhaps its attributes from the lookup service. In addition, the client may register a callback service at the lookup service to get informed about changes concerning the services available on the lookup server.

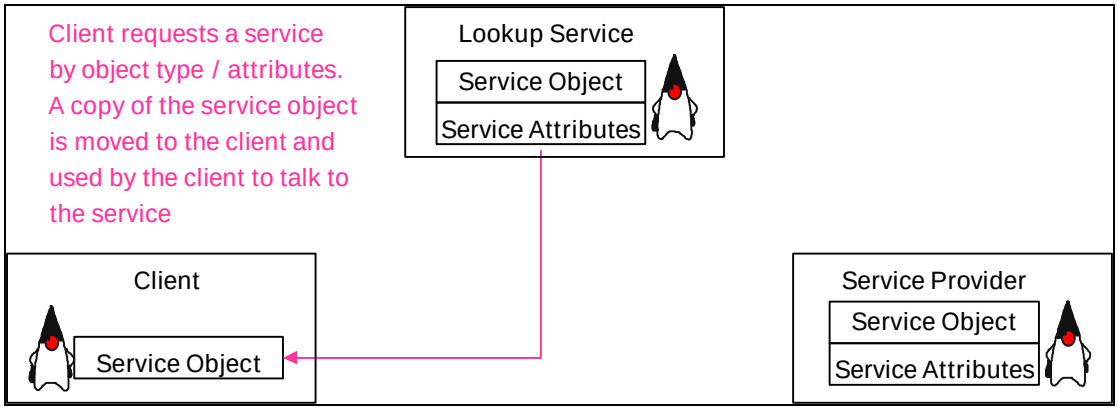


Figure 18: Lookup phase.

- Service invocation phase: The client uses the service object (proxy) received by the lookup service to interact directly with the service. The lookup service is not used anymore.

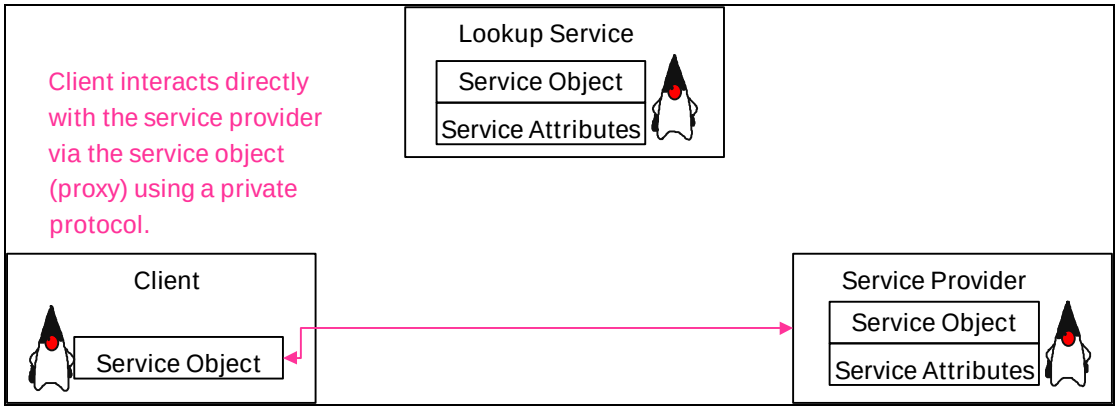


Figure 19: Service invocation phase.

4.6.2 Technological environment

Because of its platform independence, Java is used not only as a programming language for the World-Wide Web, but is also becoming increasingly popular for applications in telecommunications. Standardised Java-based Application Programming Interfaces (APIs) provide the possibility to create services that can seamlessly be used in fixed, mobile and IP based networks. One example for this is JAIN (Java API for Integrated Networks), which comprises a set of different Java-based APIs for various protocols used in the SS7-based and IP-based world.

The vision of Jini is to allow hardware and applications to interoperate in a spontaneous, dynamic, yet robust way, which could also be called "spontaneous networking". However, Jini is not yet widely deployed, despite of the fact that the Jini architecture may have applications far broader than the device-connection technology toward which SUN originally targeted it [3]. One reason for this might be that Java is required to run Jini and it is still a Sun controlled platform and not an ISO standard. This means that users of Jini are relying on the benevolence of a single company.

4.6.3 Security services

4.6.3.1 Security Services of Java

The Java Programming Language has been designed from the ground up with security in mind. Security of the language has several levels of abstractedness.

4.6.3.1.1 Language level

Language level security features of Java are performed at compile time. They can be summarised as follows:

- Invalid memory access: no pointer arithmetic is allowed in Java. This restriction avoids inappropriate memory access and higher program robustness;
- Garbage collection: when a dynamic memory area has no reference to it, the garbage collection automatically puts this area in the pool for being claimed by another object. This feature releases the programmer from determining exactly when to return a piece of memory;
- Data type checking: In Java, data type checking in done at compile time, and ensures that no illegal casts are done. This feature ensures that no block of memory is interpreted as anything else than what it really is;
- Access modifiers: Control of data members', methods' and inner classes' visibility is done by access modifiers (public, protected, private and a default mode that can be called friendly);

4.6.3.1.2 Java Virtual Machine level

The Java Virtual Machine (JVM) specification defines how class parts (variables, methods, etc.) are stored in Java class files. So, any programming language[1] can be used as an input to a compiler that transforms it into perfectly legal Java byte codes. This means that it is not enough to have a language level security, but a JVM level is needed.

This JVM-level security is implemented by doing byte code verification. This verification is done in four passes:

- Pass one: magic number checking (to ensure that it is indeed a .class file), and other verifications (without running through the code);
- Pass two: runs through the code, checking if superclass (for classes that don't inherit from the Object class) is present;

[1] [SUN1] mentions a list of cross-compilers.

- Pass three: verify if method arguments are of the appropriate type;
- Pass four: invoked only when a method is called, checks for member access (public, private or protected).

Code verification is called after the class has been loaded into the JVM and before any of its methods is ran.

4.6.3.1.3 Class loader level

Whenever a class is needed, an instance of a class named ClassLoader (or some subclass of it) is called

Class loader level security embraces the following features:

- Restrictions for class loading: the class loader will not load any classes in java.* packages from over the network, since these could try to fool the local core libraries' classes;
- Separate name spaces for classes loaded from different hosts: classes loaded from different hosts will not communicate within the JVM-space, therefore avoiding untrusted programs to get some information from trusted ones.

4.6.3.1.4 Run time level

After having been successfully verified, there is still a run-time checking:

- Additional run-time checking of assignments and array bounds: due to Java's late binding not every type checking can be done at compile time. It is therefore needed to check these aspects again,
- Array bound checking: upper and lower bounds of arrays are allways checked whenever accessed;
- Fine grained access control: access to every local resource can be checked at run-time for permissions, using the security manager that every JVM must have installed. This will be further detailed below.

4.6.3.1.5 Architectural aspects

Security architecture in Java has tremendously evolved since its first steps. This sub-section describes that evolution and the perils that still exist.

- Security models

The very first security model chosen, the so-called "sandbox" model, was an all or nothing approach: code that was downloaded from the net into a JVM (like in an Applet) had a restricted environment where it could run. Local Java code could run without any restriction. So security in done from the point of view of where did the code came from.

Then, JDK1.1 added the notion of "signed applets": digitally signed (and therefore trusted) code could run without restrictions, while unsigned code was submitted to the sandbox model. Signed applets (and their signatures) are distributed in Java ARchive (JAR) files

In JDK1.2 a number of improvements were introduced. Regardless of its origin (local or remote), code can now be subjected to a security policy. This policy allows for a user or a system administrator to define a set of permissions for code coming from several signers or locations. These permissions specify what kind of access a particular resource (a file, a directory, a network connection, a port, etc.) may have.

- Java Cryptography Architecture

Java cryptography architecture (JCA) refers to a framework for accessing and developing cryptographic functionality for the Java platform that was introduced in the first release of JDK1.1.

The JCA includes a provider architecture that allows for multiple and interoperable cryptography implementations [4]. The Cryptographic Service Provider (CSP) in JDK1.2 supplies a concrete implementation for the following services:

- one or more digital signature algorithms;
- message digest algorithms;
- key generation algorithms;
- keystore creation and management;
- algorithm parameter management;
- algorithm parameter generation;
- key factory support to convert between different key representations;
- certificate factory support to generate certificates and certification revocation lists from their encoding.

The cryptographic functionality is split between the Java Security core classes (java.security.*) and the Java Cryptographic Extensions (java.crypto.*). This split was made because US Government was happy to export classes for authentication and integrity , but was not so happy to export classes for encryption.

- Java Authentication and Authorisation Service

The Java Authentication and Authorisation Service (JAAS), released in the Java 2 platform, is a framework and programming interface that adds to the Java Platform both user-based authentication and access control capabilities [5].

The advantages of using the JAAS center around the notion of pluggable authentication and principal[2]-based access control capabilities, without requiring modifications to the Java 2 core [5]. JAAS, together with Java 2, allows for the use of code-centric security, user-centric security or both, as needed.

JAAS is seen as being a good support for introducing security at the RMI level, the Jini level or even at the JVM level, making logging in by the VM user mandatory.

4.6.3.2 Jini Security Services

Jini currently uses the same security services as all Java 2 programs. Besides that, no addional security services are foreseen.

4.6.4 Strengths and weaknesses

4.6.4.1 Java

The Java programming language has evolved to face the naturally unsafe and hostile Internet environment. We have seen that Java (language and platform) posesses both language level and architectural level security services to deal with the security problem. However, in the following we describe two of the known security holes that exist in the language. More information can be found in http://www.securingjava.com .

4.6.4.1.1 Serialisation

From the instant an object is serialised and until it is de-serialised in a specific JVM, the object is outside of the control of the Java run-time environment [6].

One possible “countermeasure” to this might be to encrypt the serialised object and decrypt it when de-serialising it. But this implies managing keys, their location and the way these keys are distributed to the de-serialisation programs. De-serialisation is a type of object creation, therefore allowing for imposing this kind of operation some restrictions whenever un-trusted code serialises an object.

[2] A principal represents a name associated with a subject, which is any user of a computing service.

4.6.4.1.2 Native methods

Native methods, as defined by the Java Native Interface (JNI) in [4], are methods written in another language then Java (such as C, C++ and assembly), stored in a library and called by Java code that runs in a JVM. By using the JNI, programmers can update programs in the library and vendors can update the JVM without any of them having to recompile and/or relink the code.

So the library where a native method resides can be easily replaced with another one containing some form of malevolent method that can then act freely as the old (potentially benign) one.

4.6.4.2 Jini

Jini is an innovative and useable technology for building reliable, fault-tolerant distributed applications. For a small home environment the currently available infrastructure might be adequate, but for mission critical applications security is a must. Unfortunately, this area is currently untouched by Jini.

By itself, Jini does **not** provide any security. It rather builds upon the existing infrastructure given by the Java 2 classes. The security features of Java 2 include fine-grained control of the "sandbox model" – code gets specific access rights depending on where the code came from and who signed it. The access granted to a set of classes is specified in policy files. Jini introduces one new permission that can be granted to classes, namely DiscoveryPermission, which controls access to the multicast request and announcement protocols. For the Jini classes to perform their duties correctly they have to be given a few permissions to access sockets and properties.

An important part of a Jini federation is code mobility. Proxies in Jini are not simple stubs for remote procedure calls, but can contain parts of or even the whole functionality of the service they represent. Code mobility is an inherent feature of RMI. The code for proxies is usually supplied by an HTTP server on behalf of the service. The security policy applies especially to these proxies, as code download is a potential security hole.

As mentioned before, current Java security does only allow access to local resources based on where code came from and who signed it. This information is enough in a rather static environment like web interactions or centralized business applications. In the dynamic world of interacting services an important additional variable in performing access decisions is who is running the code. Granting access to resources based on on whose behalf a piece of code is running is currently not possible with a standard Java 2 installation.

Another potential security hole is present in the architecture described above – the lookup service itself might be malicious. Therefore the lookup service has to be authenticated before any of its proxy code is executed. Again, Jini itself does not provide any services to support such an authentication.

In addtion, the complete lack of a other standard security services such as confidentiality or integrity protection of mobile code leaves the door wide open for intruders in all phases of the Jini Model (cf. 3.6.1.2).

4.6.5 Future trends

While Java has itself firmly established as one of the most widely-used programming languages, the future of Jini is quite unclear. It is not yet widely deployed and there is a number of competing approaches (e.g. E-speak, cf. 3.9) aiming in the same direction.

The lack of security features for Jini is seen as serious problem by many researchers. For example, there have been attempts to secure Jini-based spontaneous networks by using SSL for authentication and encryption. One drawback of such a certificate-based approach is the needed a priori knowledge of the certification authority's public key. In a spontaneous networking system it would be optimal not to need any information in advance. It is not certain, though, if this is possible.

4.6.6 References

[1] Jini Architecture Specification: http://www.sun.com/jini/whitepapers/architecture.html

[2] JavaSpaces Specification: http://java.sun.com/products/javaspaces/

[3] Waldo, J.: *Alive and Well: Jini Technology Today*, IEEE Computer, June 2000, pp.107 – 109.

[4] Java Native Interface Specification: http://java.sun.com/products/jdk/1.2/docs/guide/jni/spec/jniTOC.doc.html

[5] Lai, C., Gong, L., Koved, L., Nadalin, A., Schemers, R., "User Authentication and Authorisation in the Java™ Platform", Proceedings of the 15th Annual Computer Security Applications Conference, 1999.

[6] Security Code Guidelines Security in Java 2 SDK 1.2, The Java Tutorial, http://java.sun.com/docs/books/tutorial/

4.7 WAP

4.7.1 Description of the system

The intention of the WAP protocol stack and the WAP environment is to offer the possibility for mobile devices to access Internet services. The development and evolution of WAP is controlled by the WAP-forum (http://www.wapformum.org). The WAP consortium groups all important telecommunication and computer infrastructure OEM's with the intention to develop standards that will be accepted and implemented by a large community of OEMs.

The WAP protocol stack as it is used today, is basically an implementation of the WAE (Wireless Application Environment). The goal of WAP is to bring Internet content and advanced data services to wireless phones and other wireless terminals. An overview of the WAE environment is given in Figure 20. The WAE model adopts a model that closely follows the WWW model. All content is specified in formats that are similar to the standard Internet format. Content is transported using standard protocols in the WWW domain and an optimised HTTP-like protocol (the WAP protocol) in the wireless domain.

In addition to performing protocol conversion by translating requests from WSP to other protocols and the responses back into WSP, the gateway also performs content conversion. This is analogous to HTML/HTTP proxies available on the Web today. For example, a user with a WAP-compliant telephone requests content using a specific URL. The telephone browser connects to the operator controlled WAP gateway and sends a GET request with that URL. The gateway performs a request for the content specified by the URL. The HTTP server at the contacted host processes the request and sends a reply. The gateway receives the content, encodes it, and returns it to the browser.

The security requirements in the WAP wireless application environment vary, based on the sensitivity of the information involved and the needs of the user. The possibility exists to create applications, which do not require any security, and other applications, which require extremely high levels of security (i.e. banking applications).

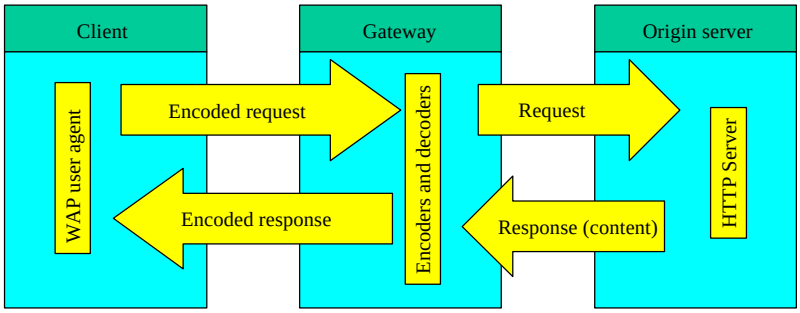


Figure 20: The WAP architecure

4.7.2 Technological environment

The key elements of the WAP specification include:

- Definition of the WAP programming model as seen in Figure 20, which is based heavily on the existing WWW model. Optimisations and extensions have been made in order to match the characteristics of the wireless environment. Wherever possible, existing standards have been adopted or have been used as the starting point for WAP technology.
- A mark-up language adhering to XML standards that is designed to enable powerful applications within the constraints of handheld devices, i.e. the Wireless Mark-up Language (WML) and WMLScripts. Unlike the flat structure of HTML documents, WML documents are divided into a set of well-defined units of user interactions. One unit of interaction is called a card, and services are created by letting the user navigate back and forth between cards from one or several WML documents. WML provides a smaller, telephony aware, set of mark-up tags that makes it more appropriate than HTML to implement within handheld devices. From the WAP gateway, all WML content is accessed over the Internet using standard HTTP1.1 requests, so traditional Web server, tools and techniques are used to serve this relatively new market.
- A specification for a microbrowser in the wireless terminal that controls the user interface and is analogous to a standard Web browser. This specification defines how WML and WMLScripts should be interpreted in the handset and presented to the user. The microbrowser specification has been designed for wireless handsets so that the resulting code will be compact and efficient, yet provide a flexible and powerful user interface.
- A lightweight protocol stack to minimise bandwidth requirements, guaranteeing that a variety of wireless networks can run WAP applications. The protocol stack is shown in Figure 21.
- A framework for Wireless Telephony Applications (WTA) allows access to telephony functionality such as call control, phone book access and messaging from within WMLScript applets. This allows operators to develop secure telephony applications integrated into WML/WMLScript services.

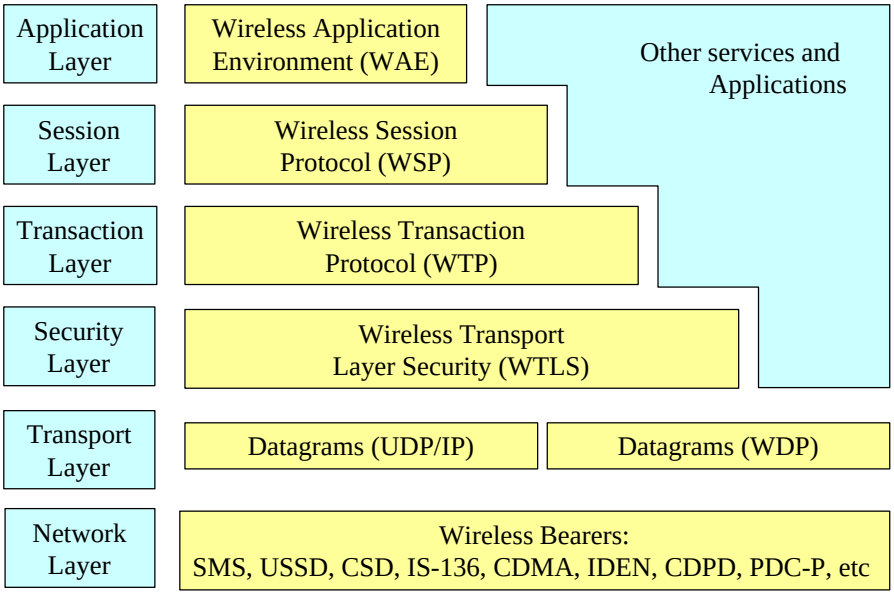


Figure 21: The WAP protocol stack

4.7.3 Security services

The WTLS (Wireless Transport Layer Security) protocol is a security protocol, designed for securing communications and transactions over wireless networks. It is based upon the industry-standard Transport Layer Security (TLS) protocol, formerly known as the Secure Socket Layer (SSL). WTLS is intended for use with the WAP transport protocols and has been optimised for use over narrow-band communication channels and incorporates new features such as *datagram support*, *optimised handshake* and *dynamic key refreshing*. It is used with the WAP transport protocols to provide transport layer end-to-end security. WTLS provides the following features:

- Data integrity – WTLS contains facilities to ensure that data sent between the terminal and an application server is unchanged and uncorrupted.
- Privacy – WTLS contains facilities to ensure that data transmitted between the terminal and an application server is private and cannot be understood by any intermediate parties that may have intercepted the data stream.
- Authentication – WTLS contains facilities to establish the authenticity of the terminal and application server.
- Denial-of-service protection – WTLS contains facilities for detecting and rejecting data that is replayed or not successfully verified. WTLS makes many typical denial-of-service attacks harder to accomplish and protects the upper protocol layers.

WTLS may also be used for secure communication between terminals.

WTLS provides an interface for managing secure connections. Applications are able to selectively enable or disable WTLS security services depending on their security requirements and the characteristics of the underlying network (e.g. confidentiality may be disabled on networks already providing this service at a lower layer).

WTLS is being implemented in all major micro-browsers and WAP servers.

4.7.4 Technical security features

4.7.4.1 WTLS Internal architecture

The WTLS protocol consists of a *record layer protocol* and a *handshaking protocol*.

4.7.4.1.1 The Record Layer Protocol

The WTLS Record Protocol is a layered protocol which accepts raw data form the upper layer client layer to be transmitted and applies the selected compression and encryption algorithms to the data. Moreover, the Record Protocol takes care of the data integrity and authentication. Received data is decrypted, verified and decompressed and then handed to the higher layer.

The record protocol is divided into four protocol clients. The protocol stack is shown in Figure 22. The different clients are described in the following sections.

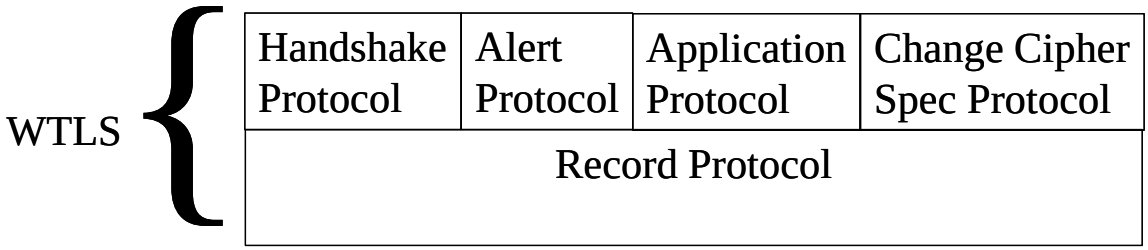


Figure 22: The WTLS protocol layer architecture

4.7.4.1.2 The Change Cipher Spec Protocol

The change cipher spec protocol exists to signal transitions in ciphering strategies. The protocol consists of a single message, which is encrypted and compressed under the current (not the pending) connection state. The change cipher spec is sent either by the client or server to notify the other party that subsequent records will be protected under the newly negotiated Cipher Spec and keys. The change Cipher Spec message is sent during the handshake after the security parameters have been agreed upon, but before the verifying finished message is sent.

4.7.4.1.3 The Alert Protocol

The Record Protocol also provides a content type of alert messages. There are three types of alert messages: warning, critical and fatal. Alert messages are sent using the current secure state, i.e. compressed and encrypted, or under null cipher spec, i.e. without compression or encryption.

If the alert message, labelled as fatal, is sent, then both parties terminate the secure connection. Other connections using the secure session may continue but the session identifier must be invalidated so that the failed connection is not used to establish new secure connections.

A critical alert message results in termination of the current secure connection. Other connections using the secure session may continue and the secure identifier may also be used for establishing new secure connections.

The connection is closed using the alert message. Either party may initiate the exchange of closing messages. If a closing message is received, then any data after this message is ignored. It is also required that the notified party verifies termination of the session by responding to the closing message.

Error handling in the WTLS layer is based on the alert message. When an error is detected the detecting party sends an alert message containing the occurred error. Further procedures depend on the level of the error that occurred.

4.7.4.1.4 The Handshake Protocol

The handshake protocol is composed of three sub-protocols which are used to allow peers to agree upon security parameters for the record layer, authenticate themselves, instantiate negotiated security parameters, and report error conditions to each other.

The handshake protocol is responsible for negotiating a secure session, which consists of the following items:

Item	Description
Session Identifier	An arbitrary byte sequence chosen by the server to identify an active or resumable secure session.
Protocol Version	WTLS protocol version number
Peer Certificate	Certificate of the peer. This element of the state may be null.
Compression Method	The algorithm used to compress data prior to encryption.
Cipher Spec	Specifies the bulk data encryption algorithm (such as null, RC5, DES, etc.) and a MAC algorithm (such as SHA-1). It also defines cryptographic attributes.
Master Secret	20-byte secret shared between the client and the server.
Sequence Number Mode	Which sequence numbering scheme (off, implicit, or explicit) is used in this secure connection.
Key Refresh	Defines how often some connection state values (encryption key, MAC secret, …) calculations are performed.
Is Resumable	A flag indicating whether the secure session can be used to initiate a new secure connection

Table 4: Handshake Protocol items

These items are then used to create security parameters for use by the record layer when protecting application data. Many secure connections can be instantiated using the same secure session through the resumption feature of the WTLS handshake protocol.

The cryptographic parameters of the secure session are produced by the WTLS Handshake Protocol, which operates on top of the WTLS record layer. When a WTLS client and server first start communicating, they agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and use public-key encryption techniques to generate a shared secret.

The WTLS Handshake Protocol involves the following steps:

- Exchange hello messages to agree on algorithms, exchange random values.
- Exchange the necessary cryptographic parameters to allow the client and server to agree on a pre-master secret.
- Exchange certificates and cryptographic information to allow the client and server to authenticate themselves.
- Generate a master secret from the pre-master secret and exchanged random values.
- Provide security parameters to the record layer.

- Allow the client and server to verify that their peer has calculated the same security parameters and that the handshake occurred without tampering by an attacker.

These goals are achieved by the handshake protocol, which can be summarised as follows: The client sends a client hello message to which the server must respond with a server hello message, or else a fatal error will occur and the secure connection will fail. The client hello and server hello messages are used to establish security enhancement capabilities between client and server. The client hello and server hello messages establish the following attributes: Protocol Version, Key Exchange Suite, Cipher Suite, Compression Method, Key Refresh, and Sequence Number Mode. Additionally two random values are generated and exchanged.

Following the hello messages, the server will send its certificate, if it is to be authenticated. Additionally a server key exchange message may be sent, if it is required. The server may request a certificate from the client, it that is appropriate to the key exchange suite selected. Now the server will send the server hello done message, indicating that the hello-message phase of the handshake is complete. The server will then wait for a client response. If the server has sent a certificate request message, the client must send the certificate message. The client key exchange message is now sent if the client certificate does not contain enough data for key exchange or if it is not sent at all. The content of that message will depend on the public key algorithm selected between the client hello and the server hello. If the client is to be authenticated using a certificate with a signing capability (e.g. RSA), a digitally signed certificate verify message is sent to explicitly verify the certificate.

At this point, a change cipher spec message is sent by the client and the client copies the pending Cipher Spec into the current write Cipher Spec. The client then sends the finished message under the new algorithms, keys, and secrets. When the server receives the change cipher spec message it also copies the pending Cipher Spec into the current read Cipher Spec. In response, the server will send its own change cipher spec message, set its current write Cipher Spec to the pending Cipher Spec, and send its own finished message under the new Cipher Spec. At this point the handshake is complete and the client and server may begin to exchange application data.

The procedure described above is the full handshake mechanism. The WTLS also defines an abbreviated handshake where only the hello and the finished messages are sent. In this case, both parties must have a shared secret that is used as a pre-master secret. Another variation is the optimised full handshake where the server can retrieve the client's certificate using the trusted third party, based on the information provided by the client in the client hello message. The information provided by the certificates, both parties are able to complete the shared secret using the key exchange method. The server has to send the server hello, certificate and finished messages to the client in order to complete the handshake on the server's behalf. The client responds with the client finished message.

4.7.4.2 Authentication

Authentication in the WTLS is carried out with certificates. Authentication can occur between the client and the server or the client only authenticates the server. The latter procedure can happen only if the server allows it to occur. The server can require the client to authenticate itself to the server. However, the WTLS specification defines that authentication is an optional procedure. Currently, X.509, X9.68 and WTLS certificates are supported. When the WTLS specification version 1.1 was released, the X9.68 certificate was not defined yet. The WTLS certificate is optimised for size.

The authentication procedure immediately follows after the client and server hello messages. When authentication is used, the server sends Server Certificate message to the client, The certified information given by the server is listed in Table 5.

Item	Description
Certificate version	Version of the certificate
Signature algorithm	Algorithm used to sign the certificate

Issuer	Defines the party who has signed the certificate, usually some CA
Valid not before	The beginning of validity period
Valid not after	The point of time after the certificate is no more valid
Subject	Owner of the key, associated with the public key being certified
Public key type	Type (algorithm) of the public key
Parameter specifier	Specifies parameter relevant for the public key
Public key	The public key being certified

Table 5: Certificate Information

Actually, the receiving end gets a list of certificates. The list is a chain of certificates where the first one is the server's own certificate. Each of the following certificates certifies the one preceding it. According to the WTLS specification, to optimise the traffic and the client processing it is possible for the server to send only one certificate; the server certificate certified with CA's public key which is distributed independently.

The sever may also send a Certificate Request to the client in order to authenticate it. The message will immediately follow the Server Certificate message and the Server Hey Exchange message (if sent). The Certificate Request message is optionally sent only if the Sever Certificate message is sent. In the message the server lists all the accepted certificate authorities. If no authorities are listed, the client may send any certificate.

At request, the client sends a Client Certificate message back to the server. The client end certificate follows the same structure as the server certificates. If the client does not have a suitable certificate the client must send an empty certificate message. Moreover, the client may send a fatal handshake failure alert message and close the connection. The Client Certificate message tends to contain multiple certificates. This is acceptable because the certificate list is processed by the server, which possesses more processing power than the client.

4.7.4.3 Key Exchange

In order to ensure a secure communication channel, encryption keys or initial values to calculate keys have to be exchanged in a secure manner. The certified exchange of public keys was described in the previous section. However, it is possible that the Server Certificate message or the Client Certificate message did not contain enough data to allow the client or server to exchange the pre-master secret (the pre-master secret is an initial value which is used to calculate the master secret). In this case a key exchange message is used to provide such data.

The key exchange mechanism of the WTLS also provides an anonymous way to exchange keys. In this procedure, the server sends a Server Key Exchange message which contains the public key of the server. The key exchange algorithm may be RSA, Diffie-Hellman, or the elliptic curve Diffie-Hellman. The message does not contain any certified information.

With both the RSA and the anonymous RSA the client encrypts the pre-master secret with the server's public key and sends it back to the server in the Client Key Exchange message. With Diffie-Hellman based algorithms the client and the server calculate the pre-master secret based on one's private key and the counterpart's public key. This message is omitted if some Diffie-Hellman based algorithm was used and the client certificate was requested so that the client was able to respond it.

If the client has listed the cryptographic key exchange methods that it supports, the server may choose whether it is going to use the clients suggestions or define another method. If the client has not proposed any method the server has to indicate them.

4.7.4.4 Privacy

Privacy in the WTLS is implemented by means of encrypting the communication channel. The used encryption methods and all the necessary values for calculating the shared secret are exchanged during the handshake.

In the first messages, the Client Hello and the Server Hello messages, random values are exchanged. In latter phases the client and the server exchange the pre-master secret. This value is transferred over a secure connection as described in the previous section. These values are used to calculate the master secret.

The used encryption algorithm is chosen in the Server Hello message. In this message the server informs the client that it has chosen a single cipher suite. The client provides the server with a list of cipher suites. The cipher suites comprise of a bulk encryption algorithm and a MAC algorithm. The first item in the list is the client's preference. If the server does not find an acceptable cipher suite, the handshake fails and the connection is closed.

Currently the most common bulk encryption algorithms are supported such as RC5 with 40, 56 and 128 bit keys, DES with 40 and 56 bit keys, 3DES and IDEA with 40, 56 and 128 bit keys. All algorithms are block cipher algorithms, no streams ciphers except NULLs are supported.

Encryption keys are conducted based on a key block. The key block is calculated form the initial values transferred during the handshake. The key block is dependent on a sequence number which makes the key block variable. The key block is recalculated in certain intervals based on the key fresh frequency. The key fresh frequency is negotiated in the Client Hello and the Server Hello messages.

4.7.4.5 Integrity

Data integrity is ensured using the message authentication codes (MAC). The used MAC algorithm is decided at the same time as the encryption algorithm. The client sends a list of supported MAC algorithms where the preferred algorithm is the first in the list. The server returns the selected algorithm in the Server Hello message.

The WTLS supports common MAC algorithms such as SHA and MD5. There are several different versions of both algorithms e.g. SHA exists with 0, 40 and 80 bit MAC sizes. The keyed MACs are calculated using SHA-1. The modified algorithms are based on SHA-1 but only part of the output is used. Same kind of versions exist for the MD5 algorithm.

4.7.5 Bootstrapping security

The handshake protocol is used to initiate a secure session. It allows peers (server and client) to agree on a protocol version, to select cryptographic algorithms, to optionally authenticate each other, and to generate a shared pre-master secret.

4.7.6 Trust model

The WTLS layer includes support for both client-side and server-side certificates. Using WTLS, the WAP client and WAP gateway can mutually authenticate each other. Moreover, WAP defines the WMLScript Crypto Library, which provides a basic mechanism for signing content that a client must send to a server. This library forms the foundation for implementing nonrepudiation transactions.

However, these mechanisms are quite limited. WAP does not provide any mechanisms for performing true end-to-end authentication between the client and the origin server. Because WTLS and SSL/TLS are incompatible, content must be decrypted and reencrypted as it passes through the WAP gateway. So, the client and server rely on the "chain of trust" built from the client to the WAP gateway and from the WAP gateway to the origin server

4.7.7 Strengths and weaknesses

Some weaknesses in the WTLS protocol are known and described in the WAP WTLS Specifications.

The use of uncertified public keys for the generation of a shared pre-master secret is provided. However, this implies that neither the client nor the server is authenticated, which makes the system vulnerable to active man-in-the-middle attacks.

Under certain circumstances, the Diffie-Hellman and elliptic curve Diffie-Hellman key agreement schemes are susceptible to a class of attacks known as “small-subgroup” attacks. Specifically in all WTLS cases there exists the threat that a small subgroup attack can lead to exposure of an entity’s private key. This threat can be prevented by checking that received public keys do not lie in a small subgroup of the group, or it can mitigated by selecting a group to have few, if any, small subgroups. Furthermore, in anonymous DH and ECDH cases there exists the threat that a small subgroup attack can lead communicating parties to share a session key which is known to the attacker. This treat can be prevented by using a predetermined group and checking that received public keys do not lie in a small subgroup of the group, or by not re-using ordinary Diffie-Hellman key pairs.

In e-commerce applications, wireless clients frequently need to strongly authenticate themselves to either a wireless gateway or an application server. In the WAP environment, client-to-gateway authentication is provided within WTLS in WAP 1.2. However, client-to-application authentication requires functions at a higher layer than WTLS.

4.7.8 Future trends

Future work items with respect to WAP security include:

- Improved support for end-to-end security
  - Various mechanisms are under consideration for extending the WTLS protocol endpoint beyond the WAP gateway.
  - Introduction of application level mechanisms for encryption and signing, which will be interoperable between WAP and the Internet world.
- Integrating Smart Cards for security functions
  - Wireless Identity Module specification will integrate Smart Cards into the security framework of WAP.
  - Uses Smart Card for storage of security parameters, as well as performing security functions.
- Providing a scalable framework for Client Identification
  - Public Key Infrastructure for provisioning and management of certificates.
  - Simpler mechanisms for clients that do not support certificates.

4.7.9 References

[1] “Wireless Application Protocol – Wireless Application Environment Overview Version 1.3”, can be found at http://www.wapforum.org/what/technical.htm .

[2] “Wireless Application Protocol – Architecture Specification”, can be found at http://www.wapforum.org/what/technical.htm .

[3] “Wireless Application Protocol – Wireless Transport Layer Security Specification”, can be found at http://www.wapforum.org/what/technical.htm .

[4] A WTLS presentation that can be found at http://www1.wapforum.org/member/developers/slides/wap-security/

4.8 I-mode

4.8.1 Description of the system

The intention of I-mode, as WAP (Section 4.7), is to offer the possibility for mobile devices to access Internet services. The development and evolution of I-mode is completely controlled by NTT DoCoMo (http://www.nttdocomo.com) and is not an open standard but a propriety system.

The services offered by I-mode are:

- ***I-mode menu sites*** – This is a list of sites registered at NTT DoCoMo's I-mode Center. This I-mode Center takes the advantage of a variety of online services offered by information providers such as mobile banking, weather information, games,
- ***Message Service*** – You can have information, such as news flashes, automatically sent to you by registering at a Message Service site.
- ***I-mode Compatible Sites*** – You can enter a website address using the keys on the mobile phone and access the Internet and view various websites. Remark that Internet sites that are not I-mode compatible may not be displayed properly.
- ***I-mode mail*** – You can send e-mail messages to other I-mode mobile phones and to Internet e-mail addresses

Connecting to sites, receiving messages and transmitting e-mail are done via the DoCoMo I-mode center (Figure 23).

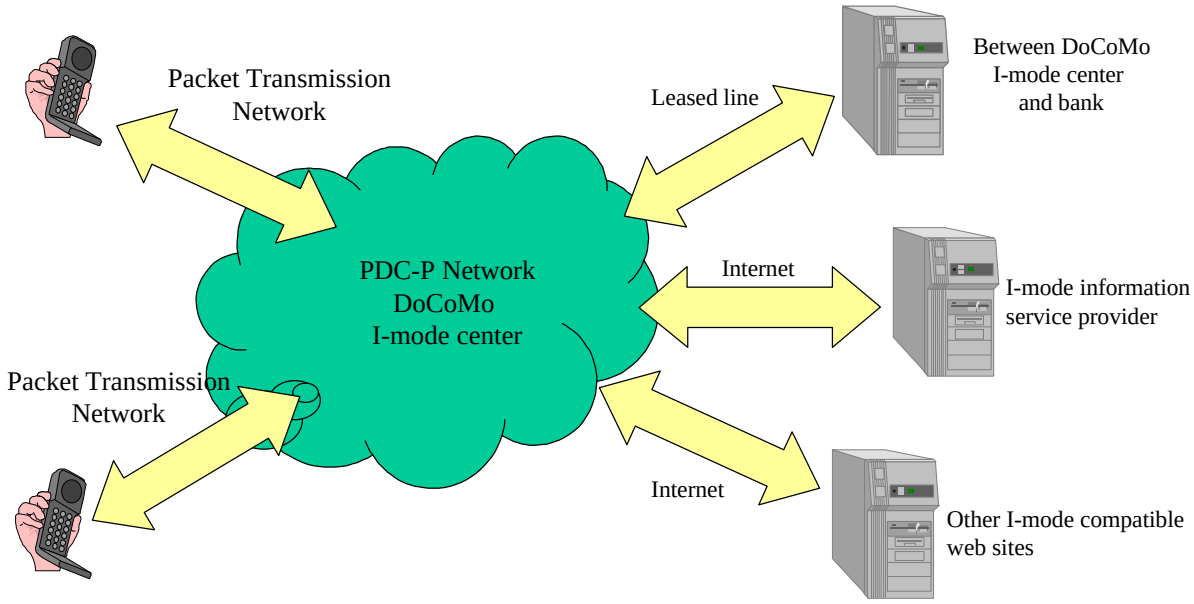


Figure 23: I-mode Structure

4.8.2 Technological Environment

NTT DoCoMo's I-mode service enables users to access a variety of information and exchange email on the Internet through simple operations of the browser installed in mobile terminals. They have developed and configured a PDC (Personal Digital Cellular) mobile packet communications system (PDC-P) that efficiently exchanges information in I-mode formats, and has also newly developed the I-mode server to connect content sites and manage users, as well as the protocol conversion equipment to link the I-mode server with the PDC-P system (Figure 24).

In order to improve communications efficiency and minimise packet communications fees, a data transmission protocol specific to I-mode communications has been developed and is being used under the PDC-P system. Because connections between the I-mode server and the Internet use generic TCP/IP technology, the PDC-P network includes a mobile message packet gateway (M-PGW) to handle conversions between these two protocol formats (Figure 24).

The I-mode server functions as a relay server between NTT DoCoMo's I-mode network (the PDC mobile packet communications system: PDC-P system) and the Internet under I-mode service. This server includes three service functions: relaying site information on the I-menu (I-mode menu) from the ISP (Information Service Provider); handling Internet mail or I-mode mail; and relaying access to the Internet.

The I-mode server actually consists of multiple server systems, each of which fulfils the assigned functions to realise various services. These include the business mobile access exchange (B-MAX) that manages the overall system; the contents mobile access exchange (C-MAX) that relays information from the ISP; the database mobile access exchange (D-MAX) that collects and analyses marketing information such as I-mode usage data; the M-PGW (Mobile Packet Gateway) that relays requests from each mobile unit; the interface mobile access exchange (I-MAX) that connects each server with the M-PGW; the mail mobile access exchange (M-MAX) that handles e-mail functions; the name mobile access exchange (N-MAX) that manages e-mail accounts; the user mobile access exchange (U-MAX) that manages subscriber information; and the Web mobile access exchange (W-MAX) that provides menu management for the I-menu. These server systems are linked to provide I-mode services as well as user authentication and charging management when subscribers access fee-based contents.

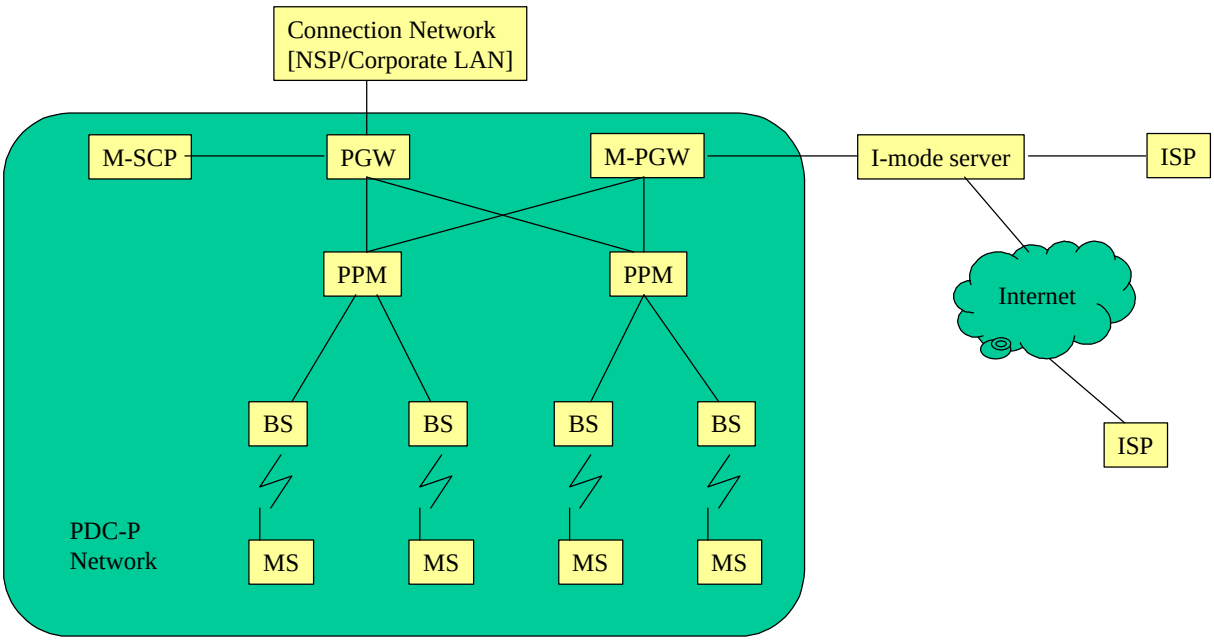


Figure 24: I-mode Network Architecture

The architecture is comparable with the WAP architecture (Section 4.7). The I-mode model adopts a model that is even more closely related to the WWW model than WAP. All I-mode content is specified in a format very similar to the standard Internet format. C-HTML (compact HTML) is used instead of normal HTML. C-HTML is developed by the World Wide Web Consortium (W3C) in 1998 and is a simplified version of HTML. However, it displays perfectly on any Internet browser.

In contradiction to WAP, where a WSP request is being translated into an HTTP request in the WAP gateway, there is no need for heavy protocol translation in the I-mode environment as HTTP/HTTPS and HTML request pass through the network without any substitution (Figure 25).

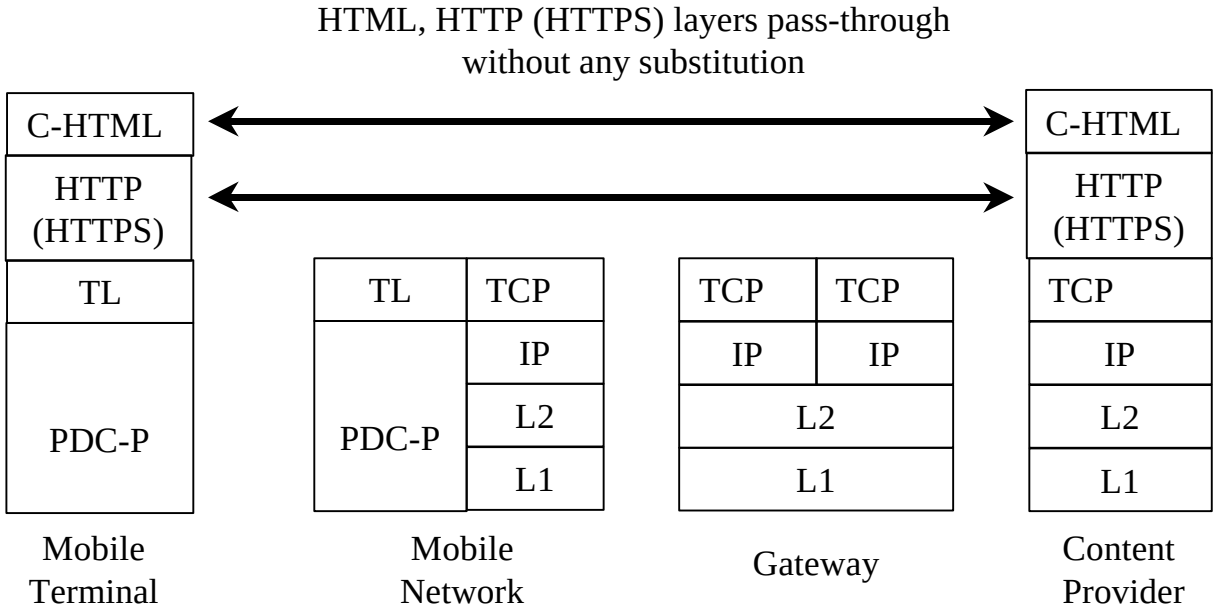


Figure 25: Protocols and protocol conversion in I-mode network

4.8.3 Security services

I-mode uses the SSL (Secure Socket Layer) protocol for securing communications and transactions over the network. The security services offered by this SSL layer include *data integrity*, *privacy*, *authentication* and *denial-of-service* protection, comparable as for WAP.

4.8.4 Technical security features

As the WTLS protocol layer in WAP described in section 4.7.4.1, SSL is composed of two layers. At the lowest level, layered on top of some reliable transport protocol (e.g.TCP), is the SSL Record Protocol. The SSL Record Protocol is used for encapsulation of various higher level protocols, i.e. the Handshake Protocol, the Alert Protocol, the Change CipherSpec Protocol and the Application Protocol (Figure 26).

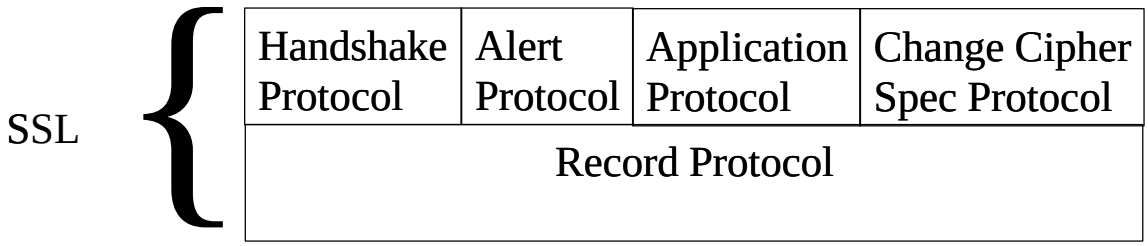


Figure 26: SSL protocol layer architecture

4.8.4.1 The Change Cipher Spec Protocol

The change cipher spec protocol exists to signal transitions in ciphering strategies. The protocol consists of a single message, which is encrypted and compressed under the current (not the pending) CipherSpec.

The change cipher spec message is sent by both the client and server to notify the receiving party that subsequent records will be protected under the just-negotiated CipherSpec and keys.

4.8.4.2 The Alert Protocol

One of the content types supported by the SSL Record layer is the alert type. Alert messages convey the severity of the message and a description of the alert. Two alert types exist:

- closure alerts: The client and server must share knowledge that the connection is ending in order to avoid a truncation attack. Either party may initiate the exchange of closing messages.
- error alerts: When an error is detected, the detecting party sends a message to the other party. Upon transmission or receipt of a fatal alert message, both parties immediately close the connection. Servers and clients are required to forget any session-identifiers, keys, and secrets associated with a failed connection.

4.8.4.3 The Handshake protocol

The cryptographic parameters of the session state are produced by the SSL Handshake Protocol, which operates on top of the SSL Record Layer. When a SSL client and server first start communicating, they agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and use public key encryption techniques to generate shared secrets.

These processes are performed in the handshake protocol, which can be summarised as follows: the client sends a client hello message to which the server must respond with a server hello message. The client hello and server hello are used to establish security enhancement capabilities between client and server. The client hello and server hello establish the following attributes: Protocol version, Session ID, Cipher Suite, and Compression Method. Additionally, two random values are generated (one by the client and one by the server) and exchanged.

Since the SSL Protocol is similar to the WTLS protocol, we refer to section 4.7.4.1 for more detailed information concerning the security principles of these protocols.

4.8.5 Bootstrapping security

The handshake protocol is used to initiate a secure session between client (Mobile Terminal) and server (Information Service Provider). When a SSL client and server first start communicating, they agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and use public key encryption techniques to generate shared secrets.

4.8.6 Trust model

In contrast with WAP, a true end-to-end trust relationship can be established between a mobile terminal and a content provider for the application layer data (HTTPS). This is possible since the mobile terminal is capable of verifying the certificate of the content provider and the content provider is able to verify the certificate of the mobile terminal.

4.8.7 Strengths and weaknesses

As for WAP, client to application authentication requires security functionality at a higher layer than SSL.

Since I-mode uses SSL and WAP uses WTLS, derived from SSL, we assume that the same weaknesses exist for I-mode as for WAP. These are outlined in section 4.7.7.

4.8.8 Future trends

A future trend of I-mode is to deliver higher bandwidths and use more powerful terminals (This is also the case for WAP).

The content language will move more towards XML and has the tendency to be more compatible with WAP (or WML) as well.

4.8.9 References

[1] "The unofficial independent imode FAQ", document can be found at http://www.eurotechnology.com/imode/faq-gen.html

[2] "Mobile Computing and I-mode", document can be found at http://www.nttdocomo.co.jp/corporate/rd/tech_e/mobi01_e.html

[3] Document to be found at http://www.nttdocomo.com/i/service/tokutyou.html

[4] "I-mode: The first successful smart phone service in the world", can be found at http://www.indiainfoline.com/bisc/art13012001.html

[5] "Compact HTML for Small Information Appliances", at http://www.w3.org/TR/1998/NOTE-compactHTML-19980209

[6] "The SSL Protocol Version 3.0" Internet draft draft-freier-ssl-version3-02.txt

4.9 E-speak

4.9.1 Description of the system

The E-speak vision is similar to that of Jini; that services can be accessed spontaneously, easily and intuitively (on a pay per usage basis) using a wide array of devices and platforms, from PDAs to supercomputers. E-speak core software is written in Java and therefore can run on any hardware/operating system that supports a JVM.

4.9.2 Technological Environment

The entire E-speak infrastructure is based on the following concepts:

- Everything is a resource.
- All resource access is mediated; e-speak sees all resource requests.
- E-speak deals only with resource representations (i.e. metadata), not resource specifics.
- All names are private. There are no global names.

The e-speak core software consists of two elements: universal e-services APIs and a run-time e-services engine. E-speak run-time software is installed on each computing device or information appliance that connects to the virtual service environment. The run-time software then provides basic infrastructure capabilities like messaging, mediation, security, naming, and monitoring for the e-speak e-services hosted on, or accessed by, these devices. Through the e-speak services interface (APIs), developers can design and create any e-service using a set of simple e-services programming interfaces. Services don't have to be written in Java but can be integrated using a library.

4.9.2.1 Resources

The resource is a service within E-speak and is described to E-speak by its metadata. There is a strict separation of resource state and resource metadata. The owner of a resource registers it with the core by providing the metadata. The core records this information in a repository and assigns the metadata a repository handle which is unique to this repository. The only way in which a client can refer to this resource is with a name which is bound to this repository handle. All resource access is through the core which uses the resource metadata to mediate and control each access.

Resource metadata is composed of a resource specification and a resource description. The resource description consists of information that provides the means of discovery for clients; it includes searchable attributes such as the name of the service and manufacturer. The resource specification includes an inbox that can be connected to the Resource Handler responsible for managing the resource, a specification of the security restrictions and a variety of control fields.

Every access to a resource through e-speak involves two different sets of manipulations: the e-speak platform uses its resource descriptions for dynamic discovery of the most appropriate resource, fine and coarse-grained access control, transparent access to remote resources, and sending events to management tools. The resource-specific handler directly accesses the resource, e.g. reading the disk blocks for a file.

4.9.2.2 Naming

Naming in e-speak is local. Each name presented to the core is interpreted in the local name space. The motivation for this is that when names reside in many places the name space can become incoherent when a name changes. Names can be bound to a resource, set of resources or a lookup request.

4.9.2.3 Contracts

A contract defines the interface for a set of services. It provides the stub used by the client to invoke methods on the service. The contract includes the secondary resources required by the services (name bindings that must be transferred), the permissions that are needed, and so on. A resource contract also contains a set of vocabularies that can be used to describe and discover resources of this type.

4.9.2.4 Vocabulary

A vocabulary is used to describe a service so that the potential clients can discover it. A vocabulary is very like an XML schema except that it is also a resource like any other. It is used to advertise rather than to invoke a service. An advertising service takes ads places by service providers and performs searches on behalf of its customers. A successful lookup on an advertising service results in a machine to contact in order to access the resource.

4.9.2.5 Communication

E-speak uses a mailbox concept to describe the interactions between clients and the core. Each client has an outbox connected to the core and one or more inboxes. A message has two parts: the envelope contains information of interest to the core and the payload carries the application-specific information. The core never looks at the payload which could be, for example, an XML file. When a client wants to use a resource it puts a message in its outbox which the core forwards to the relevant inbox.

When two machines connect, they go through a connection protocol. Once this is complete, each machine knows the resources that are available on the other machine and sends a form of the metadata describing these resources to the other machine. If a client wants to use a remote resource, its message goes to a proxy for the remote resource handle. The proxy bundles up the request and sends it to its counterpart on the other machine.

4.9.2.6 Events

Events provide a publish-distribute-subscribe mechanism for communication built on top of e-speak messaging. A publisher is an entity that generates an event notification message. The recipient of an event notification is called a listener. A distributor is an extension of a listener which receives events and forwards them to other listeners. A subscriber is an entity that registers interest in a particular event with a distributor and designates the listener to which events should be sent. The subscriber and the listener are typically the same physical entity. Similarly, it is fairly typical for a publisher to act as a distributor of its own events.

An E-speak event is just a message, which results from a subscription and consists of a set of attributes of a vocabulary. The recipient of an event does not need to know anything about the event's content *a*

priori; it can query the vocabulary to determine the event's attributes and their types and then extract the values of the attributes it is interested in. A subscriber can also specify a filter in this vocabulary when subscribing for events.

Normally, the subscriber wants to take some special action when the event is received. In particular, when a client receives an event, the callback registered for this event is invoked. A client may, for example, update the description of a service to which a client has access when it receives an event describing a change to the resource metadata.

4.9.2.7 Lookup

To use a service, the client connects to the core and gets a reference to a "finder". It uses the finder to locate a service which meets its requirements. The information given to the finder consists of a search recipe which contains the names of the requested service attributes and a reference to the vocabulary by which the attributes can be understood. Finally the service is invoked using the service stub.

Because the use of a service is only made possible by knowing its name, a client must issue a bind call to the e-speak core with a search recipe as a parameter. The e-speak system looks up the name in local lookup service and if necessary a global advertising service to locate the appropriate service and create a binding for the client in the client's name space.

4.9.3 Security Services

E-speak supports coarse-grain as well as fine-grain security services. First a user may share his/her services with specific users by creating his/her own vocabulary and making it visible only to those users. Since vocabularies partition the search space of descriptions, those who are not aware of the vocabulary may not find the services.

If requested by a client, e-speak virtualises names that identify services. With name virtualisation neither service providers nor clients need to reveal their identity to interact with each other. The engine keeps mapping information that relates virtual names to actual services. All requests to services through e-speak engines are mediated. The mediation is enforced using attribute certificates based on Simple Public Key Infrastructure (see below). Different access rights to a service are granted depending on the attributes authenticated. Service access in e-speak is based on asynchronous messaging. Layered above this, e-speak libraries provide user friendly interaction models such as the network object model. The mediated yet uniform access is the design principle that allows the e-speak engine to accommodate any type of resources and service.

4.9.3.1 Authorisation via SPKI Certificates

The SPKI was designed by the Internet Engineering Task Force (IETF) Workgroup to meet the needs for a public key certificate format, associated signature and other formats, and key acquisition protocols that could be used in a wide range of Internet applications from encrypted electronic mail and WWW documents to payment protocols.

The main purpose of a SPKI certificate is to authorise some action, give a permission, grant a capability, etc. to or for a key-holder (the one that controls a public key's private key).

A basic SPKI certificate defines a straight authorisation mapping between an authorisation and a key. If someone wants access to a key-holder's name, for logging purposes or even for punishment after wrong-doing, then one can map from key to location information (name, address, phone, ...) to get it.

This mapping has a security advantage over the more common attribute certificate mapping (one that maps an authorisation to a name). In the mapping described above, only the authorisation - key mapping needs to be secure at the level required for the access control mechanism, i.e., at the level of the application that uses it. The key-name mapping (and the issuer of any certificates involved) needs to be secure enough to satisfy lawyers or private investigators: a subversion of this mapping does not permit the attacker to defeat the access control. The care with which these certificates (or database entries) are created is less critical than the care with which the authorisation certificate is issued. It is

also possible that the mapping to name need not be on-line or protected as certificates, since human investigators only in unusual circumstances would use it.

4.9.3.2 Delegation

One of the powers of an authorisation certificate is the ability to delegate authorisations from one person to another without bothering the owner of the resource(s) involved. So one might issue a simple permission (e.g., to read some file) or issue the permission to delegate that permission further.

One issue arises when we consider delegation: the desire to limit the depth of delegation and the question of separating delegators from those who can exercise the delegated permission.

As regarding the depth of delegation, SPKI certificates allow for boolean control, that is, the permission to delegate is either TRUE or FALSE. In this way, SPKI enables an entity to decide if it wants or not to give permission to delegate a certificate it issues with a specific authorisation. For instance, imagine a commercial entity empowered by an authority to use a certain cryptographic module and able to delegate it, it can issue several certificates to allow several suppliers to use the module but not delegate its use.

There is no control on the width of the delegation tree, so a problem of proliferation of permissions may arise. A Delegator always has the ability to exercise any permission it is able to delegate.

4.9.3.3 Validity Conditions

In order to limit the use of a certificate, one must use optional validity conditions. The traditional ones are validity dates: not-before and not after. SPKI also uses three kinds of online tests to further refine validity conditions: Certificate Revocation List (CRL), revalidation and one-time. Although the issuer of the certificate may not be the same as the issuer of the online validity confirmation, in SPKI, the certificate issuer must specify the issuer of validity confirmations.

4.9.3.4 Confidentiality and Integrity Protection

E-speak provides confidentiality and integrity protection for messages exchanged between e-speak Cores and between e-speak Cores and Clients via the Session Layer Security protocol (see below).

4.9.4 Technical Security Features

4.9.4.1 CRLs

In SPKI there is only one type of CRL (Certificate Revocation List), which is timed CRL. If a certificate can be referenced in a CRL, then the CRL process is subject to three conditions.

- The certificate must list the key (or its hash) that will sign the CRL and may give one or more locations where that CRL might be fetched.
- The CRL must carry validity dates.
- CRL validity date ranges must not intersect. That is, one may not issue a new CRL to take effect before the expiration of the CRL currently deployed.

Under these rules, no certificate that might use a CRL can be processed without a valid CRL and no CRL can be issued to show up as a surprise at the verifier. This yields a deterministic validity computation, independent of clock skew, although clock inaccuracies in the verifier may produce a result not desired by the issuer. The CRL in this case is a completion of the certificate, rather than a message to the world announcing a change of mind.

4.9.4.2 Timed Revalidations

CRLs are negative statements. The positive version of a CRL is what we call a revalidation. Typically a revalidation would list only one certificate (the one of interest), although it might list a set of certificates (to save digital signature effort).

As with the CRL, SPKI demands that this process be deterministic and therefore that the revalidation follow the same rules listed above.

4.9.4.3 One-time Revalidation

Validity intervals of length zero are not possible. Since transmission takes time, by the time a CRL was received by the verifier, it would be out of date and unusable. That assumes perfect clock synchronisation. If clock skew is taken into consideration, validity intervals need to be that much larger to be meaningful.

For those who want to set the validity interval to zero, SPKI defines a one-time revalidation.

This form of revalidation has no lifetime beyond the current authorisation computation. One applies for this on-line, one-time revalidation by submitting a request containing a nonce. Those nonce gets returned in the signed revalidation instrument, in order to prevent replay attacks. This protocol takes the place of a validity date range and represents a validity interval of zero, starting and ending at the time the authorisation computation completes.

4.9.4.4 Using SPKI Certificates in E-Speak for Authorisation

The most common use of certificates in the Internet is to link an entity's name to its public key. A drawback of this approach is that, typically, having used the certificate to verify the name, a service needs to consult an authorisation database to determine the type of access to be granted.

E-speak use of certificates is more general: they are signed, linking a public key to a Name or a Tag. A Tag typically states an access right, thus to make an access control decision a service does the following:

- - Examines the tag in the certificate to see if it grants access
- - Checks if the entity making the request knows the corresponding private key
- - Verifies if the certificate has been issued by an entity it trusts

An entity doesn't have to trust all Issuers equally, or need not trust any given Issuer at all. Anyone can issue a certificate, but permission to access depends whether the Resource Handler trusts the Issuer for the service in question.

A typical scenario is when two companies are accessing a portal to use services provided by the portal. Since the data held by the services can be sensitive, both companies want to be sure that their employees are accessing the correct portal and services, moreover they might prefer to be responsible for managing their own lists of employees and control who can access the portal's services. The portal configures the services to trust certain issuers, thus making its services accessible only to the company that presents a certificate from the respective Issuer. The companies then issue certificates to each of their employees thus controlling who among them have access to the services.

In this way the company have the power to revoke access to its employees, and the portal to revoke access to the entire company.

4.9.4.5 The Session Layer Security Protocol

All messages exchanged between e-speak Cores and between e-speak Cores and Clients use the Session Layer Security protocol. This provides secure message passing between entities as well as unprotected message exchanges. Applications can choose whether to use secure message passing or not.

Session Layer Security protocol is designed to support e-speak mediation. E-speak mediation requires e-speak to modify certain parts of the message so that the message can be routed between endpoints and means there are not a TCP connection between the endpoints. These requirements means that existing protocols such as SSL (Secure Sockets Layer) or TLS (Transport Layer Security) are not suitable for end to end security in e-speak.

Session Layer Security protocol allows multiple secure sessions to be multiplexed over a single TCP connection. This means that two e-speak Cores can be connected via a single TCP connection with many Clients and have many different secure sessions to different e-speak Resources.

Session Layer Security protocol also supports tunneling. During firewall traversal we might want the firewall to control the client access rights to the internal LAN for every packet. However, we might not want the firewall to see all the traffic in clear (therefore, losing the end-to-end security property). With Session Layer Security protocol we can nest a secure session inside another one, possibly with different end points, allowing us to achieve both goals simultaneously.

Session Layer Security protocol is designed to support SPKI for access control. It performs the negotiation of access rights that need to be proven represented by multiple SPKI certificates. Session Layer Security supports the following encoding types for messages.

- CLEAR_DATA: The message is not encrypted or protected against modification.
- PROTECTED_DATA: The message is not encrypted but it is protected against modification.
- SECURE_DATA: The message is encrypted and protected against modification.

Session Layer Security has been designed to be independent of transport. However, for interoperability between e-speak Cores and interoperability between e-speak Cores and Clients, direct implementation over TCP is assumed. Other implementations are possible, including passing SLS messages over HTTP, or through shared memory.

4.9.5 Strengths and Weaknesses

4.9.5.1 Strengths

- E-speak has a comparatively evolved security system.
- Service Description: Powerful representation of services in any content language.
- Sophisticated brokering and mediation.
- Billing Metrics: E-speak provides some metrics for measuring the flow of data between E-speak services to aid with billing.
- Language-independent: Services can be written in other languages than Java, like C or Perl, provided one has the corresponding library.
- Interoperability: E-speak allows to incorporate device interconnection technologies such as HP's Chai or Sun Microsystem's Jini.

4.9.5.2 Weaknesses

- E-speak is not really a device interconnection technology, since it must rely on another technology to provide a device interconnection.
- E-speak assumes IP as the only transport protocol.

4.10 TeSSA

4.10.1 Description

TeSSA addresses some of the challenges that are posed in executing mobile code on a set of distributed hosts that are connected through an untrusted network. Particularly it proposes a secure method to allocate local resources on a hosting device to mobile executables based on authorisation. This authorisation is not based on the identity of the executable. The authorisation is based on the executable having explicit permission given by a Trusted Third Party (TTP) that is trusted by the host. This makes it possible for the executable to remain anonymous to the host. However, the trusted entity

should authenticate the owner of the executable before granting permission to resources on the targeted host.

The TeSSA architecture uses SPKI (Simple Public Key Infrastructure) certificates to carry permissions for a given resource. The purpose of an SPKI certificate is to grant authorisation to a specific action. They are designed to contain the minimum amount of information about the owner to grant authorisation by the verifier.

SPKI certificates in TeSSA are used to capture these permissions and to bind them to a temporary identity created for the owner of the executable. This temporary identity is in the form of a temporary public and a private key. The temporary keys are generated dynamically by the TTP whenever there is a request for a resource on a host. The temporary public key is contained in a SPKI certificate that grants the permissions to the owner of the corresponding temporary private key. The host will verify the issuer of the SPKI certificate and will check whether the application is bound to the SPKI certificate by verifying it using the certified public key. In a network with more than one potential host, the SPKI certificate contains explicit information on which devices could act as a host for a given executable.

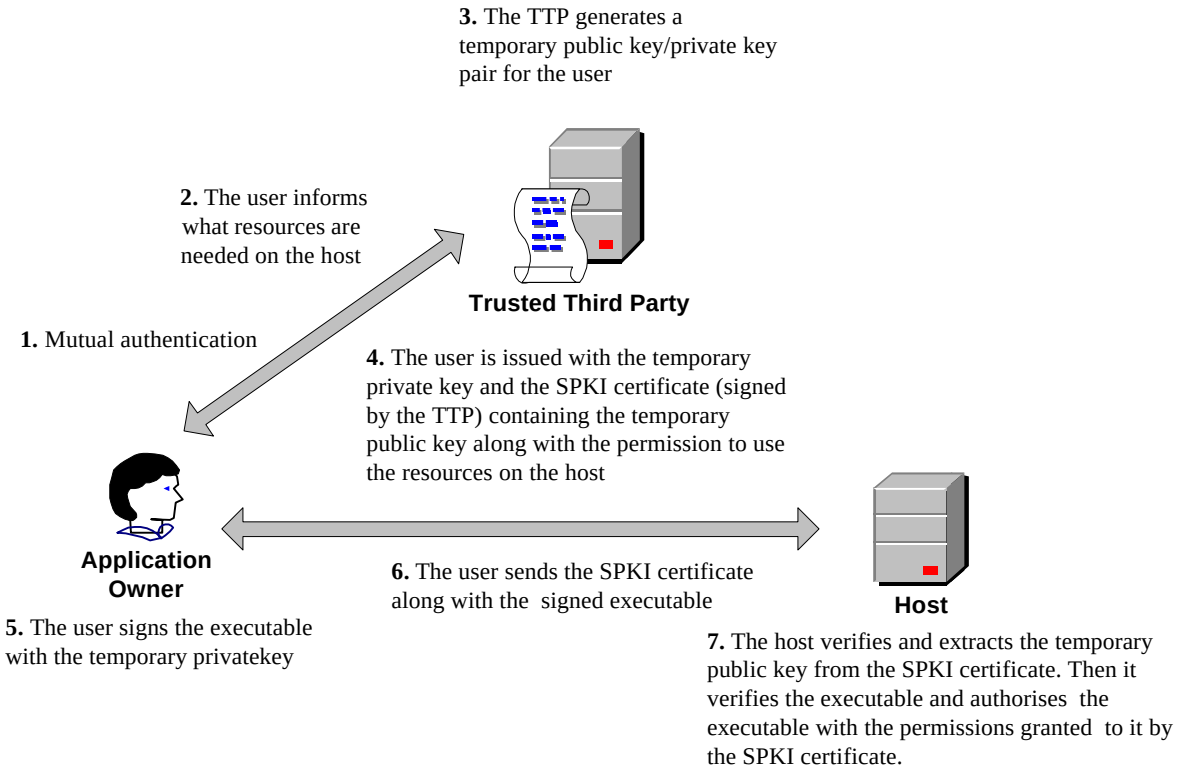


Figure 27. The TeSSA architecture

In the above example (see Figure 27), if delegation is allowed, it is possible for one executable to delegate its permissions in the SPKI certificate to other executables that are trusted by it. Since the generation of certificates by TTPs and possible by other applications occur in a dynamic fashion, TeSSA uses Elliptic Curve based DSA (ECDSA) to achieve a reasonable efficiency. The TeSSA security architecture is used in JDK 1.2.

4.10.2 Issues

A security concern withi TeSSA is that temporary private keys could be vulnerable in transit from the TTP to the user if they are not properly protected. If the attack is successful, then the private key could be used to launch attacks against the hosts.

Maintaining anonymity towards a host could be exploited by a malicious user to spread virus code to hosts within the TeSSA architecture. In such a case it would be useful to have a mechanism for the TTP to be informed of the temporary identity of the malicious user such that the original identity of the user could be established. The TTP could then revoke any outstanding SPKI certificates by informing the hosts accordingly.

From a resource point of view, the TTP server could be inundated with requests from users for the temporary identities and SPKI certificates for different resources. In addition the TTP servers will get numerous requests from hosts during their attempt to verify SPKI certificates and temporary identities of users. This will require a high degree of availability and reliability from the TTP server.

4.10.3 References

[1] Ylitalo, Jukka. :Secure Platforms for Mobile Agents, 6 January 2000
http://www.hut.fi/~jylitalo/seminar99/

[2] Telecommunications Software Security Architecture (TeSSA)
http://www.tml.hut.fi/Research/TeSSA/

4.11 MeT

4.11.1 Description of the system

The description is based on MeT Core Specification [1].

4.11.1.1 Scope of MeT

The Mobile electronic Transactions (MeT) Initiative was formed to foster coherent growth of the mobile e-business market. It aims to ensure that applications using secure transactions are developed with a consistent user experience across multiple phones, access technologies and usage scenarios.

The starting point of MeT is that the mobile phone is rapidly evolving into much more than a wireless telephone. It is transforming into a Personal Trusted Device (PTD), with the ability to handle a wide variety of new services and applications, such as banking, payments, ticketing and secure access-based operations. MeT defines how these services can be built around the PTD and realized in the remote, local and personal environments.

MeT embraces and extends existing industry standards and technologies. MeT draws upon WAP specifications for WTLS, WIM and WPKI. MeT also emraces Bluetooth wireless technology.

MeT defines three kinds of interfaces, see 3.11.1.2.2:

- the service registration interface
- the service execution interface
- user experience aspects of handling secure transactions with the PTD

MeT defines and operates in three environments, see 3.11.2:

- the remote environment, or the Mobile Internent world, by enabling access to WAP shops, WAP banks, etc.

- the physical, or local environment by facilitating services in a shop, ID services at work, etc. (possibly over Bluetooth)
- the personal, or home environment by working with Bluetooth technology to access and process Internet content with a PC, for example.

4.11.1.2 System reference model

4.11.1.2.1 Key Elements

Figure 28 shows the system reference model for MeT. This is a generic model that can be used as a reference for all usage scenarios. In many scenarios the issuer, acquirer and/or content server can be reducedto two or even one entity. In all scenarios, interfaces to the PTD, namely the service registration, service execution and security elements interfaces, remain the same. Moreover, this same reference model is applicable to remote, local and personal environments.

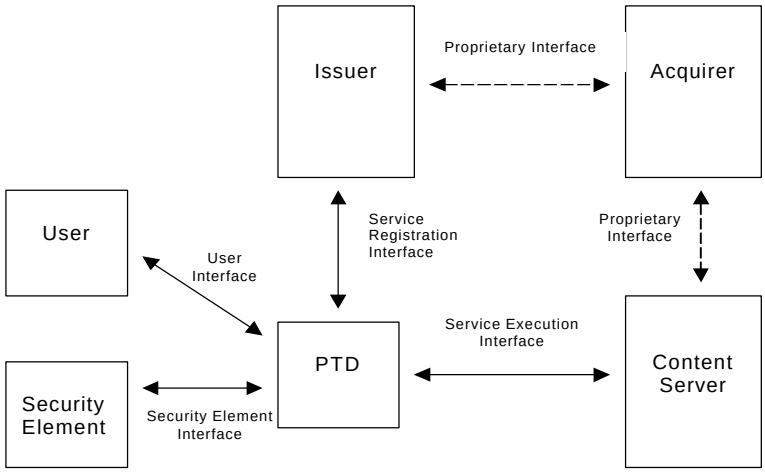


Figure 28. System reference model diagram

The key elements presented in the diagram are as follows:

1. **PTD**: The user's mobile phone is the Personal Trusted Device (PTD). The PTD is used to establish the user's ID (authentication) and authorize transactions (with a digital signature). The PTD includes a Security Element (SE) which contains the user's private-public key pairs and root certificates (used to verify other certificates). The SE may be implemented as any of the following:

- a combined SIM/WIM (SWIM) card;
- a separate WIM smart card;
- another removable device carrying WIM functionality;
- a hardware SE built into the phone: or
- a software-based SE in the phone.

The PTD also contains the user's service certificates or certificate URLs. (Service certificates and certificate URLs are provided to the user by service providers, e.g., banks, or Certification Authorities (CA).)

The MeT security requirements for the PTD are given in [3]. In this document also issues related to mobile equipment security, such as operating environment and random number generation, are discussed.

2. **Content server:** The content provider supplies content through the content server. In bothe the remote and local environments, the content sent from the content server is presented to the user on the PTD. In the personal environment, the content may be displayed on another device.

   The application running on a content server may request the user to perform authorization (sign a transaction). The PTD will be instructed to perform this command via the service execution interface.

3. **Issuer:** The issuer supplies the service certificate for a given account. In genereal, an account, or user account, is for the service offered by the issuer. This account might be associated with a monetary value or personal user data. The issued service certificate is a means for the issuer to identify the user.

4.11.1.2.2 MeT interfaces

MeT defines the following interfaces in the system reference model (Figure 28):

- service execution interface from the PTD to the content server
- service registration interface from the PTD to the issuer
- user interface between the PTD and the user
- SE interface between the PTD and a removable SE

Service execution interface : This interface is used for performing secure transactions with a content server. The following security functions are performed via this interface:

- secure session establishment
- user authentication
- authorization by user

Service registration interface: This interface is used for loading service certificates onto PTD and for personalizing the PTD for the user and for the service.

User interface: This interface represents the interactions with the user that are necessary to perform MeT transactions. It involves

- presenting information to the user on the PTD
- prompting for an input, such as signature PIN, and
- accepting the input and forwarding it to the appropriate calling routines.

A consistent user interface (at a sufficiently high level of abstraction) is a major contributor to consistency of user experience, which is a major MeT objective. The MeT Consistent User Experience (CUE) document [2] concentrates on the usability factors of PTD, and identifies requirements for consistent and secure interaction between the user and the PTD.

SE interface: This interface exists between the security element and two physical and logical entities in the PTD: the transport layer environment and the application layer environment. The interface defines the interaction pertaining to user verification, cryptographic processing, etc. that are defined in the WAP WIM specifications. Strict adherence to the WIM interface specification is required for removable SEs. For non-removable SEs, the SE interface may be proprietary tot he PTD manufacturer. Examples of non-removable security elements are an embedded hardware SE and a software SE.

4.11.2 Technological environment

The technological environments in which MeT-enabled transactions will occur can be classified into three categories: remote, local and personal.

4.11.2.1 Remote environment

In the remote environment the connection between the content server and the PTD is established via a PLMN, such as the GSM cellular network. WAP access to the internet will be gained through a WAP gateway. The WAP gateway performs the proxy server functions of protocol conversion from WAP protocols to Internet protocols. Alternatively, the content server may itself host the WAP gateway.

Transport layer security implications of the two configurations are different. In the first case, WTLS is used from the PTD to the WAP gateway, with TLS/SSL being used from the WAP gateway to the content server. In the second case, end-to-end transport layer security is provided from the PTD to the content server.

4.11.2.2 Local environment

In the local environment MeT transactions are initiated over a short-range wireless technology such as Bluetooth. A typical application would be retail shopping using an account-based payment made from a PTD.

The typical network topology for MeT local environment is as follows. A Bluetooth access node is located where the service is to be delivered (in the above example, this would be a retail store). Content providers thereby have the ability to offer different content (offers) based on the location of the user. PTD is connected to the Bluetooth access node using WAP over Bluetooth. The Bluetooth is connected to the content server over a LAN.

In the local environment MeT does not rely on Bluetooth authentication and encryption for authenticating the content server and providing a secure session. WTLS is used to provide these security services.

4.11.2.3 Personal environment

In the personal environment the PTD enables secure transactions from another communications device (a PC, for example). The PTD participates in authentication and authorization by making available service certificates and performing any signature operations required by its SE. The user may use another communications device without having to personalize it.

An example of this environment is a situation wherein a PTD user on a PC is browsing a site that requires user authentication via Internet TLS protocol. The service certificate for the site has already been provided to the user, and the private key associated with the certificate is securely stored in the SE of the PTD. The PC requests the PTD to perform the cryptographic operations requiring the private key, that is , authenticating and authorization. Except for signature operations, the PC handles all of the protocol related features of TLS. The PC and PTD can communicate using wireless technology, such as Bluetooth or IrDA or via a physical connection using USB or serial ports.

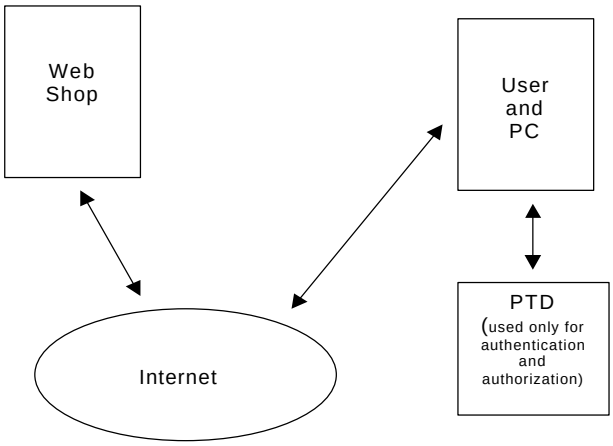


Figure 29. MeT in an example of the personal environment

The personal environment enables external devices to access PTD functions, such as key storage, making use of the security inherent to the SE. This enables issued certificates to be used across all access methods. Other devices will be able to execute MeT functions in the PTD using the personal service execution interface.

4.11.3 Security services

4.11.3.1 Secure session establishment

A secure MeT session has the following components:

- confidentiality
- data integrity
- server authentication

Ideally, the secure session is between the PTD and the content server; current technology, however, does not always allow for this.

In the remote and local environments secure sessions are provided by the WAP security layer, WTLS. Currently, according to the WAP June 2000 Conformance Release, WTLS operates only from the PTD to the WAP gateway. This provides a secure session to the content server when the content server is located at the WAP gateway. When the content server is separate from the WAP gateway, secure sessions may be established using a two-phase approach that involves using WTLS between the PTD and WAP gateway, and SSL or TLS between the WAP gateway and the content server. In this case the WAP gateway must be secured and trusted.

4.11.3.2 User authentication

User authentication is the means by which a service provider determines the identity of a user. Not all MeT services require user authentication. For example, an open Web shop, where browsing access, or window shopping, is open for all visitors, does not require authentication (purchasing, though, does require user authentication). On the other hand, a content server handling sensitive information, such as online banking server, would require user authentication before providing the content to the user.

MeT uses WTLS class 3 to perform user authentication. This allows WAP gateway to establish the identity of the user by means of the user service certificate related to the private key stored in the PTD. PTD authentication is established by the server sending the PTD a challenge, which the PTD signs and returns.

User verification is enacted by the entry of an access PIN known only to the user. This PIN allows access to the key pair used for authentication.

As discussed under secure session establishment, the authentication is to the WAP gateway and not necessarily to the content server. The upcoming WAP end-to-end transport layer security will allow authentication to a gateway located at the content server.

4.11.3.3 Authorization by user

Authorization is the means by which a service provider ensures that the user has viewed and accepted a transaction contract. Authorization is achieved by digitally signing the transaction contract using a private key associated with a user service certificate. The signed transaction contract may be stored by the service provider as a record of the transaction authorization.

MeT uses the WAP WMLScript *signTex*t function for authorization. The *signText* is a WMLScript function and operates as follows. One field is a *stringToSign*. The content server builds a contract into the s*tringToSign*. The script is sent to the PTD where *stringToSign* (contract) is displayed on the phone screen. If the user accepts the contract, he/she is requested to enter a signature PIN. This is the process of user verification for authorization. If the entered PIN is correct, the *stringToSign* is signed with the PTD's signature private key and returned to the content server. This signed contract is the content server's proof of the user's authorization of the transaction.

4.11.4 Technical security features

MeT embraces and uses the security features defined in WAP (see Section 4.7), that is, WTLS, WIM and WMLScript signText. A summary of the parts of these security technologies as adopted in use by MeT is given below.

At its current stage of development, MeT does not use the link layer security features provided in bBluetooth.

4.11.4.1 WTLS

The main features of WTLS are the following:

- Based on TLS/SSL
- Supports datagram transport protocols (e.g., UDP/IP)
- Provides bandwidth and memory optimization
- Provides online key refresh for connection of long duration
- Supports cryptographic algorithms
  - encryption: RC5, DES, 3DES, IDEA
  - integrity: HMAC with SHA-1, MD5
  - key exchange: RSA, Diffie-Hellman, ECC
- Provides authentication options: anonymous, server authentication, server and client authentication

4.11.4.2 WIM and its interface

WIM, as defined in WAP, serves as the functional and interface definition of the removable SE in MeT. Its physical implementation options include:

- smart card (SIM or other)
- other tamper-resistant hardware module

The applicable standards are the following:

- ISO 7816-4 (storage, PIN verification etc.)
- ISO 7816-8 (cryptographic operations)
- PKCS#15 (information format)

The key functionality of WIM includes the following:

- WTLS client authentication
- secure session handling
- digital signatures

4.11.4.3 WMLScript signText

Functionality

- Authorizing application level transactions
- Providing security – a service may store the signature as a proof of authorization

4.11.5 Bootstrapping security

4.11.5.1 PTD initialization

PTD initialization means equipping a SE with the initial private-public key pairs and root CA certificates necessary to execute MeT secure transactions. Some SEs may be delivered to users pre-initialized -- in the case of WIMs, with initial private-public key pairs and root certificates already in place. A PTD, furthermore, may have access to multiple SEs.

Keys may be provided externally through a removable SE, such as SWIM card, a separate WIM card, or other removable media with WIM functionality and interface. Alternatively, the keys may be generated internally in an embedded SE. The CUE aspects of the key generation process is specified in [2].

Client certificates, stored in the security element, have labels which are displayed to the user to identify the certificate and, when the user is prompted to enter a PIN, to unlock the certificate.

Root CA certificates may be provided as a part of the initialization process. The process for downloading root CA certificates in the remote and local environments is specified by WPKI. These certificates contain the public keys of root CAs that issue certificates for the user's transaction partners. There maybe more than one root certificate.

4.11.5.2 Registration and personalization

Registration is the means by which a service provider associates the user's identity with a service account. It is done by associating a user service certificate with the service. The process of registration personalizes the PTD for the user.

In MeT the PTD registers for a service by having service certificates assigned to its key pairs (separate key pairs being used for authentication and signature). In keeping with the WPKI recommendation, the PTD will be capable of handling both full certificates and URLs stored in the PTD (the URLs point to detailed certificates stored on the Web).

A fundamental concept of MeT is that multiple service certificates from different service providers may be assigned to a common key pair.

The process of certificate download is defined in WPKI. It involves making online request by the PTD to a PKI portal, which forwards the request to a CA. The CA generates the certificate and sends either the full certificate or the URL directly to the PTD. The CA enters the issued certificate in its public key database.

Other methods of certificate download may be developed in the future.

4.11.6 Trust model

The main purpose of MeT is to provide secure technological environment for mobile electronic transactions. It relies on the existing security technologies in WAP and TLS, but also defines requirements for the PTD. Successful transactions rely on the user's ability to handle the multiple security functions on PTD in a correct and coherent way. Other points of trust are the physical security of the WAP gateways and the servers of the service provider.

4.11.7 Strengths and weaknesses

At the current stage of MeT specification, only partial solutions to the authentication problem are presented. Open problems include, but are not limited to the following:

- In the initialization process, much concern is devoted to the handling of user certificates, but not a word is mentioned about how the user gets the server certificates. In the CUE document [2], in the section 4.11.1 Creating secure session, the following short recommendation is found: "In addition to connection privacy and data integrity, it is recommended that the server authentication is used when creating a secure session." Secure session is established using WTLS and TLS or SSL, but only one side of the required initialization is currently been defined in MeT.
    - The current WAP solutions do not offer end-to-end authentication from the PTD to the service. It is expected that as the need for additional application layer authentication functions is identified, it will be addressed through methods involving WAP, Bluetooth and other baseline standards of MeT. How this will be developed, remains an open problem.

The strength of MeT at its current stage of development, is in the extensive work done in the are of usability.

4.11.8 Future trends

Not known.

4.11.9 References

[1] MeT Core Specification, Version 1.0, 21-February-2001, available at www.mobiletransaction.org

[2] MeT Consistent User Experience, Version 1.0, 21-February-2001, available at www.mobiletransaction.org

[3] MeT PTD Security Requirements, version 1.0, 21 February 2001, available at www.mobiletransaction.org

4.12 ARIB MT-TA interface

4.12.1 System description

The Japanese organisation Association of Radio Industries and Businesses (ARIB) conducts studies, research and development, and standardisation activities on radio systems. ARIB is a member of the Third Generation Partnership Project (3GPP). ARIB is specifying a new interface, called "MT-TA" [1]. This is a regional standard activity outside 3GPP but it focuses on 3GPP terminals. This interface is an external interface of the Mobile Equipment (ME), i.e., between the Mobile Termination (MT) and the Terminal Adapter (TA). The interface description is only logical and a description of the

physical interface is excluded. The MT-TA interface is used to interconnect different logical functions within the ME as well as external Terminal Equipment (TE).

ARIB has "split" the mobile terminal into six different core logical function blocks:

- **TAF - Terminal Adaptation Function** is a logical block that provides communication services via the Mobile Terminal. It is classified into three types: circuit switched communication services, packet switched communication service, or both circuit and packed switched communication services. The TAF provides the Terminal Equipment (TE) with communication services.
- **MTF - Mobile Termination Function** is a logical block operating as a termination point of the mobile station's radio bearer., i.e., this function is taking care of all signalling and connection set-up toward the base stations.
- **TMF - Terminal Management Function** is a logical function performing address management and connection management between the different logical function blocks.
- **UIMF - User Identity Module Function** is a logical function enabling other functions to communicate with the User Identity Module (UIM).
- **DCF - Device Control Function** is a logical function block enabling other functions to access input/output devices on the mobile terminal such as the display, the keyboard, the sounder, the microphone, the phone book etc.
- **OMF - Operation & Maintenance Function** is a logical function block enabling operation and maintenance of various functions in the terminal.

In Figure 30 below gives an architectural overview of the MT-TA interface. In this figure the TAF exist logically inside the TA and the MTF exist logically inside the MT in the ME. The MT-TA reference point is taken from the old GSM specifications.

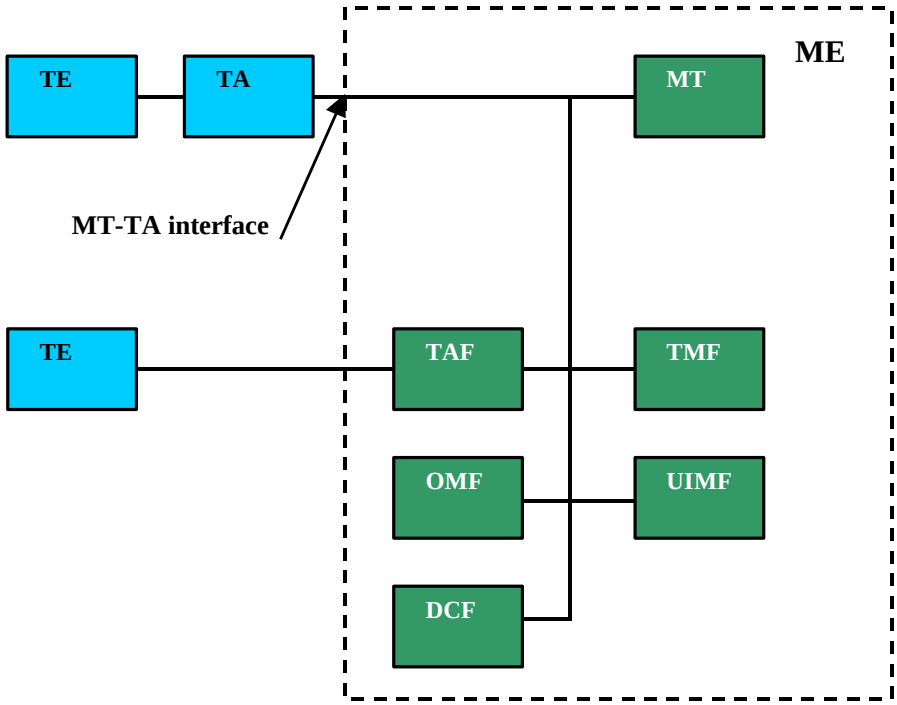


Figure 30, The MT-TA interface

4.12.2 Technological environment

The interface is described using OSI type commutation layers. However, even if the specification is referring to layer 1 as the "physical" layer, it is a higher layer protocol that is used as layer 1

connections for the interface. The MT-TA interface defines layer 2 and layer 3 protocol for control signalling and user data transmission. No specific physical interfaces (layer 1) are required for control plane. For the user plane, USB [2], IrDA [3] and Bluetooth [4] are supported. However, a list of requirements must be fulfilled for the physical layer.

For the control plane physical layer the following requirements must be fulfilled:

1. It shall be possible to connect multiple functions to the TMF.
2. It shall be possible for any function to communicate with other functions.
3. It shall be possible for the OM to monitor all communication between the other functions.
4. If a user is allowed to remove a function, it shall be possible for the system to detect if the function is connected or not.
5. The system must ensure reliable transfer (error detection and retransmission) of layer 2 packets as well as collision detection.

The physical interface for the control plane is denoted by L1s-C, the layer 2 protocol by TLP-C (Terminal Link Protocol). On Layer 3, different application protocols are defined. Examples of control plane application protocols are: Address Assignment & Connection Management Protocol (ACMP), Packet Switched Connection Management Protocol (PCP) and Device Control Protocol (DCP). The connection configuration for the control plane is shown in Figure 31.

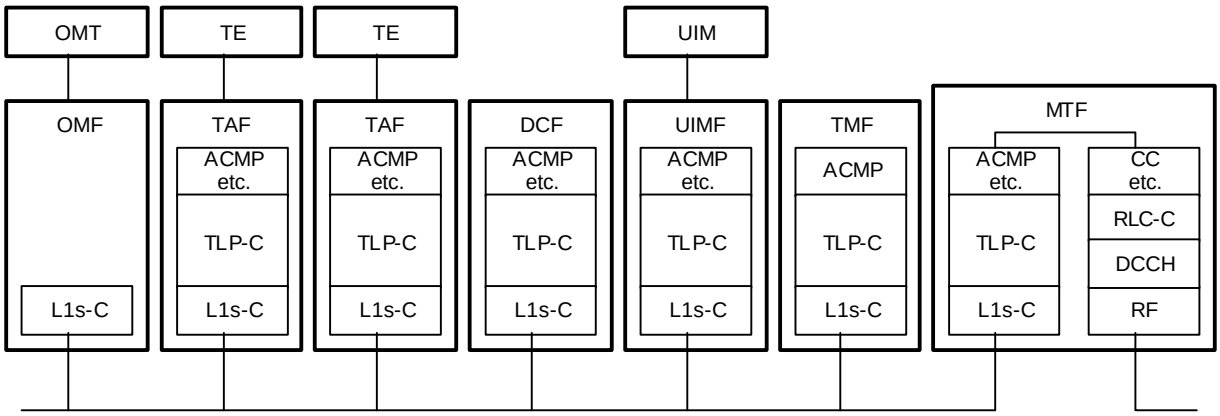


Figure 31, Connection Configuration Diagram (C-Plane)

For the user plane physical layer the following requirements must be fulfilled:

1. Connections between the MTF and multiple TAFs, between TAFs should be possible.
2. The transfer procedure should allow for synchronous transfer in case of voice-communication services and data communication services.
3. Given that multiple TAFs can be connected to the MTF, there will be occasions when these connections are used simultaneously (multiple calls). Logical CHs for multiple U-Plane connection should therefore be supported between the MTF and TAFs.
4. There are certain requirements on the data transfer rate between the MT and TA as well as maximum latency between the MT and TA.

The physical interface for the user plane is denoted by L1s-U, the layer 2 protocol by TLP-U (Terminal Link Protocol). Both synchronous and asynchronous layer 2 protocols are defined. No MT-TA specific layer 3 protocols are defined for the user plane. The connection configuration for the user plane is shown in Figure 32.

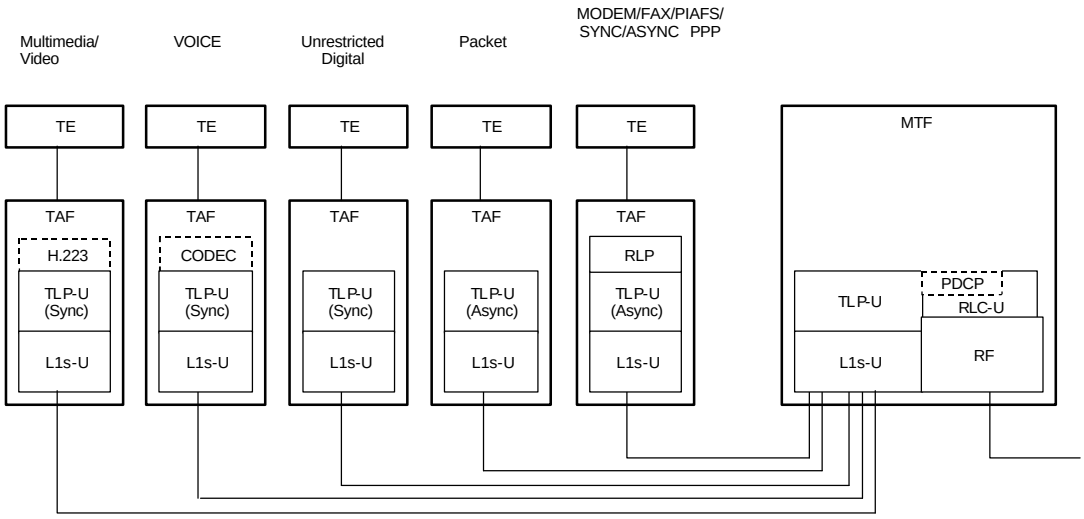


Figure 32, Connection Configuration Diagram (U-plane)

4.12.3 Security services

The current specification does not contain any secure service. It is stated that the security is provided by the link level protocol. For example if Bluetooth is used, the Bluetooth authentication and encryption provide the security. However, there is ongoing work on security for the MT-TA interface within ARIB. But, currently there is no protection on either control or user plane specified for the MT-TA interface.

4.12.4 Threat analysis

The MT-TA architecture is an open architecture where core terminal functions are allowed to pass an open interface. Even if most functions are internal, the TAF might typically be located physically outside the rest of the functions. Depending on the type of physical interface between the TA and MT, the vulnerability will vary. But, in general this new interface will open up for attacks against several core terminal functions. Both eavesdropping, denial of service and theft of service might be possible. Examples of treats towards the ME core functionality are:

- PINs travels over the bus towards the UIMF.
- The UIM might be open via the UIMF to functions that has not been authenticated. This means that any APDU command might be sent towards the UIM and a response can be received.
- Sensitive information like IMSI might be retrieved over the UIMF by a hostile function module.
- Network authenticate commands might be executed by a hostile function over the UIMF.
- Illegal circuit switched connection might be set-up.
- Illegal packet switched connections might be set-up.
- The ME can be put out of service by denial of service attacks from an illegal function module.
- Circuit switched and packet switched calls between the MTF and a legal TAF might be eavesdropped by an illegal TAF.

4.12.5 Future trends

Currently 3GPP has rejected any proposal regarding work with a MT-TA type of interface within T2. Hence, currently this is outside the scope of the 3GPP standardisation. However, there is ongoing discussion within 3GPP T2 on different terminal split use cases. In order to support these new use cases, new interfaces will be needed.

4.12.6 References

[1] "MT-TA Interface Description", Association of Radio Industries and Businesses (ARIB) , http://www.arib.or.jp/IMT-2000/V130Jan01/T12/0_T12coverV130.html.

[2] Universal Serial Bus Specification Revision 2.0, April 27, 2000, http://www.usb.org/

[3] Infrared Data Association (IrDA), http://www.irda.org/

[4] Bluetooth Special Interest Group (SIG), http://www.bluetooth.com/

5 Security Requirements

In this section we present the security requirements that we intend to make our security architecture satisfy. These requirements will be derived by first developing a role model for configurable, distributed terminals, and then generating the requirements that each role has. This latter process will be done on the basis of common sense and experience and also by examining other collections of security requirements, such as [21.133] and [3GS3].

Security requirements will be derived by considering the configurability and the dynamic aspects of the terminals together and not separately.

A sub-section will give some security features that could be used to meet the requirements and considerations regarding those security features.

These requirements will in some cases relate to application layer threats such as malicious applications. It should be noted that the resulting security architecture will **not** address these threats by any means relating to the characteristics of the content of such malicious applications, such as virus checking programs. Such means cannot be defined by a security architecture and are left out of the scope of the work of SHAMAN. However, their use will clearly be an important element in protecting the user from these threats.

5.1 Role model

The roles below are considered to be significant. Note that this role model will be refined in future deliverables when a reference architecture for configurable and distributed terminals is derived.

We distinguish between PAN component owners and users. For devices such as mobile phones and PDA the owner and user will usually be the same person. However, for components such as printers and other “service devices”, the user and owner may be different.

- PAN component user
- PAN component owner
- Application service provider
- Communications service provider
- Authorisation authority
- PAN component manufacturer
- PAN manager
- Intruder

5.1.1 PAN component user

The PAN component user is the human entity that is making use of the PAN component at a particular time. The user may be the owner of the component but this is not assumed for the role.

The user’s primary concerns with regard to the component are related to the effective use of the component. Concerns that the user may act irresponsibly in operation of the component are covered under the owner role. This is to avoid duplication and is not meant to suggest that all non-owning users are not concerned with responsible use of the component.

The PAN component may or may not have a subscription module that gives chargeable access to the services of a communications service provider.

We will assume that user to component authentication mechanisms will be located entirely within the component. We further assume that there is either only one class of user or that there is no impact on the class of user accessing a component beyond the component. Therefore, we can deduce that user to component authentication mechanisms need not (and will not) form part of the security architecture.

5.1.2 PAN component owner

The PAN component user is the human entity that owns the PAN component. The owner's primary concerns with regard to the component relate to charges and liabilities arising from the user of the component and the long term effectiveness and working order of the component.

For PAN components with a subscription module, in some situations, the owner may be the communications service provider.

5.1.3 Application service provider (ASP)

The ASP provides services to the user at the application layer. The ASP does not act as an enabler for bearer or transport layer services – this is the role of the communications service provider.

For the purposes of this study, we will only consider services where the ASP is providing some *content* to the user. This content may be non-executable text (e.g. where the ASP is a web server) where there may be no security requirements or where the requirements may mainly relate to confidentiality and integrity. On the other hand, if the content is executable, authentication of the source (i.e. the ASP) by the user is very important – as stated at the start of this section, origin authentication for applications will the only means considered by SHAMAN WP2 of preventing malicious applications. Content scanning mechanisms will not form part of the security architecture.

The ASP may not be the author of the content but we will assume that for security purposes, that they are. That is, we will assume that the ASP is the termination point of security services relating to services provided by the ASP. Confidentiality, integrity and authentication of source will extend to the ASP and not beyond. Importantly, this means that where origin authentication is provided using public key methods, it will be the ASP that digitally signs the content, and therefore requires a certificate, and not the author of the content.

5.1.4 Communications service provider (CSP)

The CSP is the party that enables bearer and transport layer services for the user. A user may require more than one CSP, for instance, they may require a cellular operator for wireless bearer services and then an "internet service provider" (ISP) for access to the Internet.

For our requirements analysis, we will assume that users do NOT require the services of a CSP for inter-PAN communications.

5.1.5 Authorisation authority

The authorisation authority is the body that authorises, where such authorisation is required, ASPs to give their applications privileged access to PAN component functionality. If digital signatures are being used for authentication and authorisation of applications, the AA will in fact be a CA.

5.1.6 PAN component manufacturer

The PAN component manufacturer manufactured the PAN component. They appear in this role model as they may want to download upgrades to the software of the PAN component. This role could just be seen as a type of ASP, but software (perhaps "native code") may well require more complex authorisation than that given to an "ordinary" ASP, so the component manufacturer is given a distinct role.

5.1.7 PAN component administrator

The PAN component administrator is not required from a service point of view, but exists to provide some sort of policing of AAs. The AAs are a powerful group in that they determine which parties are authorised to download content to clients and which are not. To guarantee the overall effectiveness in the event that an AA is using this power irresponsibly, there is a need for the overall effectiveness of the security architecture to remove the AA's right to authorise. As this task may be too difficult for the user or owner, this task falls to the "PAN component administrator", the guardian of the user's and owner's security requirements. The role of PAN component administrator might be taken by a large company for instance, with regard to all PAN components (phones, laptops, PDAs) that it provides its employees with for work use. Private users may wish their cellular operator, whom they trust, to be their administrator. Alternatively, a private user or PAN component owner may wish to be their own administrator, though security architecture designers should recognise that only a tiny minority of users and owners will have the competence or desire for this role.

5.1.8 PAN Manager

The role of a PAN manager is necessary from an internal PAN and a service point of view. It provides access control for new PAN members and components.. The role of the PAN manager is assumed either by a user or owner of aPAN component. Since this task require responsibilities beyond the usual user's responsibilities as a PAN user, the role of the PAN manager is introduced separately. It is also possible that PM takes the responsibility of the AA in authorising services to the PAN.

5.1.9 Intruder

As defined in [D02]

5.2 Security requirements

5.2.1 User

As stated in the role model, the user's requirements relate to the quality of service they receive from the PAN component(s) under their control, and also that the user's privacy and choice in their use of the PAN component is preserved. It should be assumed that in all the user's requirements, they relate to PAN components under the user's control only.

We will assume that the PAN component user will ensure that malicious applications are not downloaded to its PAN component by preventing tampering of applications in transit and that applications are only downloaded from authenticated sources. As it may not be enough to authenticate the source, but to determine that the source is one authorised to deliver executables, a further requirement may be that the user can determine, with authenticity and integrity, that the source is appropriately authorised.

As security mechanisms applied to applications "on entry" to a PAN component may not always suffice, the PAN component will have the ability to keep a log of all actions taken by applications downloaded to the PAN component and all actions of PAN component's with which the PAN component interacts.

The user will be in charge of whether or not the PAN component they control will accept another or not as a PAN manager. As this activity occurs "in the field" it is given to the PAN component user, and not to the owner.

We assume that the security of user data on the component is out of the scope of the architecture.

The following security requirements result.

1. It shall be possible to provide confidentiality of user data and signalling data in transit to PAN component.

2. It shall be possible to provide integrity of user data and signalling data both in transit to PAN component and when stored in a PAN component.
3. It shall be possible to provide authentication of the source of applications downloaded by the user.
4. It shall be possible for the user to determine whether an authenticated source is an authorised source of applications or not.
5. It shall be possible for the user to be in control of which applications are downloaded to the user's PAN component, with regard to download from both remote servers and other components in the user's PAN.
6. It shall be possible for the PAN components to keep a log of all actions taken by applications downloaded to the PAN component and all actions of PAN component's with which the PAN component interacts.
7. It shall be possible for the user to exercise complete control over the transmission of user input information from the user's PAN component to both remote servers and other PAN components
8. It shall be possible for the PAN component owner to nominate a party to act as PAN manager with regard to their PAN component.
9. It shall be possible for the PAN component owner to revoke the rights of the PAN manager with regard to their PAN component.

5.2.2 PAN component owner

The owner wishes to preserve the functionality of their PAN component and that the PAN component does not accrue any charges that the owner has not in some way authorised. The owner will achieve these goals by only allowing authorised parties to use the functionality of the PAN component and to accrue charges. The PAN component owner will also wish to control which parties can use any network access that the PAN component has by virtue of a subscription module and network access interface. The PAN component owner shall also be in charge of the security policies, with regard to AAs, observed by their component.

10. It shall be possible for the PAN component owner to only grant access to use the PAN component to authorised parties.
11. It shall be possible for the PAN component owner to only grant access to accrue charges to authorised parties.
12. It shall be possible for the PAN component owner to only grant network access and use of any subscription module that the PAN component has to those parties authorised by the owner.
13. It shall be possible for the PAN component owner to define the access policy and security policy with regard to their PAN component.
14. It shall be possible for the PAN component owner to define the access policy for applications, and negotiate the security policy with the ASP with regard to their PAN component.

5.2.3 Application service provider

The ASP wants to ensure that his applications cannot be tampered in transit to a client and may also want to deliver the content with confidentiality. He also wants to ensure that no one can send tampered with or otherwise malicious applications whilst posing as the ASP, and so get the ASP blamed. For billing purposes, the ASP may want to be able to authenticate the client.

The ASP may need to know the terminal capabilities of the client and this information may be sensitive and require protection.

15. It shall be possible for the ASP to authenticate the client receiving a downloaded executable.

16. It shall be possible for the client to authenticate the source (by definition, the ASP) of a downloaded executable.
17. It shall be possible for the downloaded executables to be sent from the ASP to the client with confidentiality.
18. It shall be possible for the downloaded executables to be sent from the ASP to the client with integrity.
19. It shall be possible for the ASP to obtain the clients terminal capabilities with integrity.
20. It shall be possible for the ASP to obtain the client terminal capabilities with confidentiality

5.2.4 PAN Component Manufacturer

The manufacturer appears in this model as they may want to download to download upgrades of phone software. In this sense, the manufacturer is an ASP. However, as the manufacturer may be downloading software operating at a lower layer than "applications", the terminal **may** need security functionality other than that provided at the application layer, to secure this download. However, this seems not to be the case, as the requirements in the ASP case (authentication, etc.) are not with respect to applications executing at the application layer but could apply to executables at any layer. However, as downloads from the manufacturer may have a more far reaching effect on the PAN component, it is important that the PAN component manufacturer is recognised as such, so he, and only he, can download such privileged executables. Therefore the single requirement below, on top of those for an ASP, results:

(As this download will be by the manufacturer to their components only, there is not strictly a need for a standardised method and a proprietary method may be sufficient. However, proprietary security mechanisms are often insecure because they have not been scrutinised over a wide enough audience. Therefore it seem preferable to have a standardised solution.)

21. It shall be possible for the client to authenticate the PAN component manufacturer and recognise him and no other party as the PAN component manufacturer.

5.2.5 Communication Service Provider

The CSP is clearly anxious that the user receives good service. However, we will assume that all these requirements are covered by those of the user. In additions to these, the CSP, where the CSP is not itself acting as an ASP or an authorisation authority, has requirements relating to the preservation of the operation of its infrastructure when there is the possibility of viruses and other malicious software. This "mal-ware" could be used to disrupt the operation of the CSP's infrastructure, for example, by some sort of distributed denial of service attack.

The question is, will the CSP rely on user (or more correctly, the user's PAN component's security mechanisms) to rule out or minimise the possibility of mal-ware, or will the CSP take action themselves? The CSP's efforts, if the latter option is chosen, will only be effective if the CSP can ensure that it is the only source of applications for the PAN component user. It will be assumed that this is not the case. It should also be pointed out that, assuming a healthy market for downloaded applications, that it will be a heavy burden on the CSP to somehow ensure that all applications downloaded by its clients are not mal-ware. This cannot be done in the presence of confidentiality between the ASP and the client. Further, the CSP, if it "checks" applications, may be liable to users if the application does turn out to be mal-ware. This may be a legal risk the CSP is unwilling to take. For all these reasons, we will assume that there are no effective ways which can be implemented at a reasonable cost for the CSP to ensure that mal-ware is not distributed to its clients. We will therefore assume that the CSP will rely on the functionality of the client.

5.2.6 Authorisation authority

The authorisation authority (AA) will need to authenticate those parties that request authorisation. She will want the authorisation levels granted to an ASP to be passed to the client with integrity, and for

the client to obey the authorisation level conferred. The AA will want the facility to "revoke" the authorisation given to an ASP if the ASP turns out to be false, malicious or incompetent.

The AA may want a number of authorisation levels that it can confer on an ASP, to ensure that ASPs are only given the authorisation they need or desire and no more.

22. It shall be possible for the AA to authenticate ASPs requesting authorisation.
23. It shall be possible for the AA to confer variable authorisation levels on ASPs.
24. It shall be possible for the authorisation level conferred on an ASP to be passed with integrity to the client.
25. It shall be possible for the authorisation level conferred to an ASP to be passed with confidentiality to the client
26. It shall be possible for the client to obey the authorisation level received.
27. It shall be possible for the AA to withdraw the authorisation level on ASPs and ensure that terminals are informed of this withdrawal of authorisation.

PAN component administrator

The PAN component administrator's role relates to policing AAs on behalf of the user.

It shall be possible for the PCA to disable authorisation capability of AAs with respect to all clients under the PCA's supervision.

5.2.7 PAN Manager

The PAN manager (PM) will need to authenticate PAN component users that require authorisation for participation in the PAN. She will want the authorisation levels granted to the PAN component user with integrity. She will also want the PAN component user to obey the authorisation level conferred. The PM will want the facility to revoke the authorisation in case of false, malicious or incompetent behaviour of the PAN component user.

28. It shall be possible for the PM to authenticate PAN component users requesting authorisation.
29. It shall be possible for the PM to confer variable authorisation levels on PAN component users.
30. It shall be possible for the authorisation level conferred on a PAN component user to be passed with integrity to other PAN component users.
31. It shall be possible for the authorisation level conferred to a PAN component user to be passed with confidentiality to other PAN component users.
32. It shall be possible for the other PAN users to obey the authorization level granted to the new PAN component user.
33. It shall be possible for the PM to withdraw the authorisation level on PAN component users and ensure that other PAN components are informed of this withdrawal of authorisation.

The PAN manager's role also relates to policing AAs on behalf of the PAN component users.

34. It shall be possible for the PM to disable authorisation capability of AAs.

6 Challenges for the initial security architecture

We started this work with a very broad technological scope and our survey study also contains several very different technologies. This has given us a good foundation for the next steps. In the continuation of our work, we will now narrow our scope and focus on refining our PAN model. Below we identify the key issues for the initial security architecture.

We expect future terminals to consist of distributed components and to be dynamically reconfigurable. In order to describe different terminal systems we need to define a general architecture for our terminal. For this we have introduced a simple terminal model based on PAN concepts, and using this simple model, we have outlined a number of usage scenarios to be covered as guidance to the scope of our work with a new security architecture for future terminals. To some extent, existing technologies can support our vision of future terminals and the usage cases we have identified, but no single technology includes the complete set of features expected from a future terminal system. Furthermore, the security of existing technologies does not satisfy the security requirements we have identified.

There are a number of open questions to be resolved in the next phase of the work concerning the precise scope of our terminal-PAN and the required functional and security characteristics of the components.

In section 6.1, we give the conclusions from our survey study. No single technology can be used as a reference model for our future work. Hence, instead of identifying security issues for some example existing technologies, we list generic security problems that will be tackled in our future work. We identify the aspects of existing technologies that we will include into our new PAN-based reference model.

In order to work with the main security issues expected from future terminals, we need a well-defined, sufficiently detailed reference model of a distributed terminal system. Our simplified PAN model does not fulfil these requirements. Hence, we need to develop a new, extended PAN reference model. In section 6.2, we describe the main features expected from our new reference model. We discuss the work that needs to be performed to develop this new model.

The threat analysis of our different cases shows that in order to counter potential threats to the basic terminal configurations new security measures need to be developed. In section 6.3, we list the main issues for secure control and automation of the terminal and our direction for future work. Furthermore, it follows from our threat analysis that we need to secure the local communication links in the PANs. The main issues for future work regarding securing the internal communication for PAN is described in section 6.4.

In section 5 we described terminal security requirements from the perspectives of the roles we have identified. In order to fulfil these requirements, a secure execution environment applicable to our PAN model must be present. In section 6.5, we list the challenges we face in our security architecture for secure configuration of PAN components.

We end this section by describing access control issues in a distributed terminal system. The different user scenarios and the survey study were used as the basis for identification of the access control problems for further investigation.

6.1 Conclusions from survey study

In our survey, we have given a short description of each system. We have described the technological environment for the system and the security services provided by the system. Technical features as well as creation of the security context and trust model were covered for the different technologies. Finally, future trends for each technology have been identified. This study has given us a nice picture of the state-of-the-art and possibilities of security for distributed terminal systems. Below we give our conclusions regarding the different categories of previous work.

6.1.1 Internal communications specifications and research

A new trust model will be part of our PAN reference model. When we define the trust model the research work (as described in the survey) on *ad hoc* trust models will be used as input. The research work on key agreement in *ad hoc* networks will be used when we consider different key exchange options for securing the internal PAN communication.

The Internet specifications for *ad hoc* networking contain several different security services. We have listed the security features provided by different *ad hoc* routing techniques and the overall security of the different *ad hoc* routing proposals (MANET). Security for *ad hoc* routing protocols was also covered in our research survey study. Still there are too many open problems in the area of *ad hoc* routing and solving general *ad hoc* routing security problems is not within our scope. This also means that large *ad hoc* networks with advanced routing mechanisms will *not* be included in our new reference model.

In our study of the IETF ZEROCONF group, we saw that there is a need for securing the basic IP configuration. We will study how to secure configuration of PAN units (see section 6.3 and section 6.5 below). We will take the ZEROCONF security requirements into consideration when we design our security architecture.

The ARIB MT-TA interface gives a low level interface description for communication between PAN units. We will define a reference model that is independent of any particular interface. The security problems identified in the MT-TA study will be addressed by our future work and are listed below (section 6.3 and section 6.4).

6.1.2 Specifications for external network and applications

We have given a survey of the WAP, I-mode and MeT technologies. We will develop a security architecture that fulfils the security requirements of these as well as future mobile applications. Our architecture should be as flexible as possible and the built-in security mechanisms in different application environments should be utilised whenever appropriate. By developing a security framework that allows flexibility about where (i.e., at which layer in the communication stack) to enforce security and different security mechanisms, we allow re-use of existing application security as well as network and link level security mechanisms like the one in Bluetooth.

The handling of downloadable executables gives rise to demanding security problems. Our survey of MExE and TeSSA describes many of the issues and their possible solutions. Most of the requirements we have identified in our role-based model in section 5 also relate to how to create a secure but still flexible execution environment. We are going to work on issues unique to our PAN reference model. To as large an extent as possible, we will re-use the work already performed within MExE. The challenges for our future work with a secure environment are described in section 6.5.

6.1.3 Programming languages and general purpose service discovery applications

The IETF service location protocol (SRVLOC) contains mechanisms for securing the needed protocol PDUs. Service discovery mechanisms are definitely to be included into our PAN reference model. However, SRVLOC is supposed to work within networks under co-operative administrative control. This is not true for all the user scenarios we have listed. One important task within our future work is to identify the requirements regarding securing service discovery information and mechanisms. There are two different possible security architecture assumptions that can be used:

- all service discovery messages and services are transmitted without any security and the service information is totally open;
- the service discovery mechanisms are secured with mechanism like those used in SRVLOC.

When using the first approach, one makes sure that the service discovery mechanisms *as such* are available to anybody. Instead one can focus on securing the services themselves. However, then there is a risk of denial of service attacks and misbehaviour of the service providers.

UPnP, Java/Jini and E-speak are other service discovery techniques. Different from SRVLOC, these techniques contain mechanisms for the service itself. This makes their security analysis more complex. On the other hand if we separate the pure service discovery from the service delivery, they have the same characteristics as SRVLOC. Our goal is to define a reference model that is independent of any particular service discovery technique and (if possible) service model. The different problems that we will work with are described in sections 6.3, 6.4, 6.5 and 6.6.

6.2 PAN reference model

Our initial, rough PAN model has been useful to describe usage scenarios and the communication situations that we target in this project. However, to be able to develop a flexible and applicable initial security architecture a more fine-grained PAN model needs to be developed. Our initial PAN model should be extended with the following:

- A trust model
- A general service model
- Service discovery principles
- Infrastructure dependency
- The model for configuration and executable provisioning including possible provisioning sources.
- Categorisation in terms of main technological features of the different types of PAN components we are targeting

One major problem in ad hoc network is the lack of supporting infrastructure and pre-defined trust relations between the different PAN components. Short-term and long-term trust relations need to be created dynamically. Trust can be on user, service, or device level. In order to perform efficient access control and define communication security policies, we need to develop a trust model applicable to our PAN reference model. We have made a survey of scientific works dealing with trust in ad hoc networks. Some of the previous research results might be applicable to the PAN reference model. Our trust model will be based on the role model we have described in section 5. The needed security facilities will depend on the trust model.

We should define what we mean by a service. A service can be both a communication service and a computing service. Our definition should probably include both types. A communication service could be anything from a radio communication link to an application to application connection.

The grade of infrastructure a PAN can assume is important for the security mechanism that can be used. A PAN can have several "infrastructures" available, e.g., medium (hardwired, wireless, partially hardwired), administration (central, de-central), key management (pre-shared keys, PKI). Autoconfiguration or forming of an ad hoc network depends on the grade of infrastructure available.

We need to give a description on how PAN components find services within the PAN. Our model should be applicable to state-of-the-art service discovery techniques.

In order to define secure executable download mechanisms we must define the basic configuration and executable provisioning mechanisms that we consider.

It might be useful to include a categorisation of the different PAN components in our reference model. Different components will have different (secure) execution and storage capabilities or other limited resources. The capabilities might affect our security architecture.

6.3 Control and automation of dynamic configuration of distributed terminals

The configuration and control refer to the behaviour of executables at the application layer. More specifically, configuration could be identified with the ability to specify, (either by the user or by an

application) on the terminal a certain behaviour for a given executable under a set of conditions before the launch of the executable.

The configuration and control of the distributed terminal (PAN) will involve some kind of user interaction during its entire life cycle from the setup of the network to its termination. The PAN user may be required to have the capabilities of PAN manager or PAN component administrator as defined in Section 5. It is assumed that all communication between the component and the user shall be via facilities provided by the component.

In addition to routine exchange of information, user intervention could involve seeking user permission for some PAN component to join the PAN, to access some resource, or to perform a certain action. Also, the component should keep the user informed accordingly if a decision is required. For example, the component should be capable of informing the user on the status of a PAN component.

A designation of an entity as “trusted” is not necessarily a universally accepted definition – some entities may accept the designation and others will not. Therefore it is important that it is possible to configure the terminal so that designations of trust are treated according to the owner’s requirements. The owner may be an individual user or may be a corporate owner that controls a number of components in use by employees of the corporation.

With a large number of potential service providers and third party components, both trusted and untrusted, it will be an impossible task for the user to know inherently which service provider’s or component’s applications were badly written or have caused the component to crash, for example. The component, on the other hand, could use the information from previously logged events in determining application behaviour.

The component shall be able to build a local trust profile for a selected number of ASP's/components based on the behaviour of their previous applications/interactions on the component. The information on application behaviour could either be located on the component or on a trusted third party server. The component shall use this local trust status along with any other trust status (that may arise due to the presence of a valid digital signature) in allocating resources to applications.

The challenge to the PAN security architecture is that the control and configuration application provides sufficient intelligence to inform/warn the user or even to prevent the user from certain configuration and control actions once it has sufficient evidence to do so. In order to achieve this goal, it may be necessary that the user shall be able to request basic information about the component, e.g. the trust status, and access rights. To meet the challenge, the architecture may involve a special PAN component database for this purpose.

The architecture should not be limited to a model where configuration and control is performed by a human user. Therefore, it should be allowed that the role of PAN manager and the role of PAN component administrator is given to an application (automated configuration and control).

Also, a distinction should be made between autonomous PANs that perform configuration and control independently, on the one hand, and between externally controlled PANs. The latter model may be imposed by an external application, which has special requirements concerning which components constitute the distributed terminal, and how each individual component is configured.

In most cases human intervention is still required to perform some configuration and control actions. A challenge to the architecture is to keep the user interaction involving security decisions to a minimum.

6.4 Internal communication security for PANs

In this section, the PAN internal requirements for the authentication of different components and the protection of the “local” radio network as well as communication between personal components and subscription modules are discussed. The security architecture should provide a framework, which is

able to incorporate security solutions including key management applications and other security protocols, as well as mechanisms for authentication and protection of the local radio communication.

6.4.1 Internal security requirements

A basic expectation on a distributed terminal from the terminal user is that sensitive information from the user should not be revealed without the user's consent. Sensitive information might be information stored on any one of the terminal components, or information transferred between terminal components. Sensitive information includes the identity of the user as well as the geographical position of the user. We categorise the requirements into four main parts

- Authentication
- User identity and location privacy
- User data privacy and integrity
- Signalling data privacy and integrity

6.4.1.1 Authentication

It must be possible for one PAN component user, PAN component owner or PAN manager to authenticate another PAN entity. Furthermore, in order to apply service level authorisation we need to consider whether service level authentication also should be provided. The authentication can be performed at different layers in the communication stack. Authentication at one layer might be sufficient for some use cases but not for other. Lower layer authentication mechanism might be utilised by applications to avoid authentication at several layers.

6.4.1.2 User identity and location privacy

We make a distinction between the user identity, which is the identity claimed by the user's subscription module when requesting network access, and component identity, which is the identity of the component. Not all components will have a component identity.

User identities are used in several different places in distributed terminals. A user identity can be used to authenticate a user before the user is getting access to the PAN. User identities are necessary for fundamental system security requirements to be fulfilled. Hence it is necessary that a user identity is available for the terminal equipment. However, it is important that user identities do not compromise the user identity privacy requirements. Unique user identities can often be replaced by *temporary* user identities that will not compromise the privacy of the user.

The PAN security architecture shall take the following requirements into consideration:

- The user terminal should only store a minimal number of unique user identities on the terminal.
- Transmission of a user identity over an external interface in clear text should be avoided if possible. It should be possible for the user to require that whenever a user identity is transferred over an external interface the user should be made aware of the transfer.
- A user identity that needs to be transmitted in clear should be chosen so that there is no relation between other user identities (e.g. name, ID-numbers, addresses etc).
- User identities stored in terminal equipment should be encrypted or should be stored in a tamper resistant module.

Even if user identities are completely avoided, the user's privacy might still be compromised, since the communication **components** might have unique identifiers. If it somehow were possible to *link* the component specific identifiers (e.g. component addresses) with a particular user, the user's privacy would be compromised. Linking user identity with, for example, a component address can be implemented in a variety of ways. Unique component identities can often be replaced by *temporary* component identities that will not compromise the privacy of the user. Therefore it is further required

that transmission of a unique (or almost unique) component identifier over an external interface in clear text should be avoided if possible.

6.4.1.3 User data and communication confidentiality and integrity

In order to perform desired user communication tasks, sensitive user data will need to be between the PAN components in a distributed terminal. Intruders might eavesdrop on user data traffic or tamper with data stored on user components. As the user may be relying on the fact that her data is transmitted confidentially, if there is some way for the data not to be sent confidentially, the user must be made aware if confidentiality is not used.

As mobile devices become more like PDA's and PC's, some users may store significant amounts of data on their mobile device. It must be possible for the user to protect this information from eavesdropping from those with (temporary) possession of the mobile device. The PAN security architecture must involve methods for protecting the confidentiality of the user traffic. It shall also be possible for the users to check whether or not the communication is confidentiality protected. Such protection methods are implemented and executed within the security context that is set up especially for this purpose. The mechanisms typically involve authentication of the communication parties, derivation of the encryption key, and finally, encryption and integrity protection of the communication.

The security architecture must also support secure storage of sensitive user information on the PAN component, possibly requiring a tamper resistant module connected to the PAN component.

Requirements on user data integrity protection will be investigated and solutions will be provided in our security architecture.

6.4.1.4 Signalling data confidentiality and integrity

Signalling data relating to a particular user (for instance, location update) may reveal important information about the user, for instance, their location, and details of who they are calling, for how long etc. This signalling data must therefore be protected from unauthorised access.

Requirement on signalling data integrity protection will be investigated and solutions will be provided in our security architecture.

6.4.2 Initialising security context for internal communication

Initialisation of the internal PAN security context is performed by a key management application. When the PAN is configured or reconfigured to allow terminal components to join the PAN some initial access control is performed. To perform access control some initial identification of the new terminal component is performed. The security architecture shall support various methods for performing the initial identification. Typical examples in practical situations include a large variety of methods, from simple physical identification performed by a human user to sophisticated cryptographic mechanisms based on pre-distributed cryptographic keys possibly involving a PKI.

At the access control step the rights of the new user and her terminal are determined according to the PAN security policy. Among other things the security policy defines what kind of cryptographic keys and mechanisms the new component is using for securing its communication with other PAN components. This may also require a negotiation procedure between the PAN manager and the PAN component administrator.

The PAN security architecture shall support various key establishment procedures for initialising the security context for securing internal PAN communications. One typical case is where the internal communications security is based on the same pre-distributed keys that were used for identification at the access control step. If the access control identification was performed by physical means, then it may be a natural choice to use user-assisted key agreement mechanisms. Fully automated security initialisation requires pre-established security context such as PKI or pre-distributed symmetric keys.

The more ad hoc communication is supported the more online user intervention is required to set up the initial security context.

A major challenge to the security architecture is to allow key agreement methods that are user friendly without compromising security. The selection of the key agreement mechanism to set up the initial security context depends also a lot on the general topology of the internal PAN communication.

6.5 Secure execution environment in a distributed terminal

The first step towards a secure execution environment for a distributed terminal is a secure execution environment in the terminal components. Some of the relevant issues for single units have already been mentioned in section 2.3. Section 4.3 presented MExE, a proposal for secure execution environment in a monolithic configurable terminal.

MExE provides a good starting point for a component-level secure execution environment. Further investigation should establish if MExE is flexible enough to meet the requirements listed in section 5, and extend the model as necessary.

Once a secure execution environment is available in the different components, the challenge is to integrate the individual execution environments into a secure execution environment for the distributed terminal building on internal communication security and PAN-wide access control mechanisms. Section 4.10 presented TeSSA, a proposed approach geared towards a distributed setting. TeSSA uses remote call mechanisms with proxies to “internalise” functionality found in other units. Further work is required to determine if this is the most suitable method of interaction between the individual secure execution environments.

The use of a PAN-wide unified permission/access control mechanism for all aspects of the distributed terminal (including PAN communication, configuration, execution, verification of content) would seem preferable. This seems to imply that every component should implement some minimal level of security functionality. For a distributed secure execution environment to be effective (or even possible) the necessary minimal level should probably include a mechanism to unambiguously identify components that belong to the PAN, and establish their intrinsic trust levels preferably with some assurance. Manufacturers (or some certification body) may provide such assurances for their components and their execution environments, much the same way as downloadable content is certified. Naturally, these default trust levels may need to be readjusted based on PAN and component policies, taking into account e.g. the security of the communication link.

An architectural issue closely related to secure execution environments is the balance of peer-to-peer vs. master-slave models. Should there be centralised control, or should the secure execution environment be distributed, and to what degree. The division of tasks in a distributed terminal is also a question for further study. Who implements checking/screening/profiling/logging? In fact, some of this functionality may even be delegated to entities “remote” from the PAN. Another configuration and control issue with relevance for a distributed secure execution environment is service discovery. Section 4 presented a number of proposals (Jini, UPnP, E-speak). Results of that review with appropriate extensions and modifications should be incorporated in the proposed security architecture.

Common to both MExE and TeSSA are the use of signed content and a degree of reliance on Java security. Issues regarding public key infrastructures have their own dedicated work package. As possible input to them, it appears necessary to investigate:

- the applicability of public key vs. symmetric key methods;
- the use of local vs. global public key infrastructures;
- the use of authorisation vs. identity certificates.

Perhaps more directly relevant to future work is the question whether a Java virtual machine can provide the necessary level of security, and to what degree and how Java or other Java-related methods, such as Jini or E-speak, should be tied to lower operating system layers.

6.6 Access Control

Access control – as the name implies – refers to a mechanism that controls how subjects may access objects in a system and therefore is a question of authorisation. Such a mechanism can be modelled as a composition of decider and enforcer components (though there may be other ways of modelling access control mechanisms). The decider and enforcer mechanisms may be collocated. Whenever a subject attempts to access an object, the enforcer first invokes the decider with the security attributes of both object and subject; the decider then makes a decision based on the security policy in force and the attributes supplied; and finally, the enforcer enforces this decision.

In the context of distributed terminals, access control is applicable in a number of contexts at various different levels. Access control is instrumental in internal communication security, for example, to restrict access to trusted communication paths to trusted/authorised users/agents/components both internal to a particular component or between components in the distributed terminal. The distributed execution environment also relies on access control to a large extent, and access control is also an integral part of the terminal configuration. User privacy including control over the use of user data is also unthinkable without effective access control mechanisms.

The range of possible actors requesting access includes

- legal/business entities such as manufacturers, operators, users, owners, or security administrators;
- hardware components such as network elements or terminal components;
- software agents such as operating systems, processes, or applications.

Similarly, possible targets of access requests include

- system resources such as bandwidth, storage, processing power;
- devices such as terminal components, peripherials;
- internal and external, hardware and software interfaces such as various ports, radio interfaces, function calls, inter-process communication, etc..

Both of the above lists are likely to be incomplete, but they clearly demonstrate a need for multiple access control domains, possibly with different administrators for different domains. As the different domains may provide conflicting policy decisions for the exact same situation (i.e. same subject/object pair, identical security attributes), a mutually agreeable method of resolving such decisions should be implemented. Depending on the business model, such methods could range from hierarchy-based priority scheme to a simple intersection (basically, veto) rule. The implementation of the (various) enforcer and (prioritized) decider components of the access control mechanism then must be verifiable (and thus, trusted) by all parties.

The different access control component need different security attributes (e.g. user identity and password; or certificate) for each component. Therefore, future work should define the minimal set of security attributes (or examine several alternatives of the used security attributes/techniques) which can be used for one access control component.

It is a major challenge to the security architecture to develop an access control framework that effectively supports multiple access control domains for the distributed terminal as a whole, and for each component of the terminal separately. Such a framework must also involve a mutually agreeable (priority) scheme to resolve conflicting per-domain policy decisions.

As the complexity of security policies grows, administration becomes increasingly difficult, especially so in a distributed environment. Consequently, the possibility of security holes must be considered. To aid in discovering potential threats or in removing exploited vulnerabilities, the access control system should keep proper logs (auditing data) of relevant access-related information.

The access control framework for distributed terminals must address this problem. With increased complexity, the issue of usability must also be addressed. There is clearly a need for a common cross-device security administration tool, which supports configurable access logs. Moreover, it is desirable

that the access control framework for distributed terminals should provide a unified cross-platform policy administration/audit tool.