IST-2000-25350 - SHAMAN

Deliverable Number	D04
Deliverable Title	Initial report on PKI requirements for heterogeneous roaming and distributed terminals
Date of delivery	10-Sep-01
Document Reference	SHA/DOC/RHUL/WP3/D04/1.0

Contractual Delivery Date	31-Aug-2001
Actual Delivery Date	10-Sep-2001
Editor	Chris Mitchell
Participant(s):	Vodafone, Royal Holloway, Siemens ATEA, T-Nova
Workpackage	WP3
Est. person months	
Security	
Nature	
Version	1.0
Total number of pages	92

Abstract:

The purpose of this report is to provide all the necessary background to support the Public Key Infrastructure (PKI) work within Shaman WP3. The main focus of WP3 is the study of PKI issues necessary to support Shaman WPs 1 and 2. Hence this report lays the foundations for a study of PKI requirements for:

- unifying security for heterogeneous access networks (WP1), and
- a unified security architecture for future distributed mobile terminals.

Keyword list:

PKI, mobile telecommunications

Table of contents

TABLE OF CONTENTS 2

EXECUTIVE SUMMARY 6

LIST OF CONTRIBUTORS 7

1 INTRODUCTION 8

1.1 SCOPE AND PURPOSE 8
1.2 CONTENTS OF THIS REPORT 8
1.3 LIST OF ABBREVIATIONS 9

2 CURRENT USES OF PKIS IN MOBILE TELECOMMUNICATIONS 11

2.1 SSL 11
2.1.1 INTRODUCTION 11
2.1.2 PROTOCOL OVERVIEW 11
2.1.3 REALISATION 11
2.1.4 PUBLIC KEY CRYPTOGRAPHY RELATED ALGORITHMS 12
2.1.5 REQUIREMENTS FOR THE UNDERLYING PKI 12
2.1.6 TLS EXTENSIONS 13
2.2 WTLS 14
2.2.1 INTRODUCTION 14
2.2.2 PROTOCOL OVERVIEW 15
2.2.3 PUBLIC KEY CRYPTOGRAPHY BASED SECURITY FEATURES 15
2.2.4 REQUIREMENTS ON THE USAGE OF PUBLIC KEY CRYPTOGRAPHIC TECHNIQUES 15
2.2.5 REALISATION 15
2.2.6 REQUIREMENTS FOR THE UNDERLYING PKI 17

3 EMERGING USES OF PKIS IN MOBILE TELECOMMUNICATIONS 18

3.1 IKE 18
3.1.1 INTRODUCTION 18
3.1.2 PROTOCOL OVERVIEW 18
3.1.3 PUBLIC KEY CRYPTOGRAPHY BASED SECURITY FEATURES 19
3.1.4 REQUIREMENTS ON THE USAGE OF PUBLIC KEY CRYPTOGRAPHIC TECHNIQUES 19
3.1.5 REALISATION 19
3.1.6 REQUIREMENTS FOR THE UNDERLYING PKI 19
3.2 MEXE 19
3.2.1 MEXE CLASSMARKS 20
3.2.2 MEXE SECURITY FRAMEWORK 20
3.3 DRM 26
3.3.1 IDENTIFICATION OF COPYRIGHTS 27

3.3.2 PROTECTION OF THE CONTENT 28
3.4 SDR 28
3.4.1 DEFINITION 28
3.4.2 SECURITY ISSUES 29
3.4.3 PKI ISSUES 29
3.5 APPLICATION LAYER USE OF PK IN WAP 30
3.5.1 CLIENT GENERATED DIGITAL SIGNATURES, *SIGNTEXT* 30
3.5.2 CLIENT VERIFICATION OF DIGITAL SIGNATURES, *SIGNED CONTENT* 31
3.5.3 CLIENT PUBLIC KEY ENCRYPTION 31

4 FEATURES PROVIDED BY CURRENT AND EMERGING PUBLIC KEY TECHNIQUES 33

4.1 AUTHENTICATION 33
4.2 INTEGRITY 34
4.3 CONFIDENTIALITY 34
4.4 AUTHORISATION 34
4.4.1 DISCUSSION ON THE DEFINITION OF 'AUTHORISATION' 34
4.4.2 METHODS OF PROVIDING AUTHORISATION USING PK 35
4.4.3 AUTHORISATION WITH X.509 PUBLIC KEY CERTIFICATES 35
4.4.4 AUTHORISATION USING ATTRIBUTE CERTIFICATES 35
4.5 KEY MANAGEMENT 39
4.6 NON REPUDIATION 40
4.7 STATUS INFORMATION 40
4.8 TIMESTAMPING SERVICES 43
4.8.1 STANDARDS DEVELOPMENT 43
4.8.2 THE STANDARDISED TIMESTAMPING PROTOCOL 44
4.8.3 EXTENDING TIMESTAMPS 44
4.8.4 VERIFYING TIMESTAMPS 44
4.8.5 ALIGNING TIMESTAMPS 45
4.9 XML KEY MANAGEMENT SPECIFICATION (XKMS) 45
4.9.1 DESCRIPTION 45
4.9.2 EXAMPLE USE OF XKMS OVER A WAP SESSION 46
4.9.3 FEATURES 46
4.9.4 ISSUES 47

5 WP1 REQUIREMENTS FOR PKI 48

5.1 INTRODUCTION 48
5.2 PUBLIC KEY AND PKI RELEVANT REQUIREMENTS 48
5.3 DERIVED PKI REQUIREMENTS 50
5.4 CONCLUSIONS 55

6 WP2 REQUIREMENTS FOR PKI 57

6.1 THE SECURITY REQUIREMENTS FROM D03 57
6.2 PUBLIC KEY AND PKI RELEVANT REQUIREMENTS 57
6.3 DERIVED PKI REQUIREMENTS 59
6.4 CONCLUSIONS 63

7 ORGANISATIONAL ISSUES 65

7.1 KEY MANAGEMENT ISSUES 65
7.1.1 KEY PAIR GENERATION 65
7.1.2 KEY STORAGE 66
7.1.3 KEY BACKUP, KEY RECOVERY 66
7.1.4 KEY UPDATE 66
7.1.5 KEY HISTORY AND ARCHIVE 67
7.2 CERTIFICATE MANAGEMENT ISSUES 67
7.2.1 REGISTRATION 67
7.2.2 CERTIFICATE GENERATION AND DISTRIBUTION 68
7.2.3 CERTIFICATE DISSEMINATION AND RETRIEVAL 68
7.2.4 CERTIFICATE VALIDATION 68
7.3 TECHNOLOGY CONSTRAINTS 69
7.3.1 STORAGE CONSTRAINTS 70
7.3.2 PROCESSING CONSTRAINTS 70
7.3.3 BANDWIDTH CONSTRAINTS 71
7.4 INTEROPERABILITY ISSUES 71
7.4.1 SIGNATURE POLICIES 72
7.4.2 CERTIFICATE TRANSLATION 72

8 REQUIREMENTS ON PKI-INVOLVED PARTIES 74

8.1 'MOBILE NETWORK' OPERATORS 74
8.2 MANUFACTURERS OF MOBILE DEVICES 74
8.3 TRUST-SERVICE PROVIDERS 74
8.4 THE USER 75
8.5 THIRD PARTY APPLICATION PROVIDERS 75
8.6 PROVIDER FOR MOBILE SERVICES (NOT OWNING A NETWORK) 75

9 PKI ISSUES FOR FUTURE MOBILE SYSTEMS 76

9.1 PKI TECHNOLOGIES NOT REQUIRING MAJOR NEW RESEARCH 76
9.2 PROBLEMATIC AREAS 76
9.2.1 COMPLEXITY OF PKI FOR LIMITED DEVICES AND FOR SMALL TRUST MODELS 76
9.2.2 THE COLLISION OF TRUST MODELS AND COMMERCIAL MODELS 77
9.2.3 REVOCATION OF PUBLIC KEYS AND PUBLIC KEY CERTIFICATES 77
9.2.4 INTEROPERABILITY OF PUBLIC KEY CERTIFICATES AND PKIS 78
9.2.5 PKI-BASED AUTHORISATION ISSUES 78
9.2.6 THE USE OF X.509 EXTENSIONS 78
9.2.7 NON-COMPLIANT PROVIDERS AND USERS OF PKI 79
9.2.8 THE EFFECT OF CERTIFICATE LIFETIMES 80

ANNEX A CERTIFICATE TYPES 81

A.1 THE DEVELOPMENT OF PUBLIC KEY CERTIFICATE STANDARDS 81
A.2 X.509 CERTIFICATES 81
A.3 THE PKIX CERTIFICATE PROFILE 83
A.4 WTLS CERTIFICATES 85
A.5 SPKI CERTIFICATES 86
A.6 PROPRIETARY CERTIFICATE FORMATS 87
A.7 UNDERLYING TECHNOLOGY 88

REFERENCES 89

Executive summary

This report provides the necessary background to support the Public Key Infrastructure (PKI) work within Shaman WP3. The main focus of WP3 is the study of PKI issues necessary to support Shaman WPs 1 and 2. Hence this report lays the foundations for a study of PKI requirements for:

- unifying security for heterogeneous access networks (WP1), and
- a unified security architecture for future distributed mobile terminals (WP2).

The report contains five main parts.

The first part (contained in Chapters 2 and 3) reviews current and emerging uses of PKIs in mobile telecommunications. This includes summaries of the uses of PKI by SSL (TLS), WTLS, IKE, MExE, DRM and SDR.

The second part (Chapter 4) considers the services that can be supported by PKIs, notably authentication, integrity, confidentiality, authorisation, key management, non-repudiation, status information and timestamping services.

The third part (Chapters 5 and 6) summarises the main requirements identified in Work Packages 1 and 2 of the SHAMAN project. The requirements are categorised according to the services described in Chapter 4.

The fourth part (Chapters 7 and 8) considers the requirements imposed on entities providing PKI services. Chapter 7 addresses organisational issues, including key management, certificate management, technology constraints, and interoperability issues. Chapter 8 considers requirements on specified types of entity involved in the provision of PKI services, including network operators, device manufacturers, users, third party application providers, and mobile service providers.

The fifth and final part (Chapter 9) highlights the main issues identified in this report. These issues include: public key revocation, interoperability of certificates and PKIs, authorisation issues, the use of X.509 extensions, and non-compliant providers and users of PKI.

List of contributors

Name		Affiliation	Email	Phone
Chandrasiri	Pubudu	Vodafone Ltd. (VOD)	pubudu.chandrasiri@vodafone.com	
Dankers	Jozef	Siemens ATEA nv (ATEA)	jozef.dankers@siemens.atea.be	+32 14 253218
Garefalakis	Theo	Royal Holloway University of London (RHUL)	theo.garefalakis@rhul.ac.uk	+44 1784 414160
Mitchell	Chris	Royal Holloway University of London (RHUL)	c.mitchell@rhul.ac.uk	+44 1784 443423
Schaffelhofer	Ralf	T-Systems Nova GmbH (TNO)	ralf.schaffelhofer@t-systems.de	+49 6151 833044
Schwiderski-Grosche	Scarlet	Royal Holloway University of London (RHUL)	scarlet.schwiderski-grosche@rhul.ac.uk	+44 1784 414346
Schmitz	Roland	T-Systems Nova GmbH (TNO)	roland.schmitz@t-systems.de	+49 6151 832033
Sondh	Jagjeet	Vodafone Ltd. (VOD)	jagjeet.sondh@vodafone.com	
Wright	Tim	Vodafone Ltd. (VOD)	timothy.wright@vodafone.com	+44 1635 676456

1 Introduction

1.1 Scope and purpose

The purpose of this report is to provide all the necessary background to support the Public Key Infrastructure (PKI) work within Shaman WP3. The main focus of WP3 is the study of PKI issues necessary to support Shaman WPs 1 and 2. Hence this report lays the foundations for a study of PKI requirements for:

- unifying security for heterogeneous access networks (WP1), and
- a unified security architecture for future distributed mobile terminals (WP2).

1.2 Contents of this report

Following this introductory section, the first main part of this report (Chapter 2), reviews how PKI facilities are used in current mobile telecommunications protocols (notably within SSL/TLS and WTLS). This is followed, in Chapter 3, by a description of emerging uses of PKI in mobile telecommunications systems. Chapter 4 then describes the features provided and supported by current and emerging PKIs. These three chapters set the scene for the remainder of the report.

Chapters 5 and 6 review the PKI requirements emerging from work packages 1 and 2 of the Shaman project. These two work packages provide the main raison d'être for this work package, and hence these requirements are of fundamental importance here. When put with the requirements identified in Chapters 2 and 3, this provides the primary motivation for the work to be performed within this part of the Shaman project.

Further PKI requirements are identified in Chapters 7 and 8. Specifically, Chapter 7 focuses on organisational issues that arise for the providers of PKI services, and Chapter 8 considers requirements impacting consumers of PKI services.

The main report concludes in Chapter 9, where the main research issues for this workpackage are identified and briefly discussed.

Finally, an Annex lists and briefly describes various certificate formats of current importance.

1.3 List of abbreviations

Table 1 – List of abbreviations

Term	Definition
AAA	Authentication, Authorisation and Accounting
APDU	Application Protocol Data Unit
API	Application Program Interface
ASN.1	Abstract Syntax Notation 1
ASP	Application Service Provider
BER	Basic Encoding Rules
CA	Certification Authority
CCM	Certificate Configuration Message
CER	Canonical Encoding Rules
CLDC	Connected Limited Device Configuration
CPS	Certification Practice Statement
CRL	Certificate Revocation List
DER	Distinguished Encoding Rules
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
DPD	Delegated Path Discovery
DPV	Delegated Path Validation
DRM	Digital Rights Management
DSA	Digital Signature Algorithm
DTMF	Dual Tone Multi Frequency
EEMA	European Electronic Messaging Association
GPRS	General Packet Radio Service
HTTP	HyperText Transfer Protocol
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
IPsec	IP security
MAC	Message Authentication Code
ME	Mobile Equipment
MExE	Mobile Execution Environment
MIDP	Mobile Information Device Profile
MS	Mobile Station – a MExE term

OCSP	Online Certificate Status Protocol
PAN	Personal Area Network
PEM	Privacy Enhanced Mail
PER	Packed Encoding Rules
PIN	Personal Identification Number
PK	Public Key – usually used as an abbreviation for Public Key Cryptography (PKC)
PKI	Public Key Infrastructure
PKIX	The family name for the IETF PKI standards
PM	PAN Manager
PSE	Personal Security Environment
RFC	Request For Comments (the title given to IETF standards track documents).
RSA	Rivest-Shamir-Adleman – a public key cryptosystem
SA	Security Association
SCVP	Simple Certificate Validation Protocol
SDMI	Secure Digital Music Initiative
SDR	Software Defined Radio
SGC	Server Gated Cryptography
SHA-1	Secure Hash Algorithm revision 1 – a standardised cryptographic hash-function (see, for example, [41]).
SIM	Subscriber Identity Module
SMS	Short Message Service
SPKI	Simple PKI
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security – see also SSL
UMTS	Universal Mobile Telecommunication System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WAP	Wireless Application Protocol
WTLS	Wireless Transport Layer Security
XKMS	XML Key Management System
XML	eXtensible Markup Language

2 Current uses of PKIs in mobile telecommunications

The aim of this chapter is to provide a survey of the uses of public key cryptography (and supporting PKI) in security schemes or protocols currently used by applications in mobile environments. It identifies the requirements posed on the usage of public keys and on the underlying public key infrastructure. Identified security schemes are as follows.

- The SSL/TLS protocol, which is the basis for WTLS. Within IETF some extensions to TLS (extension fields in the TLS hello messages) are specified which make it more amenable to use within wireless environments,
- The WTLS protocol tailored to provide transport end to end security for WAP.

2.1 SSL

2.1.1 Introduction

Secure Sockets Layer (SSL) is a protocol that provides a secure channel between two communicating entities. SSL runs over a *reliable* transport protocol, i.e., a transport protocol that guaranties that all packets are delivered exactly once and in the right order. By far the most widely used such protocol is TCP.

The security features that SSL provides are authentication, data integrity and data confidentiality. In order to achieve those goals, SSL employs authentication and key agreement mechanisms to establish security associations (now called sessions), and cryptographic mechanisms to ensure confidentiality and data integrity. The secure channel that SSL provides is transparent, which means that the data that is written at the one end is exactly the data that is read at the other end. This feature makes SSL a very attractive security solution, as nearly every protocol that can run over TCP can be run over SSL with only slight modifications.

Our description is based on the book SSL and TLS [69].

2.1.2 Protocol overview

In this section we give an overview of SSL. For clarity, we will refer to the entity that initiates the connection as the client and the entity that responds as the server.

SSL operates in two phases, the handshake and the data transfer. The handshake authenticates the server to the client, and establishes the cryptographic algorithms and the keys that are to be used in the subsequent data transfer phase. Client authentication is optional, and is performed only by request of the server. We note that in practice it is the client who provides sensitive information to the server. Therefore, the server does not need to know who the corresponding party is. However, situations might arise in which not everyone is allowed to access the server. In those cases, the server has the option to request the client to authenticate itself before establishing a session.

2.1.3 Realisation

We turn now to a more detailed description of the two phases of SSL.

2.1.3.1 Handshake

The purpose of the handshake is to establish an SSL session, i.e., a security association between the client and the server. Optionally, the server may authenticate the client. Those tasks are accomplished using the following steps.

1. The client sends the server a list of the supported cryptographic algorithms together with a random number that will be used later for key generation. Note that this random number is not (and need not be) secret.

2. The server chooses the set of algorithms that will be used, and sends the set back together with the server's certificate, and a random number that will be used later for the key generation.
3. The client uses the server's certificate to authenticate the server's identity, and also extracts the server's public key. It then generates an unpredictable string, called the *pre-master-secret*, encrypts it using the server's public key, and sends it to the server.
4. The client and the server compute the encryption and the MAC keys using the *pre-master-secret* and the client's and server's random values.
5. The client sends a MAC of all the messages so far.
6. The server sends a MAC of all the messages so far (including the message of step 5).

The above protocol achieves its goals since:

- The cryptographic algorithms to be used are agreed upon (steps 1 and 2);
- The keying material is securely exchanged, since it is encrypted using the server's public key (step 3).

Steps 5 and 6 are present to ensure that the communication has not been tampered with; that is, the MACs of the handshake messages ensure that the messages that the client and the server have received are the ones that were sent by the server and the client respectively (no messages have been deleted or inserted). Furthermore, the random values that the client and the server provide are used in the key generation process, which protects against replay attacks.

If the server requires client authentication, three more messages are present in the above scheme. The server sends a ClientAuthentication message, to which the client responds with a Certificate message (containing the client's certificate), and a CertificateVerify message, which contains a string that is signed with the client's private key associated with the certificate.

2.1.3.2 SSL record protocol

The SSL record protocol is the part of SSL that accomplishes the actual data transfer. It works by breaking the data to be transmitted into fragments, which are independently protected and transmitted. On the other end of the channel, each record is independently decrypted and verified. Each record consists of a record *header* (of little interest here), and the *encrypted payload*. The encrypted payload in turn consists of the actual data fragment followed by its MAC (computed using the MAC key that was generated during the handshake phase). Both data and MAC are encrypted using the encryption key. The purpose of the MAC is to ensure data integrity, and the purpose of encryption is to ensure confidentiality.

2.1.4 Public key cryptography related algorithms

As was described in the previous section, the protection of the data is achieved using secret key cryptography. The reason that secret key techniques are preferred to public key when transmitting large quantities of data, is due to their relative efficiency. The first phase of the protocol however, makes essential use of public key techniques to achieve its goals. Here we summarise some of those techniques that are available in SSLv3 and TLS.

In the basic version of the protocol, the server (and optionally the client) is authenticated using a certificate. The most commonly used certificates are signed and contain RSA keys. Thus, the pre-master-secret is encrypted using RSA. There are, of course, other public key methods that meet the same goals. Of those, the Diffie Hellman (DH) protocol for key agreement and the Digital Signature Algorithm (DSA) for digital signatures have been made mandatory for SSLv3 and TLS. Other ciphers, such as elliptic curve-based methods, are also available (but not mandatory).

2.1.5 Requirements for the underlying PKI

The use of public keys requires an infrastructure for issuing, dissemination, retrieval, revocation and validation of certificates. The certificate type that is used by SSLv3 and TLS is the X.509v3 standard

[58]. SSLv3 predates the profile defined by the IETF PKIX working group [20], and there is no profile specified within SSL [11]. It might be expected that server certificates would conform to [20], but an informal study [80] (which can be confirmed by regular use of SSL with all warning options switched on) indicated that in fact many SSL certificates do not conform with [20]. One major reason concerns permission to use 128 bit security. At one time, the US government would only allow 128-bit security to be used for certain financial applications. This was policed by only allowing SSL clients to use 128 bit ciphers if an appropriate server certificate were presented. These certificates could only be issued by VeriSign, and they included a private extension to indicate to Microsoft and Netscape browsers that 128-bit security could be used. For Microsoft browsers, this was known as "Server Gated Cryptography" (SGC). As the certificate was presented after the server had set the cipher to be used in message *ServerHello*, SGC actually involved the initiation of a new SSL handshake. SSL server certificates are also non-conformant with [20] for more mundane reasons, such as serial number formats.

The requirements on such a PKI are imposed by the SSL handshake protocol. The main requirement is that certificates must be integrity protected, although they can be exposed without danger. As they are subject to change they should be downloadable (over the air in case of mobile user equipment). There is no requirement for storing them in a tamper resistant place.

There is no explicit procedure in SSL for the use of certificate revocation lists (CRLs).

2.1.6 TLS extensions

In this section, we describe the relevant TLS extensions, as defined in [5]. The document defines several extensions that address problems that arise mainly in environments where memory, bandwidth, and/or computing power are limited. Typical examples of such environments are wireless and mobile networks. The extensions are backwards compatible, i.e., communication is possible between TLS 1.0 clients that support the extensions, and TLS 1.0 servers that do not support them, and vice versa.

2.1.6.1 The general mechanism

The use of an extension is always initiated by the client. This is done during the handshake protocol, by means of an *extended client hello* message. In this message, the client specifies one or more extensions to be used. In response, the server may send an *extended server hello* message (instead of a *server hello* message), in which the server lists those requested extensions that it is prepared to support. Each extension consists of an extension type and the extension data. The extension types described in [4] are:

- dns_name: allows the client to provide the TLS server with the DNS name of the server they are contacting.
- max_record_size: allows clients to negotiate the maximum record size to be sent.
- client_certificate_url: allows clients to negotiate client certificate URLs.
- trusted_ca_keys: allows clients to indicate which CA root keys they possess.
- truncated_hmac: allows clients and servers to negotiate the use of truncated MACs.
- status_request: allows clients and servers to negotiate the use of OCSP as an alternative to CRLs.

In this document we describe the PKI related extensions: client_certificate_url, trusted_ca_keys, and status_request.

2.1.6.2 Client certificate URLs

This extension is relevant only when client authentication is requested. In this case, TLS specifies that the client is to send its certificate to the server during the TLS handshake. In constrained environments, it may be preferable for the client to send a URL containing its certificate (instead of sending the certificate itself).

To achieve that, the client sends to the server an *extended client hello* message, with extension type "client_certificate_url". The extension data field is empty. If the server is willing to accept certificate URLs, it responds with an *extended server hello* message, with the same extension type, and an empty extension data field.

After the negotiation has been successfully completed, the client may send a *CertificateURL* message instead of a *Certificate* message, which essentially contains a list of one or more URLs, each containing a single certificate and the SHA-1 hash of that certificate. HTTP is the default protocol for retrieving the certificates, and must be supported by servers supporting this extension.

Servers must check that the SHA-1 hash of any certificate retrieved matches the given hash. Once the certificates are retrieved they are processed as usual.

2.1.6.3 Trusted CA keys

Memory constrained clients may possess only a small number of CA root keys. It may be advantageous to indicate to the server which CA root keys they possess, in order to avoid repeated handshake failures.

The client sends an *extended hello* message, with type trusted_ca_keys. The extension data contains a list of the CAs whose keys the client possesses. Each CA root key is identified in one of four possible ways:

- null: no CA root key identity supplied.
- key_hash_sha: contains the SHA-1 hash of the CA root key.
- cert_hash: contains the SHA-1 hash of a certificate containing the CA root key.
- x509_name: contains the X.509 distinguished name of the CA.

2.1.6.4 OCSP status request

In constrained networks, it may be preferable for clients to request the use of OCSP instead of CRLs to save bandwidth. In this case, the client send an *extended client hello* message with extension type status_request. The extension data field contains a list of OCSP responders that the client trusts. If the server is prepared to support this extension, it responds with an *extended server hello* message, with extension type status_request, and empty extension data field. Later in the handshake protocol, the server may send a *CertificateAndOCSPResponse* message in place of the *Certificate* message. The *CertificateAndOCSPResponse* message contains the certificate and the OCSP response (using the ASN.1 *OCSPResponse* defined in [26]). The client must then process the certificate as if it was received in a *Certificate* message. The client should also process the OCSP response and it should abort the handshake if the response is not satisfactory.

2.2 WTLS

2.2.1 Introduction

The WTLS (Wireless Transport Layer Security) protocol is a security protocol, designed for securing communications and transactions over wireless networks. It is used with the WAP transport protocols to provide transport layer end-to-end security. The security services that are provided by the WTLS protocol are *privacy*, *data integrity* and *authentication* between two communicating applications, the WAP client and WAP server.

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 provides functionality similar to the Internet transport layer security systems TLS and SSL, but has been optimised for use over narrow-band communications and incorporates new features such as *datagram support, optimised handshake* and *dynamic key refreshing*.

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

2.2.2 Protocol overview

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

The *record layer protocol* takes the service primitives, from the higher layer client, to be transmitted, optionally compresses the data, applies a MAC (*data integrity* protection), encrypts (*privacy* protection) and transmits the result. Messages received from the peer node are decrypted, verified, decompressed and delivered, via service primitives, to the higher layer client.

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. The protocol operates on top of the record layer protocol.

2.2.3 Public key cryptography based security features

The WTLS protocol uses public key cryptographic techniques to generate a shared pre-master secret (*dynamic key management*) and to authenticate the client (*identity authentication*), hosted in the mobile user equipment, and the WAP server. Digital certificates certify the binding of a public key to an identity. 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 eavesdroppers and active man-in-the-middle attackers. See [78], appendix B for a more detailed description.

2.2.4 Requirements on the usage of public key cryptographic techniques

To guarantee optimum security, sensitive data has to be stored on, and security functionality using this sensitive data (e.g. cryptographic operations) needs to be performed by a tamper-resistant device, so that an attacker cannot retrieve sensitive data. Such data is especially the permanent private keys used in the WTLS handshake with client authentication and the master secrets (based on the pre-master secret), protecting secure sessions, which are relatively long living – which could be several days.

The WTLS protocol must take the requirements set by wireless networks into account:

- Low bandwidth which results in relative slow interactions between client and server, and relatively low transfer rates.
- Limited processing power and memory capacity of the mobile user equipment, which includes smart cards.
- Restrictions on the use of cryptography, e.g. lawful interception may be required.

2.2.5 Realisation

This chapter describes how the public key cryptography based security features are realised by the WTLS protocol, accommodating the requirements on the usage of public key cryptographic techniques.

The establishment of the shared pre-master secret and client and/or server authentication are realised by the handshake protocol, which starts with negotiating the security parameters to be used by the communicating peers. Among these security parameters the "*key exchange suite*" parameter determines the key exchange algorithm, used for the shared pre-master secret establishment. If the key exchange algorithm indicates the use of public keys and the communicating peers require authentication, then the public keys need to be certified.

The certificates are used as follows:

- The WAP client in the mobile user equipment sends a message to the WAP server indicating that a secure session is requested. Amongst other information, the WAP client passes a list of cryptographic key exchange algorithms it supports to the server. Each entry in the list contains a client identifier that allows the server to retrieve a *client* certificate, of which the certificate type is appropriate for the selected key exchange algorithm, or an uncertified public key from its own sources. Supported certificate types are the *X.509v3* certificates, the *WTLS certificates*, which are optimised for size, and the *X9.68* certificates. Additionally, the client passes a list of trusted authorities, with a similar format as the list of cryptographic key exchange algorithms, which identifies the trusted server certificates known by the client.
- The WAP server responds by sending the "selected key exchange suite". If the selected key exchange suite requires public key authentication, then the server sends its *server* certificate. In case the server certificate is signed by a trusted third party, not identified (known) by the client, then a certificate path needs to be established which leads to the certificate identified (known) by the client (i.e. the certificate received in the trusted certificate list). The client to authenticate the identity of the server uses the server certificate. To optimise client processing, the chain should have minimal length. For server certificates, it is possible to have only one certificate: the server certificate certified by a CA of which the public key is distributed independently.

  The server needs the client certificate for client authentication. If the server is not able to obtain the client certificate from its own sources, based on the received client certificate identifier, or from some certificate distribution service (*optimised handshake*), then the server may request the client to send its certificate (*full handshake*). The server may provide the client with a list of the names and types of acceptable certificate authorities. These names may specify a desired identity for a root CA or for a subordinate CA. If no authorities are sent to the client, then the client may send any certificate. A client certificate chain is likely to contain several certificates. This is acceptable, because the server, who is assumed to have sufficient processor capacity, processes the chain.

  If public key techniques are used for the establishment of a shared secret key for the "selected cipher suite", without authentication (anonymous negotiation), then the server is allowed to only send its public key, instead of the server certificate.
- The WAP client verifies that the server certificate is valid and has been signed by a CA whose certificate is stored in the mobile user equipment and is trusted by the user. The client has stored a limited number of public keys for CA that certify server certificates. If the certificate is valid, a unique session key is generated, corresponding to the selected cipher suite, and encrypted with the server's public key.

  If the server has requested a client certificate, the client sends its certificate or its uncertified public key, dependent on the selected key exchange suite. Instead of an actual certificate, the client MAY send a certificate URL. The server SHOULD implement a protection mechanism against denial-of-service attacks based on clients sending an invalid URL.
- The WAP client sends the session key encrypted with the server's public key to the server so that both have a copy. The server will decrypt the message using its corresponding private key and recover the session key. The session key is the basis for the establishment of the shared pre-master secret.

Once the shared pre-master secret (length dependent on the negotiated key exchange method) is established a master secret (fixed length of 20 bytes) is derived at the peers from the pre-master secret and random values, exchanged between client and server. The master secret is passed to the record layer as input for the calculation of the necessary key material (encryption keys, MAC secrets…), for the provision of confidential and tamper-proof communications.

2.2.6 Requirements for the underlying PKI

The use of public keys requires an infrastructure for issuing, dissemination, retrieval, revocation and validation of certificates. The requirements on such infrastructure imposed by the WTLS handshake protocol and how they can be accomplished are not considered by WTLS protocol description [78].

Identified requirements:

- Certificates must be integrity protected, although they can be exposed without danger. As they are subject to change they should be downloadable (over the air in case of mobile user equipment). There is no requirement for storing them in a tamper resistant place.
- The requirements on the use of public key cryptographic techniques imposed by wireless networks, described in chapter 2.4.1.4, also need to be accomplished by the underlying PKI.

3 Emerging uses of PKIs in mobile telecommunications

The aim of this chapter is to provide a survey of the uses of public keys in security schemes or protocols likely to be used by applications in future mobile environments. It identifies the requirements posed on the usage of public keys and on the underlying public key infrastructure.

3.1 IKE

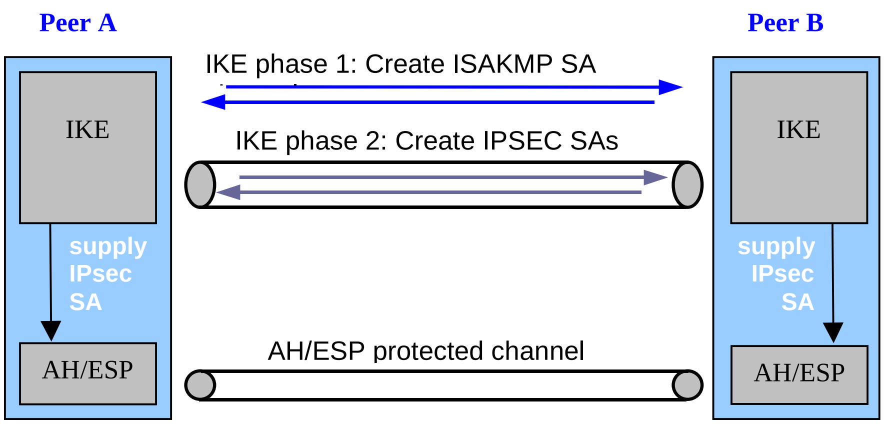
3.1.1 Introduction

The IKE (Internet Key Exchange) is a key management protocol [19], which is used in conjunction with the IPSec protocol [18]. Prior to an IP packet being secured by IPSec, Security Associations (SAs) must exist, which can be created manually or dynamically. The Internet Key Exchange protocol is used to create them dynamically, providing authentication of the negotiating IPSec peers. As such IKE provides automated key management.

3.1.2 Protocol overview

IKE defines three modes of automatic key exchange, each consisting of a fixed number of messages to be exchanged between the peers. The IPSec peers that want to set up SAs between each other take the role of initiator and responder, where both sides can initiate an IKE exchange.

The different key exchange modes can be assigned to two protocol phases. IKE phase 1 creates a so-called ISAKMP SA between the IKE peers, with mutually authenticating the IKE peers. This ISAKMP SA is subsequently used to secure IKE phase 2 which is used to create these IPSec SAs that are required for the AH (Authentication Header) and ESP. Note, that the ISAKMP SA in contrast to an IPSec SA is bi-directional.



The two-phase design mainly follows from performance aspects. IPSec SA negotiation happens in the second phase using the so-called *quick mode* key exchange. This can be run repeatedly and very fast over a single ISAKMP SA from phase 1, which takes more time to be created.

For phase 1, IKE uses an authenticated Diffie-Hellman key exchange based on asymmetric cryptographic mechanisms. The standard currently defines four different methods to authenticate the IKE peers. Three of them are based on asymmetric cryptography, one is based on pre-shared symmetric keys. All these authentication methods can be used either in *main mode*, requiring six messages, or in *aggressive mode*, requiring only three messages. Only main mode offers identity protection for the negotiating nodes. For a detailed description of the different key exchange modes and authentication methods see section 5 of [19].

3.1.3 Public key cryptography based security features

As mentioned above, IKE supports asymmetric cryptography based authentication methods (during the negotiation of an ISAKMP SA), which may be based on the usage of *RSA signatures* or on the usage of *RSA encrypted nonces*.

RSA signatures provide non-repudiation for the IKE negotiation, which means that you can prove to a third party that you had an IKE negotiation with the remote peer. RSA signatures require the use of a Certification Authority (CA).

RSA encrypted nonces provide repudiation for the IKE negotiation, which means that you cannot prove to a third party that you had an IKE negotiation with the remote peer. This authentication method is used to prevent a third party from knowing about your activity over the network. RSA encrypted nonces require that peers possess each other's public keys, but do not use a Certification Authority. Instead, there are two ways for peers to get each other's public keys: manually configured or from a previously used RSA signatures during a successful ISAKMP SA negotiation.

3.1.4 Requirements on the usage of public key cryptographic techniques

As already mentioned in 3.1.2, IKE uses an authenticated Diffie-Hellman key exchange for key management. There are no specific requirements defined in [19] on the use of this key exchange method.

3.1.5 Realisation

This chapter describes how the public key cryptography based security features are realised by the IKE protocol, accommodating the requirements on the usage of public key cryptographic techniques.

Mutual authentication is performed during the negotiation of an ISAKMP SA. Both peers send each other nonces, random numbers the other party must sign and return to prove its identity. With the *RSA encrypted nonces* method the signature is verified using an uncertified public key. With the *RSA signatures* method certified public keys are used for signature verification. Both parties send each other their certificate, they obtained from a CA, and the signed nonce in the payload of the message. **X.509v3 certificates** are used for certifying the public keys.

3.1.6 Requirements for the underlying PKI

When the asymmetric cryptography based authentication method makes use of RSA signatures, then there is need for a PKI. The IKE protocol specifies the use of the PKIX profile of X.509v3 certificates, described in RFC 2459 [20], [21]. Many implementations have not adhered completely to the PKIX certificate profile specification, however. To achieve interoperability in the IPsec market, the Internet draft “A PKIX Profile for IKE” [33] was produced. It lays out a profile for using PKIX in IKE and “advises” that all requirements and suggestions in RFC2459 and other PKI-related documents be followed, except where the Internet draft “A PKIX Profile for IKE” specifies differently.

3.2 MExE

MExE (Mobile Execution Environment) provides a standardised execution environment for both 2nd and 3rd generation mobile devices. Its goal is to allow applications to be developed independently of the device hardware (by 3rd party providers), and then be downloaded and executed in a standardised environment.

The technical specifications of MExE were initially developed by ETSI (the European Telecommunications Standards Institute), and were transferred to 3GPP (3rd Generation Partnership Project) in July 1999. The functional specifications of MExE are given in [1].

There are two main security issues relevant to MExE. One is to ensure that the software that is downloaded to the mobile device is trusted. This trust is based on the trust of the MExE administrator, the operator and the manufacturer on the party that provides the application. This (high level) security requirement is satisfied by means of a PKI. The other security issue is to ensure that a certain application can take only those actions that it is allowed to. This is achieved by restricting the application to run in a sandbox, i.e., within a safe area to execute Java code. Thus, the application's access to resources is restricted by the sandbox within which it runs, which in turn is determined depending on how trusted the application is.

3.2.1 MExE Classmarks

MExE Classmarks, i.e. predefined sets of functionality designed for different categories of MExE device, are defined according to the underlying technology (e.g., Java, WAP), not on the capabilities of the mobile device. However, it is likely that different Classmarks will correspond to devices with different capabilities.

Classmark 1 is assigned to WAP devices with limited processing power. They are expected to have simple text based interfaces, similar to the currently used mobile phones.

Classmark 2 is assigned to devices that support a variant of Java 2 Standard Edition for mobile devices. They are expected to have greater processing power to be able to support a Java run time environment. Compared to the simple mark-up language that WAP devices support, Java is a full-featured programming language, that allows the development of far more sophisticated applications.

Classmark 3 is assigned to devices that support CLDC and MIDP, two new technologies developed by Sun Microsystems. CLDC is targeted at devices with 256K memory and 16/32- bit processors. Those devices are expected to have very tight resource constraints. They will not be able to support a signed content model or the fine-grained permissions of Java 2. However they will be able to implement a Java sandbox for differentiating between trusted and untrusted applications. MIDP next generation is currently working on support for signed content via domain model.

3.2.2 MExE security framework

The core of the MExE security architecture is based on a signed content trust model. Underlying its design are some key principles that include the following:

- The mobile equipment (ME) and the data within the ME are owned by the user of the mobile station (MS);
- The SIM card and operator controlled areas within the ME are owned by the network operator;
- Privacy of user data is given high priority.

MExE executables (applications) are signed with a signature key associated with one of the three MExE security domains described below. The signature on the executable is verified on installation of an executable in the MS. The type of signature key used determines the security domain within which the verified executable will operate. A MExE security domain is essentially a set of MS functionality and network services ('actions') that the executable is given privilege ('permissions') to access, with limits being placed on the functionality or services depending on the domain. Where an executable has permission to perform an action, it does so by calling a function in a MExE API. No device functionality or network service is accessed directly.

Three security domains are defined:

- Manufacturer – executables signed using a MS Manufacturer signature key;
- Network Operator – executables signed using a Network Operator signature key;
- Third Party – all other third party executables, signed using a Third Party signature key.

In fact, it is not a requirement that the operator or the manufacturer sign the applications directly. It is sufficient if the application is signed by a key that can be verified using a certificate chain, that terminates at either the operator or the manufacturer.

Executables belonging to one of the above Security Domains are implicitly regarded as 'trusted'. MExE executables that are not signed, or are signed with a signature key that cannot be verified at the MS (e.g. MS does not store corresponding verification key) are designated 'untrusted'. Untrusted executables operate within a 'sandbox', an environment with very limited access to MS functionality and services.

3.2.2.1 Security domain privileges

MExE executable permissions are assigned across 12 categories of MS functionality and services.

In summary, no MExE executables have access to Device Core Functions, SIM Smart Card Low Level functions or Network Security functions. Executables running in the Network Operator Security Domain have access to all other categories of actions. The MS manufacturer domain has the same privileges as the Network Operator domain except for the Network property access category where the Manufacturer domain has no access. It should be added that the manufacturer domain has the capability to download and install core software on the terminal. This privilege is exclusive to the manufacturer. Finally, the Third Party Security Domain has the same privileges as for the MS Manufacturer domain except for the Network Services category. Further restrictions of privilege can be found in the following table.

No.	***Functionality Category***	***Main Functions***	***Restrictions***
1	Device Core Functions	Start/Stop Radio; Turn Device On/Off; Write Time/Date; Activate/Modify User Profile.	-
2	SIM Smart Card Low Level access	Send APDU (low level smart card command); Slot Management (power on/off, reset, etc); Access to these functions is via a high level API.	-
3	Network Security	Run GSM algorithm; Verify/Unblock/Modify CHV/2; Activate/Deactivate CHV (CHV/2 and CHV are PINs programmed into the SIM card).	-
4	Network Properties	Get International Mobile Subscriber Identity (IMSI); Get Home Network; Select Network.	-
5	Network Services	Initiate/Accept Voice or Data Connection; Call Forward/Deflection; Multi-Party Call; Explicit Call Transfer; Terminate/Hold/Resume an Existing Connection; Send Point to Point Message (e.g. SMS or USSD); Generate DTMF; Query Network Status; QOS Management.	Multi-party, call deflection and explicit call transfer are limited to numbers explicitly supplied to the MExE executable by the MS user.
6	User Private Data access	Read/Write/Delete/Get Properties of User Data (e.g. user files, phone book stored on the MS); Read/Delete Stored SMS; Modify User Preferences.	User preferences cannot be activated or deactivated, only modified. The MS User will have the ability to define what preferences are accessible from each MExE Security Domain, with the default being No Access.

7	MexE Security Functions access	Install/Uninstall/Replace Certificate for a Domain; Encryption/Signature Creation/Signature Verification/ Hash/Non-Repudiation APIs.	A certificate can only be administered (installed/uninstalled/replaced) by the certificate issuer or the entity that owns the public key within the certificate.
8	Application access	Get Application List; Launch Application; Get Application Status; Stop/Suspend/Resume Application.	Only the MS user and the entity that launches an application has the privilege to stop, suspend or resume it.
9	Lifecycle Management	Install/Uninstall a MExE Executable.	-
10	Terminal Data access	Get Manufacturer Software Version; Read Time and Date.	-
11	Peripheral access	Activate/Access/Configure Smart Card/SIM Card, Sound, Speaker Volume, Serial Ports, Parallel Ports.	-
12	Input/Output User Interface access	Access to Input Devices (Keyboard, Mouse, etc), Output Devices (Display), Output Notification (Light, Sound, etc).	-

3.2.2.2 Enforcing domain privileges

3.2.2.2.1 Classmark 1 devices

Due to the resource constraints of Classmark 1 devices, a signed content trust model will not be supported. This means that the MExE security domains cannot be implemented. Instead, all executables downloaded into the MS will be treated as ‘untrusted’ and will run within a sandbox configured with privileges providing limited access to the MS user interface and persistent data. With user permission, untrusted executables may also initiate voice and data connections, generate DTMF and add phonebook entries.

A class loader ensures that executables run in their own name space and, in general, do not interfere with other executables running within the MExE device. A security manager implements the sandbox model by enforcing the access control policy outlined above.

3.2.2.2.2 Classmark 2 Devices

MExE Classmark 2 devices are expected to have sufficient storage and processing capacity to support the signed content trust model. This model can be combined with the Java sandbox model as follows. Each MExE domain will effectively be a sandbox configured with the privileges described in previous section. In addition, there will be a sandbox to confine the execution of untrusted applications, as described in the previous section for Classmark 1 devices. MExE executables downloaded to a device will be verified using the Operator, Manufacturer or Administrator root key. Depending on which verification key is used, the executable will be loaded into the run time environment (the device’s Java Virtual Machine) and set to execute within the sandbox associated with the key’s domain.

MExE Classmark 2 devices can optionally implement a more sophisticated version of the MExE domain model, using the concept of fine-grained access control introduced in Java 1.2. This new version of Java allows a security policy to be defined specifying who has access to what resources/privileges on the MExE device. The ‘who’ part of the policy is characterised by a digital certificate/public key identifying a specific signer, together with the location of the Java class (executable), represented as a URL.

So, for example, any executable signed with the certificate indicated, and which originates from the URL location indicated in the policy is assigned the corresponding execution privileges as set out in the policy.

In relation to the three MExE domains, the privileges associated with each act as an upper bound on the privileges that any executable belonging to the relevant domain can have. So for example, as a default, an executable signed with a TTP root key (i.e. a TTP application) would have access to all service categories apart from Device Core Functions, SIM Smart Card Low Level functions, Network Security, core software download and Network Property functions.

As a further example, suppose we have a TTP root key 'Entertainment' specifically for signing games. The games are downloaded from two (fictitious) web sites, www.adventure.com/games and www.playstation.com/games. These games do not need to perform cryptographic functions such as signing, encrypting and hashing, so we wish to disable their access to MExE security functions. This can be done by adding an entry to the security policy, stating 'Entertainment' as the signer and the two web sites as the locations of the MExE executables. Alongside, we can list all service categories to which access is allowed, i.e., all categories apart from Device Core Functions, SIM Smart Card Low Level functions, Network Security and Network Property functions, and MExE security functions.

Within the Java run-time environment, executables are loaded into individual Java security domains called 'protection domains' using a secure class loader. Executables that signed by the same key and downloaded from the same (URL) location are placed in the same protection domain. All instances of Java classes invoked within the protection domain are granted the same permissions. Privileges accessible from the protection domain are set out in the security policy and enforced by a run-time component called the Access Controller.

Continuing our example, suppose two TTP applications are downloaded, game X from www.adventure.com/games and game Y from www.playstation.com/games/. Both are signed by the entity 'Entertainment'. Following verification of X and Y, they will be placed in different Java protection domains, each with access to all categories of actions apart from Device Core Functions, SIM Smart Card Low Level functions, Network Security and Network Property functions, and MExE security functions.

Less restrictive policy entries are also allowed that do not require applications to be signed, or do not specify a location, allowing applications to be downloaded to the device regardless of their origin. Alternatively, if a security policy file does not exist or is in the wrong format, all executables are loaded into a default 'untrusted' sandbox.

3.2.2.2.3 Classmark 3 devices

Classmark 3 devices are expected to have very tight resource constraints. They are based on J2ME CLDC (Connected Limited Device Configuration) and MIDP (Mobile Information Device Profile). The J2ME CLDC and MIDP platform is aimed to provide a complete end-to-end solution for secure delivery of dynamic content for the extensible next-generation devices.

In order to apply to various types of devices, J2ME defines in CLDC a minimum platform that provides virtual machine features, and minimum libraries, that must be available on all devices of this class.

CLDC provides the high level libraries without considering any specific device categories. They provide a framework to support input/output and networking in a generic and extensible manner.

MIDP is based on CLDC, and can be viewed as an extension of certain of its features. For instance, MIDP provides support for a subset of the HTTP protocol, which lacks in CLDC.

From [1], it is not clear whether Classmark 3 devices will be able to support the signed content model. However, they will implement a Java sandbox for differentiating between trusted and untrusted applications.

3.2.2.3 User permissions

Regardless of the security domain, MExE executables must obtain 'User Permission' for an action before it can be performed. For example, this might take the form of a system permission request screen that displays the name of the MExE executable, its status (Security Domain or Untrusted status), name of the entity signing the executable, together with the actions for which user permission is required. The MS user would then select an 'OK' or 'Accept' button on the device to indicate that the required permissions are granted.

Clearly, it might be inconvenient and intrusive to require that the user grants permissions every time an action is performed by a MExE executable. Therefore, the scope of a user permission can vary from a single action to unlimited instances of the same action. More specifically, there are three types of user permission:

- *Blanket* – Once a user grants permission for an action performed by a particular MExE executable, the executable retains this permission until it is explicitly revoked by the user or the executable is uninstalled.
- *Session* – Once a user grants permission for an action performed by a particular MExE executable during a particular run-time session, the executable retains this permission for the duration of the session. Alternatively, the permission can be explicitly revoked by the user at any time.
- *Single Action* – User grants permission for a particular MExE executable to perform a single instance of the action. If the executable wishes to repeat the action, then it must explicitly request permission again from the user. The permission can be explicitly revoked by the user at any time.

3.2.2.4 Signature Verification

To support a signed content trust model, the MS must have the ability to verify the signed MExE executables transferred from the MExE Service Environment. MExE specifies the following requirements to support the verification process.

- Root verification keys must be securely installed in the MS. A MExE MS cannot verify the executables for a particular security domain unless it has a verification key corresponding to that domain. Verification keys are authenticated using digital certificates that can be marked 'trusted' or 'untrusted'. Only keys contained within certificates that have a 'trusted' status can result in successful verification of an executable.
- An additional 'disaster recovery' root public key should be securely installed in the MS and used to install new root verification keys when all other such keys on the MS are considered invalid. This is a recommendation rather than a mandatory requirement. A separate disaster recovery key is required for each security domain (that is, potentially there will be three such keys installed in the MS).
- Certification chains will be supported, with at least one level of certificate supported below the root verification key (i.e. a certificate chain of length of at least 1 certificate).
- Root verification keys cannot be shared between security domains.

The requirements for root verification keys associated with the Operator, Manufacturer and Third Party domains are examined in more detail below.

- Operator Root Verification Key – At least one such key is stored in the MS. If multiple keys exist, then only one key can be valid for signature verification at any one time, and is used to verify MExE executables intended to operate in the Network Operator Security Domain. The MS user cannot add or delete the operator root verification key or any key obtained from a certificate chain derived from the root verification key. A further requirement is that the Operator Security Domain 'belongs' to the same network operator that issued the SIM card within the MS i.e. the root verification key for the Operator domain is issued by the same operator that issued the SIM card.
- Manufacturer Root Verification Key – At least one such key is stored in the MS. If multiple keys exist, then only one key can be valid for signature verification at any one time, and is used to verify MExE executables intended to operate in the MS Manufacturer Security

Domain. As for the Operator domain, the MS user cannot add or delete root verification keys relating to this domain, or keys whose trust is derived from the root verification key.

- Third Party Root Verification Key – At least one such key is stored in the MS for verification of all other MExE executables, with no defined upper limit on the number of keys. The MS user is able to manage (add, remove or replace) this type of verification key if the user is designated the 'Administrator' for the MS (see Administrator Root Verification Key description). Certificates for Third Party Verification Keys can be issued by Network Operators and Manufacturers, using their respective root keys. For example, the MS Manufacturer may install a default set of certificates containing Third Party root keys, within the MS. Additionally, a third type of verification key, an 'Administrator' key can be used to generate the certificates.
- Administrator Root Verification Key – This type of key is subject to similar constraints as the Operator and MS Manufacturer keys: only one valid Administrator verification key shall be in use within the MS, and the MS user cannot add or remove such keys. The MExE specification (optionally) allows the Administrator key to be the same as the Operator root verification key, or the Manufacturer root verification key.

Aside from signing Third Party verification keys, the Administrator has a second special function: it is used to sign Certificate Configuration Messages (CCMs) – these are specially formatted messages downloaded into the MS as part of the TTP root public key management process. An Administrator key may or may not be present on the MS depending on who is assigned the Administrator role. If the Administrator is the MS user, then an Administrator key is not required. If the Administrator is a remote entity then such a key is required in order to securely download CCMs. The Administrator key, if one is required, need not be installed in the MS at manufacture stage. A mechanism is described within the MExE specification for remote download of the key to the MS. A remote Administrator does not have the right to delete TTP certificates – only the MS user has the ability to do this. Nevertheless, the administrator can disable TTP root using the CCM.

MExE does not specify the format for digital certificates. This is because MExE is at present unable to agree on a standard, rather than because of a lack of desire to standardise. It is envisaged that a variety of standard formats, ranging from X.509 to WAP WTLS could be used. However, certain security relevant certificate extensions are currently defined within MExE that apply to X.509 only, such as the 'Access-Restriction' extension. If the certificate of a key used to verify a MExE executable, or any certificate in a certificate chain used to verify an executable, has the 'Access-Restriction' extension, the executable in question will not be allowed access to the network services category of actions described in section 3.2.2.1. This is a mandatory access control mechanism in the sense that the executable will be denied access to network services regardless of the user permissions granted on the executable.

3.2.2.5 Certificate configuration messages (CCMs)

A CCM is essentially a simplified certificate revocation list used by a remote Administrator to maintain the list of valid (trusted) Third Party root verification keys on the MS. They are not used if the Administrator is the MS user. The CCM can be used only to enable (make 'trusted') or disable (make 'untrusted') verification keys, but not to remove them from the MS. Only the MS user can delete certificates from the device.

The contents of a CCM are outlined below:

- *Version* – MExE version. This determines the CCM format.
- *Certificate Advice* – Specifies an action applying to all certificates referenced in the CCM. This is one of {Enabled-All, Disable-All, Enable Present List only, Enable-List, Disable-List}. See below for explanations of each parameter.
- *Issue and Expiry Timestamps* – Specifies the time period during which the CCM is valid. This is used to detect 'replay' attacks. Such attacks involve a previously valid CCM being re-sent and accepted by a victim MS in an attempt to change security settings on the MS.

- S*igner Info* – Indicates the type of signer issuing the CCM. In the initial version, the signer will always be the Administrator.
- *List Length* – Total number of certificate 'fingerprints' in the CCM. A fingerprint is derived from the hash value of the certificate, computed using hash algorithms MD5 or SHA-1, and is used instead of a full certificate.
- *Hash Type and Hash Length* values for each certificate fingerprint contained in the CCM.

A global action, applying to all certificates in the CCM is specified in the 'Certificate Advice' field of the message. This action can be one of:

- *Enable-All* – All TTP root certificates on the MS are enabled, regardless of their status (trusted/untrusted).
- *Disable-All* – All TTP root certificates on the MS are disabled, regardless of their status (trusted/untrusted).
- *Enable Present List only* – All TTP root certificates currently on the MS are enabled. Any TTP root certificates subsequently loaded onto the device should not be enabled until there is a change of Administrator. The 'Enabled Present List only' option can be used to freeze the set of TTP roots for a certain period of time.
- *Enable List Enabled* – If a certificate's fingerprint appears in the CCM, the certificate should be enabled, otherwise it should be disabled.
- *Disable List Disabled* – If a certificate's fingerprint appears in the CCM, the certificate should be disabled, otherwise it should be enabled.

3.2.2.6 Download of executables and certificates

No secure data download method is specified for Classmark 1 devices, however, it is likely that any methods will draw upon the WAP security mechanisms (e.g. WTLS and Cryptographic APIs). For Classmark 2, secure transportation of MExE executables, certificates and CCMs is provided by the use of Java Archive (JAR) files to create signed packages for transmission and download to the mobile device. Signed packages will comprise a label indicating the content type (e.g. MExE native library, TTP/Operator/ Manufacturer/ Administrator Certificate, CCM), the actual content, and a signature sealing both items.

The download methods described previously require that at least one root key already resides in the device, that can be used to verify and enable additional verification keys downloaded over the air. When this is not the case, MExE defines a semi-automated procedure for downloading a certificate containing the Administrator Root Verification Key into the MS:

1. User and Administrator make contact.
2. Administrator Service Centre obtains any required information from the user and informs the user of the location of the administrator root certificate by SMS or other means.
3. User downloads the Administrator root certificate (e.g. as Java Signed Packages) to the MS.
4. On completion of download, the MS computes the hash value for the certificate.

User contacts the Administrator to obtain the hash value (at least the first 8 bytes of the hash value as computed by the Administrator), and compares the hash value computed by the MS. If there are no discrepancies, the certificate is accepted, otherwise the User must contact the Administrator.

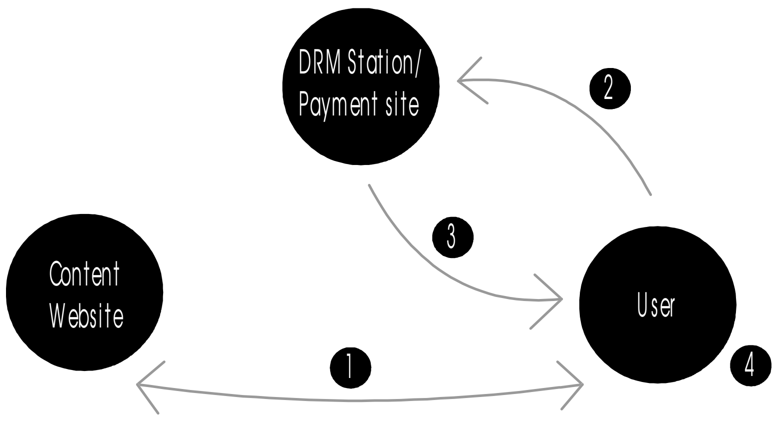
3.3 DRM

Digital Rights Management (DRM) is a term used to describe technologies designed to enforce copyright protection by preventing the illegal use of content. It allows distributors of electronic content to control access to that content. This can also restrict printing rights to a document, viewing or hearing a file a limited number of times, re-distribution of files, etc. DRM is typically carried out using encryption and authentication techniques, together with additional mechanisms such as watermarking or fingerprinting. Often individual encryption keys are applied to enable only the entitled person to use the content. A good example for this is the scrambling of Pay-TV-programs that can only be unscrambled if the user possesses the right decryption key. A DRM scheme will involve the verification of a 'licence' and an associated policy is carried out to authenticate that the digital content it has come from the required licensing body and not an untrusted party.

In many scenarios the DRM software will reside in the user's workstation, so that when he downloads the content the DRM software will check the user's license. If the license is invalid or if a license does not exist, it will then contact a financial clearinghouse for payment. A key is then assigned and the file can be decrypted for use. The publisher of the content can configure access in various ways, e.g. restrict the printing rights, the number of times a music file is played, forwarding rights, etc.

Various proprietary methods have been used to implement a full DRM solution; examples of open source solutions include SDMI (Secure Digital Music Initiative) [81] and IPMP (Intellectual Property management and Protection Standard within MPEG) [55]. They are all trying to achieve the same objective, namely to protect the intellectual rights of content and prevent forward distribution of content without a revenue intake.

The following figure outlines a basic DRM scenario:



1. User goes to a website and downloads a music file.
2. User tries to play the file, but does not have the decryption key or policy file related to the music file and is re-directed to a payment website.
3. User pays for the content and receives a decryption key for the content along with a policy file, which defines the user's rights to that content.
4. The user decrypts the content and is able to use the content according to the policy. For example, the user may have paid to play a specific music file five times. Intending to play the music file a sixth time, the user is redirected to the payment website.

In general, a DRM system has two main components [55]:

- for the *identification* of copyrights on the content,
- for the *protection* of the content.

3.3.1 Identification of Copyrights

According to [55], for identification purposes, a copyright descriptor should be attached to the content. This descriptor consists of two parts: a unique 32-bit identifier that points to a registration authority, and a copyright number given by that authority, and follows its own syntax. In [81], the proposed format (called SDMI file format) is as follows: a file consists of several blocks, each having a short header followed by the data. Certain blocks within each SDMI file have identification purposes. For instance, the first block (called the SDMI File Block) is used to the file as an SDMI music file. The *Intellectual Property Identification Block* contains information about the content.

The *Licence Block* contains all the information about the terms and conditions of use that apply to the file. As an example, the License Block may contain entries of the form:

- This SDMI file may be played until Aug. 8, 2001;
- This SDMI file is free; etc.

Copyright information should remain attached to the relevant content according to existing international laws. However, it is not seen as an objective of ISO/IEC JTC1/SC29/WG11 to enforce this information to be present or correct.

3.3.2 Protection of the Content

The protection of the content has two aspects to it. The licensed user must be authenticated, and the content must be protected both in transit and in storage.

One way to authenticate the user is by means of a PKI. Thus the user authenticates himself to the distributor using a digital certificate.

One way to protect the content is by use of symmetric encryption. Then the issue of key establishment arises. Presumably, the distribution authority generates a secret key, and uses it to encrypt the content. Subsequently, the secret key may be sent to the licensed user over an insecure channel, encrypted with the users public key (which was obtained from the digital certificate of the user).

3.4 SDR

As defined by the SDR Forum (http://www.sdrforum.org), the term *Software Defined Radio* (SDR) is used to describe radios that provide software control of a variety of modulation techniques, wide-band or narrow-band operation, communications security functions (such as hopping), and waveform requirements of current and evolving standards over a broad frequency range. The frequency bands covered may still be constrained at the front-end requiring a switch in the antenna system.

3.4.1 Definition

SDR is an enabling technology applicable across a wide range of areas within the wireless industry that provides efficient and comparatively inexpensive solutions to several constraints posed in current systems. For example, SDR-enabled user devices and network equipment can be dynamically programmed in software to reconfigure their characteristics for better performance, richer feature sets, advanced new services that provide choices to the end-user and new revenue streams for the service provider. SDR is uniquely suited to address the common requirements for communications in the military, civil and commercial sectors.

The basic concept is based on the use of a simple hardware platform built using SDR to enable customers to modify both the network and the end-user device to perform different functions at different times. This allows:

- End-users to realise 'true' choices with 'pay as you go' features, device independence and a single piece of scalable hardware that is compatible at a global scale;
- Network operators to differentiate their service offerings without having to support a myriad number of hand-helds and to move to adjacent markets as well as offer new, tiered services to increase their revenue mix;
- Infrastructure suppliers to lower cost and insure themselves against price-erosion through concentrated efforts on a common hardware platform and reduced component count;
- Applications Developers to enhance value without concern regarding hardware type;
- Terminal providers to add features, patches and capabilities to devices for broader market participation.

3.4.2 Security issues

The ability to reconfigure mobile devices via software downloads immediately raises serious security issues, directly analogous to those underlying MExE. Mobile equipment owners may unwittingly load and execute code that damages their mobile equipment, potentially rendering it incapable of further operation. Indeed, if the software disables radio operation then it may be impossible to access further code which could be used to re-enable operation.

In addition users may download code which gives their terminals unauthorised capabilities (analogous to the existing and growing market in unauthorised software for modifying car engine management systems). For example, whilst the use of some parts of the radio spectrum may be unregulated, other parts of the spectrum, e.g. that reserved for legacy systems and/or military or emergency service use, will remain regulated for the foreseeable future. Unauthorised software could enable an SDR to operate in ways contravening these regulations, and could become so widespread that breaches of regulation become impossible to police. This could have very serious and damaging effects on existing radio-based communications infrastructures.

On the other hand, once the potential flexibility of SDR becomes widely known, and indeed devices are marketed and sold on the basis of their flexibility, users will expect to be able to download software in an open and relatively free way. This user expectation could conflict seriously with requirements both to protect the user against damaging their devices and protect other users against devices operating in malicious ways.

Thus there are two main security issues for downloaded software:

- To protect the origin and integrity of the software against accidental or deliberate corruption, and
- To provide an authorisation system which enables the SDR to make an automatic decision as to whether or not to accept the downloaded software (i.e. use it to reconfigure the SDR).

Both these issues can, and are likely to be, addressed through the use of public key cryptography and PKIs. We examine how this might be achieved, and what the main issues are, in the next section.

3.4.3 PKI issues

The addition of a digital signature to a piece of code can be used by the code's recipient to verify its correctness and origin. The public key necessary to verify the certificate can be obtained from a public key certificate, either sent with the signed code or retrieved from a repository by the code's recipient.

Two main issues remain:

- If the code is signed by a CA for which the SDR does not possess the necessary public key, then a certification path (or other mechanism) will need to be deployed to enable the SDR to obtain a verified copy of the public key necessary to verify the code. Whilst mechanisms to achieve this exist, it is not clear how appropriate these will be to the mobile environment.
- Once the code has been verified, the SDR must decide whether to accept the code on the basis of the following:
    - the identity of the CA which signed the code;
    - the policy identifier(s) in the certificate(s) which were verified in order to obtain the code signer's public key;
    - the policy statement built into the device by the manufacturer, together with any policy statements input by the mobile device's owner and/or user;
    - any information associated directly with the code, e.g. that is within the scope of the signature sent with the code (this might include details of the intended scope of use of the code).

  It is not clear whether policies exist which would enable complex decisions of this kind to be made automatically in a sensible way.

Finally, it is not clear whether a network operator will have any means of affecting the policies used by a mobile device to determine whether downloaded code is acceptable. If not, then this could have a damaging effect on operators if they are held responsible for damage to equipment inflicted by malicious code, and/or damage to network availability caused by devices operating in malicious ways.

3.5 Application layer use of PK in WAP

As well as the use of WTLS at the transport layer, a number of techniques using PK and PKI at the application layer have been specified in WAP. These are:

- Client generated digital signatures, using the WMLScript function, signText,
- Client verification of digital signatures,
- Client encryption.

3.5.1 Client generated digital signatures, *signText*

SignText is specified in [77]. It is a WMLScript (WMLScript is an adaptation of JavaScript) function that can be called by an origin server or any entity presenting WMLScript to the client, which causes the client to generate a digital signature on a hash of text which is passed along with the function call.

For example, the client might receive a function call:

> *SignText*("5000 IBM shares. Price 150p.", *other parameters*)

This would be presented to the client if its owner were being asked to confirm such a share purchase. A digital signature on the transaction clearly brings the potential for authentication, integrity and non-repudiation of the transaction.

In order to ensure that only the legitimate owner of a mobile client can authorise signatures, operation of signText requires input of a PIN. This is a PIN separate from the PIN used to enable the SIM, which is often disabled by users and operators.

It is anticipated that the client's private key used to generate the signature is stored in a hard ware implementation of the WAP Identity Module (WIM) on the same smartcard as the client's SIM or WIM. This, however, is not mandated by the function – the function caller specifies which key should be used to generate the signature in the function call, and the client uses the specified private key, whether it is on the WIM on the SIM ICC, on a WIM on a second smartcard, on a WIM based in the mobile terminal itself or a private key that is not in a WIM at all but just stored somewhere in the mobile terminal. The WIM is the WAP Forum's specification for a hardware resistant module used for storing and processing sensitive cryptographic parameters such as private keys and WTLS master secrets. It is specified in [76].

Clearly the client's certificate is required to verify client signatures and there are options in signText to cause the client to return its certificate along with the signature. Alternatively, the client can return a URL for its certificate, in order to save bandwidth and if (to save storage space on the client) the client does not possess its own certificate.

Following guidance in [28], the client's signature contains, mandatorily, a hash of the data "to be signed" passed and the MIME type of the data being signed. If the client possesses a time source, it should include the UTC time in the signature also, if not, the client encloses a random nonce, in order to give some protection against signature replay by an attacker. The client's certificate is not part of the signature, and this makes the format of signText non-compliant with the ETSI format for legal/advanced electronic signatures given in [9].

The signature generated by the client is not completely in S/MIME compliant format. However, the signature is performed on a MIME-encoded object formed by the client and the signature can therefore be converted into S/MIME SignedData format for processing by S/MIME compliant software. This is significant, as there is a large existing base of products that can verify S/MIME compliant signatures. In an interesting move, in order to remove the need for the client to support an ASN.1 encoder for the

MIME object signed by the client, an ASN.1 template is given in the specification. This template can be stored in the client and when signText is called, the data hash and UTC time or random nonce are inserted into gaps in the template and the whole string signed.

3.5.2 Client verification of digital signatures, *Signed Content*

The signed content specification [74], specifies a generic framework for verification of digital signatures on content by WAP clients. The framework could be used to download signed executables, for instance, to a mobile phone. The specification is in draft status at the time of writing.

[74] specifies the format that signed content must be in. S/MIME [28] is used for this.

The specification is written to allow different types of signed content to be downloaded and for there to be different authorisation rules for each type of content. [74] does not assume that just because a signature can be verified using certificates on the terminal that the content is authorised to be executed– the signer of the content ("the signer") must be authorised to download content of the type that is being downloaded. "Execute" here is used loosely; if the content is simply data passed with authenticity and integrity, then "execute" in this case could just mean "display".

This authorisation is indicated in the signer's certificate. A number of options are given for deciding whether this authorisation is correctly within the signer's certificate. These are:

- The signer's certificate's root is authorised to download the relevant content type.

  This can be done by the presence of an OID for the content type to the PKCS15 structure in which the root is stored on the terminal or WIM.

  Within this method there are two ways for indicating whether an end entity (EE) certificate chaining up to a root authorised for a particular content type is authorised. These are:

    - There must be an explicit indication within the EE certificate. For example, it could be specified that for X.509 certificates, the Extended Key Usage extension containing the OID for *id-kp-codeSigning* must be present.
    - The root can be marked to indicate that all certificates that chain up to that root are authorised to download content.

- The signer's certificate is directly authorised by the presence of an identifier for the certificate on the terminal or WIM. This list of identifiers must be "securely" stored and updated.
- The terminal owner has chosen to directly trust the content. This does not mean that the signer is trusted for future content.

For each content type, one or more of the above methods must be specified (with parameters as appropriate) in order for the terminal to decide if a particular signer is authorised to download content of that type. In essence the "trust model" for each content type must be specified.

3.5.3 Client public key encryption

There are two, draft, methods for application layer public key encryption. One, *encryptText*, is used for encryption of single items of data and is envisaged for encryption of PINs, passwords and other short, sensitive data. The other, *initContext*, method uses public key techniques to generate shared symmetric keys that can be used a number of times for application layer symmetric encryption.

3.5.3.1 encryptText

A WMLScript function, *encryptText* can be sent to a client. The function contains "the data" to be encrypted (clearly there is no point asking the client to encrypt data sent to the client. Therefore it is likely that this argument in the function will generate (e.g. by a request for the user to enter some input, e.g. a password) the data to be encrypted). The function also contains a certificate for the public key with which the content will be encrypted. Actually, the client generates a symmetric key, encrypts the content with this, and encrypts the symmetric key with the public key received in the function call.

This is the method used in S/MIME and the content is formatted in a form that can be converted to S/MIME *EnvelopedData*. A user prompt can also be sent in the function call.

There is a second version of the function, *encrypt*. This is designed for encryption on the client where there will be no user involvement. This would be the case if the WAP client were an unmanned terminal, or if the function were part of an application that repeatedly and automatically sent information to a server, e.g. the location of the client. The function is the same except there is no user prompt, and the public key can be sent just as a public key, and not in a certificate.

3.5.3.2 InitContext

The WMLScript function, *initContext*, would be used where a number of items of data were to be encrypted at the application layer over a short space of time, and repeated call to *encryptText* would be inefficient.

The *initContext* function triggers the start of a process of server authentication, optional client authentication and generation of the shared symmetric keys. If this involves a three-pass protocol, a second WMLScript “function”, *initContextFinal* is used to complete the protocol.

After these symmetric keys are generated, they can be used for application layer encryption and integrity protection using separate, as yet undefined, functions.

The symmetric keys should not, for security reasons, be used long term. Function *closeContext* is used to direct the client to destroy the keys and their context.

4 Features provided by current and emerging public key techniques

This section will describe the security features (confidentiality, authentication, etc.) and "feature supports" (key management, key status) provided by public key (PK) techniques and examine how well the features are provided.

The intention is that by comparing the features required by WP1 and WP2, as listed in sections 5 and 6, we can determine, in section 9, whether existing PK techniques will meet the requirements of WP1 and WP2 or not, and suggest methods to extend PK if it is not sufficient. Further, by reflecting on what existing PK techniques do well and do not do well, we can determine if we need to improve or adapt the way that PK is used.

The security features and feature supports that will be examined are:

- Authentication
- Integrity
- Confidentiality
- Authorisation
- Non-Repudiation
- Key Management
- Key Status

This examination is intended, for maximum applicability, to be algorithm independent, though where there are significant differences in the way that different PK algorithms provide a particular security feature, this will be stated.

4.1 Authentication

ISO/IEC 10181-2 [43] defines authentication as the provision of assurance of the claimed identity of an entity. We define the entity wishing to authenticate itself and provide this assurance as the *authenticating entity*, and the party seeking the assurance as the *assured entity*.

With PK, authentication is achieved primarily by the process of an entity proving it has possession of a private key. This proof of possession may be performed by the authenticating entity generating a digital signature that the assured entity can verify. Alternatively, the authenticating entity can perform the proof of possession using its private key to decrypt some information encrypted with the authenticating entity's public key and then proving possession of the decrypted information to the assured entity. This latter form is used for server authentication in WTLS and SSL. This latter form can only be used when the PK algorithm provides encryption, for example, as RSA does. The latter form can be used for key management too, as the encrypted data can be symmetric keying material.

It should be noted that the assured entity must be in possession of the public key of the authenticating entity. Where the transfer of the public key from the authenticating entity to the assured entity takes place over open systems, the assured entity must be given assurance that the public key has not been tampered with in transit, and that it really is the public key of the authenticating entity and not the public key of an intruder. Where the assurance that a particular public key belongs to the authenticating entity is performed using a public key certificate, the assurance (i.e. the validity period of the certificate) must be still valid, and the assured entity must have the root CA and perhaps intermediate CA certificates required to validate the certificate of the authenticating entity. Further, when certificates are used, some PKIs support the revocation of the certificate, that is, the provision of a message (that of course must itself be transmitted with integrity and authentication) declaring that the certificate should no longer be used as assurance that a public key belongs to a particular entity.

4.2 Integrity

[82] defines integrity as "The property of information that it has not been changed by unauthorised parties". We will define "integrity protection" as the security features that provides proof to an entity that data it receives has either preserved its integrity in transit or not.

We define the entity sending the integrity protected data as the *sending entity* and the entity receiving the integrity protected data as the *receiving entity*.

Integrity protection is provided using PK by the sending entity signing the data to be protected, or more commonly, a hash of this data, with its private key ,and sending the signature along with the data to be protected. The receiving entity, which must be in possession of the sending entity's public key, verifies the signature on the hash. Under the assumption that the sending entity is the only entity in possession of the sending entity's private key, if the signature is verified, the integrity of the data is proven.

Integrity is usually only provided using PK when there is a single item of data to be protected, or where proof of the integrity must be preserved. It is likely that for the download of applications, a topic of consideration in WP2, integrity protection would be provided using PK techniques. If there are a number of items of data that can be sent together, and where there is no need to preserve proof of integrity protection, an integrity protected *channel* would be used, such as that provided by TLS or IPSec, where the integrity protection is provided using secret key techniques (although the symmetric key is established using PK key management techniques).

4.3 Confidentiality

[82] defines confidentiality as "the property of information that it has not been disclosed to unauthorised parties". The sender and receiver of data protected by confidentiality are defined as the *sending entity* and *receiving entity* respectively.

Confidentiality is provided using PK (solely PK) by the sending entity encrypting the data with the receiving entity's public key. The receiving entity decrypts the data using its private key. Under the assumption that the receiving entity is the only entity in possession of the receiving entity's private key, the data is confidential and known only the sending and receiving entities.

Due to the relatively high computational overload, PK is only normally used for providing confidentiality where there are small amount of data, as is the case for key management. In most circumstances other than key management, confidentiality is provided using secret key techniques.

4.4 Authorisation

4.4.1 Discussion on the definition of 'authorisation'

We define authorisation as the granting to an entity of permission to perform a certain action or the granting of some privileged status to that entity. Authorisation takes place in communications networks by the sending of some message to an entity that indicates that the received entity is authorised according to the contents of the message.

Clearly only certain parties should be granting privileges and permissions. There is therefore a requirement for well behaved clients (clients that only wish to receive authorisation messages from entities that are entitled to issue such messages) to authenticate the source of the message, and to determine, using some sort of internal list, if the source is itself authorised to issue authorisation messages. We can consider this latter function as essentially an access control function.

As the authorisation messages may be transmitted across open networks, their integrity must be protected.

In some cases, there will be a need to provide confidentiality of authorisation messages, but we will assume that this is provided by means other than the means of providing the basic authorisation message.

Clearly there must be agreement between the originating and receiving entities of the format of authorisation messages, so that the receiving entity will know when an authorisation message has been received and what the authorisation message signifies.

We can therefore say that provision of authorisation messages to a receiving entity is provision to that entity of *integrity protected messages of a pre-defined format that the receiving entity can authenticate the origin of*, and determine from some form of internal access control list, whether the originating entity is allowed to issue the authorisation message received and therefore whether the message should be followed or not.

4.4.2 Methods of providing authorisation using PK

Clearly an integrity protected message of a defined format whose origin can be authenticated can be provided using PK with a digital signature by the originating entity on a message of a pre-defined format. This method has traditionally not been used, but is now being used along with XML to provide … (whatever I find out about SAML). Traditionally, authorisation messages have been passed either using X.509 (identity) certificates or using X.509 attribute certificates.

4.4.3 Authorisation with X.509 public key certificates

X.509 public key certificates (hereafter just referred to as “X509 certificates”) are described in Appendix A.2 of this document. X.509 certificates were designed to assist in the provision of authentication, by binding an identity to a public key. However, since the introduction of extensions in version 3, X.509 certificates (along with appropriate rules for certificate processing applications) have been used in effect for authorisation purposes. For certain extensions, the inclusion of the extension in the certificate of an entity implies that the entity is authorised to use a particular privilege when signing objects (which may or may not be other certificates). Some examples are now given.

The *basicConstraints* extension is used to indicate whether an entity is a CA or not, i.e. whether it is authorised to issue certificates itself, or whether it is an end entity (EE) whose certificate should be the lowest in a certificate chain.

The *keyUsage* and *extendedKeyUsage* extensions can be used to award privileges to the certificate holder related to the uses that the key may be put to. If the *cRLSign* bit within the *keyUsage* bit string is set, this means that the certified entity’s private key can be used to sign CRLs. The presence of the OID for *id-kp-codeSigning* within *extendedKeyUsage* means that the entity is authorised to sign executable code. This techniques is used within the draft WAP Signed Content specification as a possible way to indicate code signers that are authorised to download code to a WAP client.

Private extensions can also be defined and used within a particular user group. One widely used example is the Server Gated Cryptography extension that was used by Microsoft to indicate SSL servers that had been authorised to use 128 bit encryption in the SSL session.

4.4.4 Authorisation using attribute certificates

4.4.4.1 Introduction

Resource allocations and access control are two major issues that are at the heart of information security today. The increasing numbers of users requiring access to these resources have put traditional methods under increasing pressure. The complex systems architectures and relationships between different responsible parties have not helped the situation either. So far traditional digital certificates have been modified to accommodate the additional requirements. However this approach has a number of problems associated with it. Attribute certificates have been identified as a potential solution to some of these challenges. This document highlights the benefits and shortfalls in attribute certificates in meeting these challenges.

4.4.4.2 Background

When digital certificates were first invented the primary use for them was to identify a subject to a third party. It was left up to the third party how that result is used. For example, if it is to control access to a file on a database, then an ACL (Access Control List) is required to map the authenticated user to the corresponding access rights. This requires the owner of the resource to maintain an ACL. In addition this approach is restrictive since a user has to be pre-registered with the owner of each and every resource prior to using it. A user without an entry in an ACL for a particular resource will not be able to access it even if he or she is identified correctly.

Therefore it was realised that digital certificates on their own had limitations. For the majority of applications they need to carry more information than the identity of the subject concerned. As a result, extensions were added to the specification to carry additional information within the X.509v2 and v3 certificates. The X.509v3 certificate carries information about the identity of the user and also information about what the user can and can't do in a given environment. However it would not make sense to mandate all these extension within the specification but where reliance is placed on an extension, there must be some way to ensure that the extension is processed properly. Hence the notion of criticality was introduced. If an extension is present in a certificate and is marked as critical then if the client processing the server does not understand the extension, it must reject the whole certificate. This ensures that only those clients that understand the extension (which may, for instance, limit the validity of the certificate in some way) will process it.

This support has its share of limitation associated with it. For example this means that certificates are required to carry specific extensions that are applicable to a particular resource. A certificate with any other extension would simply be rejected. Also access rights in general tend to vary with time and this also means that certificates need to be re-issued whenever the user's rights on a given resource change. In addition, combining identification and authorisation means that the issuer of the digital certificate is required to have knowledge about the resource that the certificate is intended for. This is not practical since the certificate issuers in general would be highly trusted organisations and it would be silly to contact them every time one of the access rights to a file needs to be updated on a certificate extension.

4.4.4.3 General Features of Attribute Certificates

One way around this problem is to effectively split the contents of a X.509v3 certificate into two different parts. The first part carries information to identify the subject. This is exactly what digital certificates were originally designed to do. The second part on the other hand carries information about the public key certificate of the user and the authorisation given to that subject by the owner of the resource. This second certificate is known as an attribute certificate. The X509.v3 [58] specification together with a draft specification of attribute certificates [10] are available from the PKIX (PKI Expert) group in the IETF.

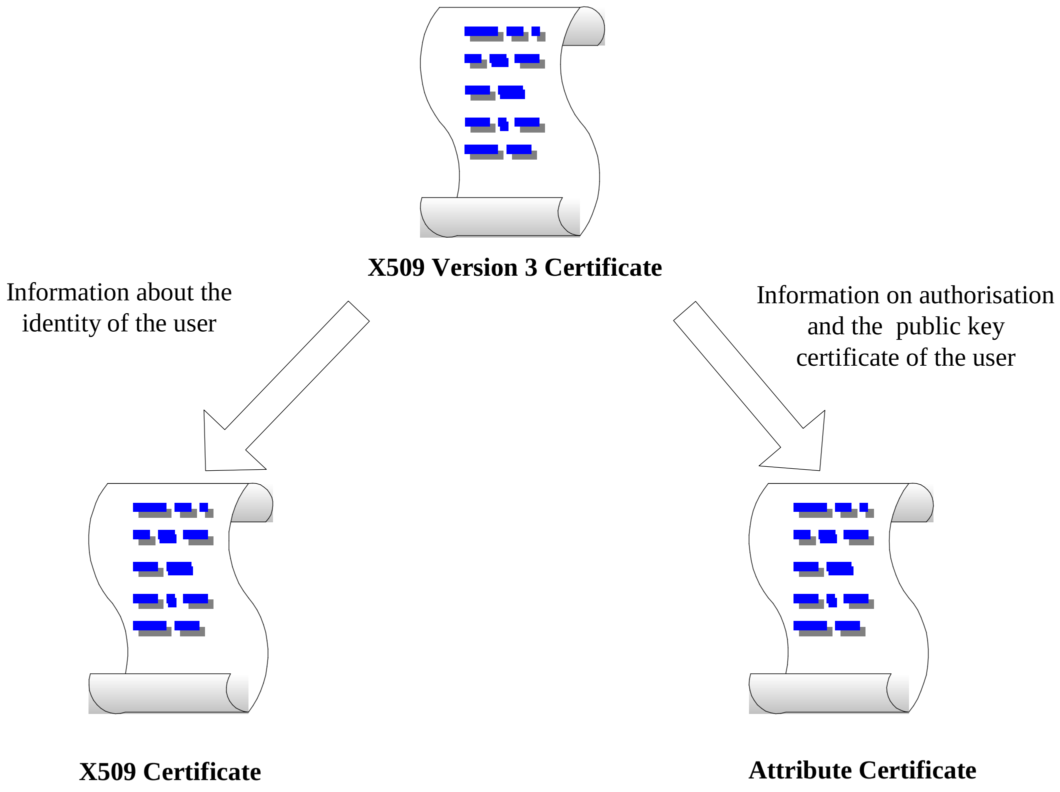

X509 Version 3 Certificate
Information about the identity of the user
Information on authorisation and the public key certificate of the user
X509 Certificate
Attribute Certificate

Figure 1 – An attribute certificate carries the information carried in the extensions of a X509v3 certificate

An attribute certificate on its own has no value. The corresponding public key certificate should also accompany it since the attribute certificate references the public key. This is important since it is the only way of proving that the attribute certificate does belong to the subject of the accompanying public key certificate. The attribute certificate itself is issued either by the owner of the resource or by a third party trusted by the owner of the resource. Therefore an attribute certificate in general will be tied to a particular resource. However it is possible to make attribute certificates applicable more widely if the owners of the resources co-operate with each other. That is if they agree some common terms on which they accept each other's certificates.

Attribute certificates also provides with an alternative solution to certificate revocation. By simply making them short lived, the problem of revocation could be addressed to a certain extend. However, the revocation of public key certificates should still be looked into separately.

4.4.4.4 How does an Attribute Certificate work?

First of all, the user needs to have a valid digital certificate. In addition a valid attribute certificate for the resource in question is required. This certificate should be issued by the owner of the resource or by a party trusted by the owner of the resource. When the user requires access to a resource, the attribute certificate together with the digital certificate is sent to the owner of the resource. The owner of the resource authenticates the user via the public key certificate. It then verifies that the referenced public key in the attribute certificate belongs to the user in question. On the successful completion of these the user is granted the permissions authorised by the attribute certificate.

It should be noted that there could be occasions when a chain of attribute certificates might have to be processed to verify the authority of the issuer itself. Similarly, a certificate chain might have to be processed when the public key certificate is verified as part of the attribute certificate verification.

4.4.4.5 Applications of Attribute Certificates

The concept behind an attribute certificates is still new to the industry. The flexibility in management together with the ability to deliver fine-grained authorisation means that attribute certificates would prove to be useful. In effect what attribute certificates are enabling is a flexible way of carrying authorisation information about a subject for a given resource. There is no restriction what that resource could be.

For example it could be a database with files, in which case the authorisation information in the attribute certificate is used for access control. At the same time, there is no limit on the granularity of the authorisation information carried. It could be at the individual file level or it could be at the folder level. The granularity obviously comes at a cost, the more fine-grained the authorisation is the larger the attribute certificate size.

Another possible application for attribute certificates is in access control for resources in an execution environment. This could be at different levels. For example it could be at the API level. In this scenario, an application will have access to a standard set of APIs; if however the application requires access to a non-standard or a restricted API, then the attribute certificate accompanying the application should have explicit authorisation to access the API concerned. At the same time, the attribute certificate could enforce access control at the hardware level. In this scenario, the certificate information could refer to a printer or access to a smartcard reader. Also, there is nothing stopping these two types of information being combined to a single attribute certificate. However, it would make sense to keep them separate since it gives the flexibility to change one without having to reissue the whole certificate.

Attribute certificates also provides non-repudiation. That is the issuer of an attribute certificate cannot deny it at a later time. This is particular important in protecting the owner of the resource if the user has misused it via an attribute certificate issued by the issuer in question.

Some users prefer to gain access to resources while remaining anonymous. In an identity based access control system this is simply not possible. Since attribute certificates carry explicit information about what the user is authorised there is no need to match the identity of the user to a local access control list. The attribute certificate itself relies on the accompanying public key certificate to prove the ownership of the attribute certificate to the user in question. However, there is nothing stopping the user from assuming a temporary identity from the attribute certificate issuer. In this way, the user’s true identity is revealed to the issuer of the attribute certificate but not to the owner of the resource. A similar concept is used within the Telecommunications Software Security Architecture (TeSSA) using Simple PKI (SPKI) certificates [72]. This is only possible through a third party issuer of certificates and not through the owner himself.

4.4.4.6 Attribute Certificates and Authorisation

It is clear from the above examples that attribute certificates are a way of carrying authorisation information in a secure manner. However it does not authenticate the user to the resource. The authorisation information itself could be delegated down a tree of attribute certificates. The entity at the top of the authorisation tree itself should be trusted directly by the owner of the resource.

4.4.4.7 Issues

The ease of introducing new attributes means that if all parties start defining their attributes without standardising them, there could be interoperability problems. Also, with the potential to have an attribute certificate for every possible resource used by a user, it could soon become a challenging task to manage all these different certificates, especially within a resource constrained device.

There is increased responsibility on the issuer (most probably a CA) of attribute certificates. The issuer is expected to have knowledge of the authorisations contained in a given attribute certificate. Also the issuer needs to maintain a set of criteria that is up to date for matching user requests to authorisations. For example, the issuer should know if a particular service is available on the resource concerned before authorising it in the attribute certificate. As with all PKI solutions if the issuer’s private key is compromised the system as whole will fail.

At the same time attribute certificates does not solve the revocation problem fully. In fact the challenges in revocation of public key certificates still remains. Since all attribute certificate systems are dependent on public key certificates this still is a challenge. At the same time making attribute certificates themselves short-lived does not solve the problem entirely. There could well be instances when a certificate could get compromised while the certificate validity lasting for a couple of hours. Depending on the resource and the application concerned there could be a real risk involved. At the same time, the more distributed the issuing of attribute certificates are, the harder it is for an individual issuer to reach and revoke certificates that could well be across many logical as well as geographical locations.

4.5 Key management

For the purposes of this document, key management is defined as the process of agreeing and/or transporting symmetric keys over public networks (this rather restricted definition is sometimes used for the term *key establishment* – see, for example, [44]). We treat key agreement and key transport separately.

Key transport using PK can be achieved by the simple encryption of the symmetric key with the public key of the entity to which is being sent the symmetric key. This method is used in SSL with RSA, where the client encrypts the client generated pre-master secret (from which all keying material for the session is derived) with the server's public key. The method is also specified in the WAP WMLScript function for application layer encryption, *encrypt(Text)*, in which the client encrypts the content with a client generated symmetric key and encrypts the symmetric key with the public key of the receiving entity.

Authentication is often performed during the same protocol or exchange in which the key transport occurs – it makes sense to authenticate the entity to which you are sending sensitive data. However, the authentication is not part of the key transport operation, so key transport can be considered alone. Key transport can therefore be seen just as an application of the use of PK for *confidentiality*. The comments and advantages and disadvantages for use of PK for confidentiality in section 4.3, therefore also apply to the use of PK for key transport.

Key transport can only be performed using PK when the PK algorithm supports encryption. RSA can therefore be used for key transport and is by far the most important algorithm for key transport (indeed for any use of public key cryptography). It is also possible to use ECC for key transport, through the ECES algorithm, but this is not widely done.

PK can be used for key agreement if the PK algorithm is a key agreement technique, e.g. Diffie-Hellman in either its traditional or ECC variants. As key agreement using Diffie-Hellman provides for confidentiality but not authentication (two entities can generate a shared secret key, but they have no confirmation of each other's identity) DH key agreement is often combined with authentication. This authentication is usually in the form of a certificate for the DH public key.

DH key agreement can be seen as the combination of two incomplete (one pass only) key transport operations (which are 'two pass'). In traditional key transport, entity A sends its public key to entity B, and B responds with a symmetric key encrypted with A's public key. In DH key agreement, entity A still sends its (DH) public key to B, and B responds with its (DH) public key, instead of an encrypted symmetric key. Both entities derive the shared symmetric key from the combination of the other's public key and their private key.

Therefore, the comments and advantages and disadvantages of using PK for confidentiality also apply, in that an entity can establish confidential communications with many other entities with which it has no existing trust relationship. The disadvantages of processing overload, and if, authentication is also required, reliance on certificates and PKI, also apply.

A further disadvantage to the use of PK key agreement for key management, as compared to the use of secret key techniques for key management is that with key agreement (as oppose to key transport) both entities require a DH key pair. This imposes on both entities the requirement of either securely storing the DH private key, if the key is long lived (a "Static" DH key) or of generating a DH key as it is required ("Ephemeral DH"), which can be a significant processing burden for a limited device.

4.6 Non repudiation

ISO/IEC 13888-1 [45] defines the goal of a *non-repudiation service* as to generate, collect, maintain, make available and validate evidence concerning a claimed event or action in or5der to resolve disputed about the occurrence or non-occurrence of the event or action. It goes on to define various types of non-repudiation, including *non-repudiation of origin*, which is intended to protect against the originator of a message falsely denying having sent it.

When non-repudiation is provided using PK techniques, as described in detail in ISO/IEC 13888-3 [46] it is provided by one entity generating a digital signature on the data signifying the operation (for example, a share transaction) that the entity will not be able to repudiate. There must be a key pair in existence for the entity performing the non-repudiable transaction, and this entity must securely store the private key used for providing the non-repudiation property. A valid copy of the entity's public key (and the signed transaction) must be in possession of any entity wishing to verify that the entity did carry out the transaction.

Non-repudiation is a property of digital signatures by virtue of the fact that digital signatures provide both integrity protection (the entity signed *this* data, because it can be proven that the data has not changed) and authentication (*only* the entity in question signed the data, and this can be proven). Therefore, the general comments on the use of PK that apply to its use for authentication and integrity also apply to non-repudiation.

Thus, in particular, the effective provision of non-repudiation using asymmetric (PK) cryptography rests on the degree to which the public key can be relied upon. For non-repudiation this means that it is not sufficient for public keys to be distributed by secure channels between users, since non-repudiable evidence is required of the validity of a public key. This is typically provided by the use of public key certificates, distributed by means of a PKI.

4.7 Status information

A certificate is revoked if the period of its validity has expired, or if the private key has been compromised or in danger of being compromised. Compromise, or risk of compromise, might arise for a variety of reasons, including cryptanalytic technology advancement or unauthorised access to the private key store. The issue of notifying the relevant parties of revocation can cause problems. If a certificate is revoked, it means that everyone who has a copy of that certificate should no longer use it, since the private key might be in the possession of an attacker. This means that every person wishing to use the public key must know about the key being revoked. Passing this revocation information to everyone who might wish to use the public key is the issue here.

Various ways have been identified to tackle this problem, and these are explained below.

4.7.1.1 Certificate Revocation Lists (CRLs)

The format of CRLs and certificate revocation was first defined by the ITU-T X.509 group in the late 1980s (see [56]). A profile of CRLs has been specified by the IETF in RFC 2459 [20], [21] as part of the PKIX development effort. Section five of RFC 2459 specifies the use and profile for a CRL.

4.7.1.1.1 Description

A CRL is a time-stamped list that gives information on revoked public key certificates. CRLs are signed by a CA and are made available by some means. Public key certificates are identified in the CRL by their serial numbers, and CRLs are updated 'off-line' at regular defined intervals. This means that they get updated every hour, day, week, etc. The updated CRLs are typically posted to a public server so that they can be retrieved by all certificate users.

When a public key certificate has been revoked, the revocation can only be notified in the next CRL update; this may be in the next hour, day or week depending on the implementation. Once the public key certificate is included in a CRL, the recipient of the certificate has the job of checking for revoked certificates. If Alice retrieves Bob's public key certificate, then Alice has to check against the latest revocation list before trusting the certificate. If the process it carried out periodically then Alice will have to retrieve the latest revocation list from its location and then check for revocation.

4.7.1.1.2 Features

A CRL is a signed list of revoked certificates. Public key certificates bind a public key with a user name. CRLs provide the user with information on revoked certificates, hence protecting the user against using an invalid public key.

A certificate gives the user a form of authentication as it binds personal information with a public key; certificates also provide authorisation and can be used to support non-repudiation, and thus the protection of these certificates is therefore very important. A CRL provides information for the current status of a certificate, providing revocation information to a recipient to show if the certificate is invalid or if it is acceptable to use. The recipient will rely on a CRL in order to carry out the functionality that a certificate supports, e.g. authentication, authorisation, and non-repudiation.

4.7.1.2 Online Certificate Status Protocol (OCSP)

The OCSP protocol work has been carried out by the IETF within the PKIX group. OCSP is an internet draft (draft-ieft-ocspv2-01) [26]. It specifies a protocol determining the current status of a digital certificate without the need for a CRL.

4.7.1.2.1 Description

OCSP tries to overcome some of the problems that CRLs have. The main problems are CRL latency and size. OCSP tries to overcome these problems by providing an online list of revoked certificates. This means that no latency occurs from when a certificate is revoked to the potential publication of the revoked certificate.

A typical OCSP session has a responder and recipient (of a request). The recipient sends a message to the responder enquiring about the status of a particular certificate. The responder can reply in various ways, but a mandatory basic response includes:

- The version of response syntax,
- Name of the responder,
- Identification of response type,
- Responses for each of the certificates in a request,
- Optional extensions,
- A signature algorithm OID and a signature computed across a hash of the response.

The response of each certificate in the request holds:

- Identification of target certificate certificates status value,
- Response validity interval and optional extensions.

The status values of the certificate are:

- Good,
- Revoked
- Unknown

The good response indicates that the certificate (to the knowledge of the OCSP responder) has not been revoked; however this does not mean that the certificate was ever issued, or the time that the response was produced is within the certificate’s validity date. The revoked state indicates that the certificate has been revoked and the unknown state is sent when the responder does not know about the certificate in question.

All OCSP responses will be digitally signed if it is an authorised responder (a responder that has been given permission or authority by the root CA). The key for the signing must belong to either a CA who issued the requested certificate, a responder who is trusted by the recipient or a CA designated responder who holds a specially marked certificate (indicating the responder may issue OCSP responses) issued directly by the CA who issued the requested certificate.

4.7.1.2.2 Features

OCSP provides an online CRL to clients. The entry revocation to the list is carried out upon notification, providing an up to date database at all times. The OCSP has to authenticate itself by signing its responses, a certificate chain is sent along with the signature.

A certificate gives the user a form of authentication, as it binds personal information with a public key; certificates also provide authorisation and can be used to support non-repudiation, and thus the protection of certificates is clearly important. A OSCP server provides information for the current status of a certificate, providing revocation information if the certificate is invalid, or if it is acceptable to use. It also provides additional information such as certificate validation and path discovery if Delegated Path Validation and Delegated Path Discovery are ‘plugged in’ to use with the server. The recipient will rely on a OCSP server in order to carry out the functionality that a certificate offers i.e. authentication, authorisation and non-repudiation.

4.7.1.3 Delegated Path Validation (DVP) for OCSP

DVP is work that has been carried by the IETF. DVP is an Internet draft (draft-ietf-pkix-ocsp-valid-00). DVP builds on the OCSP framework’s extensibility by defining an Internet-standard extension to OCSP that can be used to fully delegate all path validation processing to an OCSP server.

4.7.1.3.1 Description

Transferring certificate chain validation [26] to the OCSP server reduces the processing burden on the client, especially in the wireless environment. The entire certificate chain gets verified on the server rather than just the public key certificate. The chain is not returned; instead the status of the chain is returned. If the status is good, then the client can then use the public key.

4.7.1.3.2 Features

The OCSP-DVP server will do all the verification work for the client. Once the work is carried out, the server will send a verified OK or false response. This is a useful feature as certificate chains can get fairly long, and each certificate to be verified may have its own chain.

4.7.1.4 Delegated Path Discovery (DPD) for OCSP

Like DVP, DPD work has been carried out by the IETF and also works with OCSP. DPD is an Internet draft (draft-ietf-pkix-ocsp-path-00). The specification establishes an Internet extension to OCSP to address the need to acquire certification path data from an OCSP server rather than replicate the same functionality for each certificate.

4.7.1.4.1 Description

The OCSP-DPD server accumulates the set of certificates from which certificate chains may be constructed; this will include certificate revocation data for each certificate. If the recipient were to acquire this status information for himself, then he would have to visit different sources and it is most likely that each source will have its own retrieval policy e.g. HTTP, FTP, etc. DPD centralises all validation and will provide a method of download that the client supports.

4.7.1.4.2 Features

DPD will be used in cases where the certificate chain is not passed down to a client. This means the client will not have to ‘hunt’ the chain for itself. Once the chain is discovered the server will pass it back down to the client, the client then will make a DVP call for the verification process of the chain. (See OCSP).

4.7.1.5 Simple Certificate Validation Protocol (SCVP)

The SCVP protocol work has been carried out by the IETF within the PKIX group. SCVP is an internet draft (draft-ieft-pkix-scvp-0a) [35]. The SCVP protocol allows the client to offload certificate handling to a server.

4.7.1.5.1 Description

SCVP can transfer all certificate processing work to an SCVP server, reducing the processing burden on the client. However this is dependent on the client, as the client can choose which services he wants from the SCVP server. The protocol attempts to allow the centralisation of administration of PKI policies. SCVP servers can be untrusted resources for client just for data collection or other services that will not require security, or they can act as trusted servers for validation of certificate information. Trusted servers can pass all revocation information to clients, and the server will send the certificate chain needed for path validation to authenticate itself.

SCVP offers a complete validation procedure rather than just information on certificate revocation. The client can specify what types of check he wants on a requested certificate. The SCVP server will hold CRLs and will have access to OCSP responses. The client thus only needs one point of access for all certificate revocation requests. Each response from the server is signed and the client must check the signature it receives. The client must have access to the server’s public key and must know and trust the root that it chains to.

4.7.1.5.2 Features

SCVP provides more than just a certificate validation process. It concentrates on passing some or all PKI-related processing on to the server. This makes client software more lightweight, which is especially desirable in a wireless environment.

A certificate gives the user a form of authentication, as it binds personal information with a public key; certificates also provide authorisation and can be used for non-repudiation issues, and thus the protection of these certificates is therefore very important. A SCVP server provides information regarding the current status of a certificate, providing revocation information to a recipient to if the certificate is invalid or if it is okay to use. Additional information is provided SCVP server depending on the recipient request. SCVP handles a PKI rather than just the revocation information. The recipient will rely on a SCVP server in order to carry out the functionality that a certificate offers i.e. authentication, authorisation and non-repudiation.

4.8 Timestamping services

The use of data stored on easily modifiable media raises the issue of how to verify when these data were created. Digital time stamping can help establish the timeliness of data, where:

- A time variant parameter must be tied to the data in a non-forgeable way to avoid repudiation of the data’s existence prior to a certain point in time,
- Data must be provided in a way that preserves its confidentiality.

One very important application of a time-stamping service is to provide long term value for digital signatures. A time-stamp can give evidence that a digital signature was generated prior to the date of revocation of the key used to sign it. This prevents the revocation of a key invalidating all signatures generated prior to the date of revocation.

4.8.1 Standards development

A three part standard on Time-stamping services (ISO/IEC 18014 parts 1-3) is being developed by ISO/IEC JTC1/SC27. In parallel to the ISO/IEC work on time-stamping services, the IETF have also been developing a draft on time-stamping services [36]. The three parts of the ISO/IEC standard are as follows:

- *Part 1: Framework* [52], defines formats of time-stamp request and time-stamp response messages,
- *Part 2: Mechanisms producing independent tokens* [53], i.e. tokens for which verification of a token is independent of the verification of all other tokens,
- *Part 3: Mechanisms producing linked tokens* [54], i.e. tokens for which verification of tokens is linked, to strengthen the evidence provided by time-stamp tokens.

The (draft) standard specifies a protocol for time-stamping the hash value of data, allowing for integrity and confidentiality protection. The hash of the data is time stamped and signed by the Time Stamping Authority (TSA). The signature guarantees the integrity and authenticity of the time stamp. A time certificate providing these elements will be sent to the requester of the time stamp.

The draft standard provides ASN.1 specifications for the messages used in the supported types of transaction. In particular, data structures are defined for the following three types of message:

- Time stamp request,
- Time stamp reply,
- Time stamp verification.

4.8.2 The standardised timestamping protocol

The following entities may be involved when a time-stamp is requested:

- An *entity* possessing data to be time stamped acts as the *requester* of a time stamp. An entity obtaining proof that the time-stamped data received has a valid time stamp is the *verifier* of a time stamp.
- A *time-stamping authority* (TSA) offers a time-stamping service. The TSA offers evidence that data existed at a certain point in time and guarantees the correctness of the time parameter.
- A *temporal data authority* (TDA) gives additional corroborating evidence of a point in time contained in a time stamp. It creates a temporal data token associating a message with a particular event that occurred prior to that point in time, and thus gives supplementary evidence for the time included in a time stamp. An example is the actual closing value of the Dow Jones Average.

All the entities introduced communicate in a two-way handshake protocol. A user sends a request to the TSA and gets a time stamp. If further evidence is needed, or if the user asks for additional data, the TSA requests these data in the same way from the TDA, e.g. via a two-way handshake protocol.

4.8.3 Extending timestamps

Time-stamped data may be re-stamped at a later time. This may be necessary for one of the following reasons:

- the TSA’s signature key will expire soon;
- the TSA may soon be replaced by another TSA.

Time-stamped data may thus get a time-stamp renewal prior to the end of the TSA key’s validity. This extends the validity period of an existing time-stamp.

Data may also be re-time-stamped after the TSA’s key has expired or is revoked. This is then a new time-stamp that has no connection to former time-stamps associated with the data, and the service is called time-stamp re-issue.

4.8.4 Verifying timestamps

The validity of a time stamp may be verified in three different ways:

- An entity checking the Time Stamp Token first verifies the TSA's certificate. If necessary, this includes verifying the TSA's certification path.
- Another TTP verifies that the TSA has provided a time stamp. This is appropriate when the issuing TSA does not belong to the entity's security domain.
- The client or a TTP verify whether the TSA has kept a record of the time-stamping event and whether this record contains relevant data.

4.8.5 Aligning timestamps

Two time-stamping servers may need to align their work. It may be necessary to compare two events time-stamped by different servers. If servers use absolute time, then, as far as their clocks are trusted, it is easy to compare the timestamps they produce. However, if servers use relative time, or clocks are not trusted, problems arise.

Cross time stamping is implemented by exchanging evidence information. TSA_1 sends evidence of its activity to be time-stamped by TSA_2. Cross time stamping permits two servers to align their activities.

4.9 XML Key Management specification (XKMS)

4.9.1 Description

The XML Key management Specification (XKMS) work is carried out by The World Wide Web Consortium (W3C).(xkms-200010330) and is a document that specifies protocols for distributing and registering public keys, suitable for use in conjunction with the proposed standard XML Signature (XML-SIG).

The XML key management specification comprises two parts: the XML key information service specification (X-KISS) and the XML key registration service specification (X-KRSS).

XKMS uses common data elements in a message set. The <ds:keyinfo> element is specified in the XML-SIG specification. Identifiers include KeyName, KeyValue, X509Cert, X509Chain, OCSP, etc.

X-KISS allows a client to delegate part or all of the tasks required to process XML signature <ds:KeyInfo> elements to a TRUST SERVICE. A key objective of the protocol design is to minimise the complexity of applications using XML signature. By becoming the client of the trust service, the application is relieved of the complexity and syntax of the underlying PKI used to establish trust relationships, which may be based upon a different specification such as x.509/PKIX, SPKI or PGP.

The X-KRSS specification defines a protocol for a web service for the registration of public key information. The protocol provides authentication of the applicant and, in case that the key pair is generated by the client, Proof of Possession (POP) of the private key.

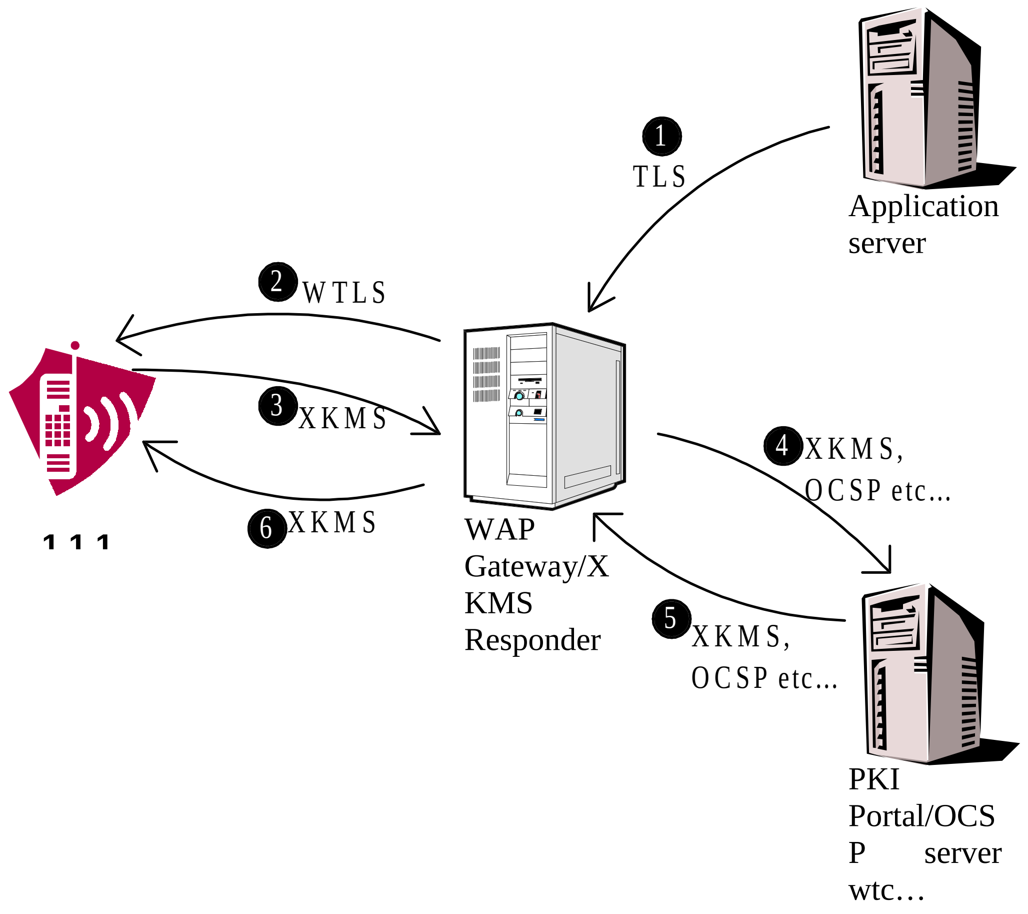
XKMS can be used to retrieve revocation information from an OCSP server without the client having to support OCSP. It provides schemas for other valuable functionality such as encryption and signing, but in this context revocation is the focal point. The revocation information is defined under X-KRSS; it allows the user to request the registration service for the revocation of the bound public key on their certificate. This is achieved by sending the Keybinding schema (an element that asserts a binding between data elements that relate to a public key) with an invalid status:

```
<prototype ID=”keybinding”>
        <Status>Invalid</status>
```

The message is signed by the private key; however it can be signed by the pass phrase that is generated and given to the user at registration time. Once the server knows that a client certificate is invalid, it can then pass the details on to an OCSP server.

In addition to the revocation, XKMS can also validate (with regard to the expiry date) certificates. It has NotBefore and NotAfter tags. If the current time lies between the tags, then the certificate is deemed to be valid on these basis; further to this the XKMS responder can forward validation requests to an OCSP server for the validation of certificate chains.

4.9.2 Example use of XKMS over a WAP session



1. After the client initiates a request to the gateway; for example to start a session with a secure web site, the application server responds with the request information via Transport Layer Security (TLS)
2. The gateway returns this information to the client
3. The client, realises it must validate a certificate associated with the information, creates an X-KISS <validate> message and sends it to the XKMS responder.
4. The XKMS responder determines the revocation status of the certificate. This can be obtained from a CRL, OCSP server etc.
5. Revocation information is passed back to the XKMS responder in an XKMS schema, the XKMS responder then signs this with its private key.
6. The XKMS responder returns the result to the client, the client then verifies the XML signature and uses the validated certificate specified.

The PKI portal can be used for other functionality along with certificate validation. Other uses can include certificate chain path validation, certificate chain discovery, etc.

4.9.3 Features

An XKMS responder can be used to retrieve verification, validation and other details of certificates and certificate chains from an OCSP responder or a CRL. This provides a standardised way of passing and requesting data between the client and the various points needed to access the OCSP or CRL servers. In addition to this XKMS provides a medium for a user to flag her own revoked certificate, this means the client does not need to support OCSP but an OCSP server is still needed for checking against revoked certificates which the XKMS responder communicates with.

4.9.4 Issues

Although XKMS provides a medium for revocation data it does not deal with the revocation itself i.e. the XKMS server will still need to do OCSP. A third party such as an OCSP server holds revocation information. The transferring of this data can be done using XKMS. However XKMS allows for a client to flag that her certificate has been revoked and then this data can be passed on to the OCSP server. This means that the client will need to have support of XML-SIG, XML and XKMS schema's to communicate with the XKMS responder.

5 WP1 requirements for PKI

5.1 Introduction

This chapter identifies the requirements from section 4 of Shaman project deliverable D02 [70] that **may** be relevant to public key based mechanisms and/or the associated public key infrastructure.

The PK or PKI relevance strongly depends on the method of authentication that will be selected by WP1: a symmetric or asymmetric cryptography based method.

The choice between both authentication methods may lead to quite different security architectures and is mainly dependent on the feasibility of defining a global PKI or the possibility for interoperability between different PKIs.

Where possible, the WP1 requirements are mapped to related security features that are described in chapter 4 of D02 regarding how well they are provided by public key techniques.

5.2 Public key and PKI relevant requirements

The following table lists all the requirements from [70] with a mapping to related security features, where possible, and whether the requirement is PK or PKI relevant.

Req.	Related security features (Authentication, Authorisation, Confidentiality, Integrity, Non-repudiation, Anonymity, Availability)	PK or PKI relevant
01	End entity authentication	Y
02	Service authorisation	Y
03	Fraud detection	N
04	N/A	N
05	N/A	Y
06	Confidentiality, integrity, authentication	Y
07	N/A	Y
08	N/A	Y
09	Denial of service	N
10	N/A	Y
11	N/A	N
12	N/A	N
13	N/A	N
14	N/A	Y
15	N/A	N
16	Data origin authentication, integrity	Y
17	Entity authentication, integrity	Y
18	Integrity, confidentiality	Y
19	N/A	N
20	N/A	Y

21	N/A	Y
22	N/A	Y
23	Entity authentication, data origin authentication, confidentiality, integrity	Y
24	N/A	Y
25	Entity authentication, anonymity	Y
26	Data origin authentication, integrity, non-repudiation	Y
27	N/A	Y
28	N/A	Y
29	Entity authentication, data origin authentication, authorisation, integrity	Y
30	Integrity	Y
31	N/A	N
32	N/A	N
33	N/A	N
34	N/A	Y
35	Entity authentication, data origin authentication, authorisation, integrity	Y
36	Entity authentication, data origin authentication, integrity, non-repudiation	Y
37	Confidentiality, anonymity	Y
38	N/A	N
39	N/A	N
40	Authorisation	Y
41	Confidentiality, anonymity	Y
42	N/A	N
43	N/A	N
44	Confidentiality, integrity, anonymity	Y
45	Data origin authentication, entity authentication, integrity	Y
46	Data origin authentication, entity authentication, authorisation, integrity	Y
47	Data origin authentication, entity authentication, confidentiality, integrity	Y
48	N/A	Y
49	Availability	Y
50	N/A	N
51	N/A	N
52	N/A	N
53	N/A	N
54	N/A	N
55	N/A	N
56	Authorisation, availability, confidentiality	Y

57	N/A	N

5.3 Derived PKI requirements

This section describes, where needed, in more detail the PK or PKI relevance of the requirements.

Requirement 1:

Mechanisms shall be available to prevent or adequately minimise the misuse or misappropriation of network services.

This requirement can be fulfilled by the use of public key techniques as explained in 4.1. The support of authentication is based on the use of public key certificates, assuring the public key is the one of the authenticating entity (and not of an intruder or tampered with), and on a common point of trust between the authenticating and assured entities. As certificates are subject to change (expiration, revocation), they have to be (preferably) downloadable from the PKI, which manages the certificates. If a global PKI can not be provided, interoperability between different PKIs is required.

Requirement 2:

Mechanisms shall be available to ensure that customers do not obtain a higher QoS than authorised.

This requirement can be fulfilled by the use of attribute certificates, which contain the QoS level the subject in the attribute certificate (customer) is restricted to. The attribute certificates may be issued by the communication service provider with whom the customer has the service subscription or by a party trusted by the communication service provider and by the access and core network providers which network infrastructure is used to deliver the service. Attribute certificates are a flexible way of carrying authorisation information about a subject for a given resource in a secure manner. An alternative solution could be the inclusion of certain data into the identity-based certificate of a user. This could either be an extension or (if not to many QoS-levels are provided) the issuing party itself (that means several virtual issuing parties).

Obviously the first approach is more flexible, e.g. regarding issuing authorities and revocation issues, but the second is easier to implement.

Requirement 5:

Security mechanisms shall be provided such that they comply with laws and regulations within the appropriate jurisdiction.

This is a general requirement, which also applies to public key based mechanisms and the PKI. This may be achieved by issuing certificates according to the policies applicable within the appropriate jurisdiction. Interoperability issues may arise because of differing policies.

Requirement 6:

Mechanisms shall be available to prevent or adequately minimise the unauthorised disclosure, modification, misuse or misappropriation of user data, including signalling information, location information, billing and accounting information, and user traffic.

This requirement concerns the provision of the security features confidentiality, integrity and data origin authentication, which all may be provided by the use of PK mechanism, as described in 4. Symmetric cryptographic mechanisms are more appropriate to support these security features, however. The required secret session key may be obtained as a result of the initial entity authentication, which could be PK based, couples with a dynamic key management mechanism.

Requirement 7:

Security mechanisms to protect network services shall be standardised such that they are available when roaming.

In case the security mechanisms are based on PK mechanisms, standardisation is required to avoid interoperability issues concerning certificate formats, certificates policy statements (CPSs) and certificate policies (CPs).

Requirement 8:

The strength and cost of the security mechanisms should be balanced against the corresponding security risks.

This is a general requirement, which also applies to public key based mechanisms and the PKI.

Requirement 10:

It shall be possible for all critical security functions (e.g. where user-specific keys are involved) required for heterogeneous access within the same communication service provider domain to be implemented on a single physically and logically secure module. However, the use of multiple security modules within the same domain is not precluded, neither is it mandated that separate security modules should be used between different domains. The security module should be physically secure as well as logically secure. For security reasons, it may be desirable for the security module to be distributed and managed separately to the rest of the terminal, i.e. it may be desirable for a human user to insert and remove a security module after the terminal enters the distribution chain. This is for further study.

This requirement is more difficult to fulfil for asymmetric as it is for symmetric cryptographic techniques, because asymmetric cryptographic operations require more powerful computing devices. Certificate based security methods also demand more storage capacity. Technology constraints taken into consideration for the development of a PKI are described in chapter 7.3

Requirement 14:

Access security mechanisms should be unified across access networks of different types.

Access security can be provided by PK based mechanisms. The unification of these services is not in the scope of this work-package, but at least interoperability can be considered, e.g. by specifying a very general certificate-format.

Requirement 16:

If running on the user's hardware, applications need to be distributed by the application service provider in a reliable and secure way, that the application user can trust that the application is received complete and unmodified from the application provider.

This requirement can be fulfilled by signed code. The client verifies the signature. The only possible direct involvement of a PKI in the action may be an online status request for the certificate used to sign the code. The authentication also provides integrity of the downloaded application.

Requirement 17:

The application service provider requires authentic information on the application user resp. the service subscriber that is required for billing for the application delivery and usage. In an anonymous environment where the application service provider may provide the service on the user's delivery of an anonymous payment token, appropriate measures to prevent the duplication and reuse of the payment tokens has to be installed.

The application service provider needs to authenticate the application user before delivering the application to the user. Entity authentication can be done by PK based mechanisms.

Requirement 18:

User data transferred to the application service provider while using the application service must be protected against modification, copying, or eavesdropping by others. This requirement is related to both the user's personal information required for billing as well as the application related data.

The security services integrity and confidentiality can be provided by PK mechanisms. However, for performance reasons the confidentiality service should be provided by secret key based mechanisms. The secret key can be generated during the authentication phase.

Requirement 20:

The application service provider must be able to protect its application software and data to an appropriate level against deletion, modification, substitution and unauthorised download.

This requirement deals with the provision of access control services to systems. Such services are not usually cryptography-based. However, access control always relies on user authentication, and hence, at least indirectly, these requirements may have PKI relevance if user authentication is based on public key techniques.

Moreover, in a distributed environment access control can be supported by the use of extensions in the identity certificate or by attribute certificates. The extensions, respectively the attributes, indicate the privileges a user is granted by the certificate issuer, which might be the application service provider (in case of attribute certificates) or a party trusted by the application service provider (e.g. the communication service provider).

Requirement 21:

To some extent, it might be advisable to an application service provider to protect his installation of hard- and software against analysis by competitors ("resource hiding").

This requirement deals with the provision of access control services to systems. Such services are not usually cryptography-based. However, access control always relies on user authentication, and hence, at least indirectly, these requirements may have PKI relevance if user authentication is based on public key techniques.

Requirement 22:

It may be required for some application service providers to hide the internal organization and configuration of their hardware and software from competitors.

Same comment applies as for Requirement 21.

Requirement 23:

The content service provider needs a secure channel to the subscriber to allow for the restriction of content spreading. [Editor's note: Solutions to fulfil this requirement will most likely be located at the application layer and therefore might be out of the scope of this project.]

Illegal use of content can be provided by the use of PK techniques as is done by DRM, described in 3.3.

Requirement 24:

The content service provider needs to be able to protect the content data stored on his infrastructure against modification and unauthorised use.

Same comment applies as for Requirement 20.

Requirement 25:

The content service provider requires authentic information on the content user required for billing for the content delivery and usage. In an anonymous environment where the content service provider may provide the service on the user' s delivery of an anonymous payment token, appropriate measures to prevent the duplication and reuse of the payment tokens has to be installed.

The same comment as for requirement 17 applies.

Requirement 26:

User requests for content shall be performed in a way to allow for appropriate billing. An accountable link between the user of the content and the subscriber (who actually has to pay for the content sent to the authorised user in his administrative domain) must be possible.

The request for the content must have associated non-repudiation protection to prevent the content requester subsequently denying the request. This is typically provided by use of a digital signature, the public key for the verification of which must be distributed by an appropriate PKI. At the same time there must be a verifiable link between the content requester and the subscriber, which would also typically be provided through the inclusion of appropriate fields in one or more digital certificates. More specifically, the link may be achieved by the use of attribute certificates indicating whether the user is allowed to use the service or not. The issuer of the certificate may be the subscriber of the service or a party trusted by the subscriber. Obviously this will be the communication service provider of the subscriber.

Requirement 27:

It may be required for some content service providers to hide the internal organization and configuration of their hardware and software from competitors

Same comment applies as for Requirement 21.

Requirement 28:

The terminal supplier requires integrity protection of the terminal's supplierware against unauthorised modifications

Integrity of 'terminal supplierware' is often guaranteed by non-cryptographic means. However, as terminals become more like general purpose computing devices (e.g. PCs), the use of cryptographic techniques to guarantee the integrity of the supplierware may become more relevant.

Requirement 29:

The terminal supplier requires strong authentication and authorisation to allow/restrict remote upgrade and management of the supplierware in the terminal components.

This will typically be provided by cryptographic means, and, in the future, there will be increasing pressure to support PK techniques. Within a closed environment (e.g. where the supplierware is provided by component manufacturer) this should cause no major problems. However, in a multi-vendor 'open' environment, terminal components will need to be able to decide whether they are authorised to accept particular pieces of supplierware on the basis of a public key certificate for the provider of the supplierware. This can be done by MExE, which uses public key techniques.

Requirement 30:

The terminal supplier requires a reliable method for verifying the integrity of the supplierware in the terminal components, and disclaims any and all liability/guarantees in case said verification fails.

This may be provided by a combination of cryptographic and non-cryptographic means (including physical security).

Requirement 34:

The infrastructure supplier requires integrity protection for the operating system and crucial operating parameters of the infrastructure components against unauthorised modifications.

Same comment applies as for Requirement 21.

Requirement 35:

The infrastructure supplier requires strong authentication and authorisation to enable remote upgrade, management, and configuration of operating system parameters and functionality in the infrastructure components (independent of day-to-day operation and maintenance).

Same comment applies as for Requirement 29.

Requirement 36:

The subscriber requests that he is billed only for services that the users in his administrative domain really used. Therefore, authentication information required for billing must be usable in a tamper-resistant way to avoid the chance of impersonating other users, which could lead to unauthorised billing for users that have not used the service.

The provision of this service is most easily arranged using digital signatures, along with the associated PKI to distribute public keys. The service provider will need to have a means of verifying that the certificate owner (for the service user) is genuinely associated with a subscriber (presumably through PKI facilities). Moreover, the service provider will also need to have a means of verifying that the identified subscriber is one who is authorised to receive service (and that there is some likelihood of receiving payment). A possible method to support this requirement is described under Requirement 26.

Requirement 37:

The subscriber requests that his personal data (like address, accounting information, current location information while using a mobile service, ...) is used in a confidential way unless public use is explicitly permitted by the user.

Whilst some of the confidentiality requirements can be met by non-cryptographic means (i.e. via physical security) it is probably necessary for the user to transmit some of this information across the mobile network. In such a case this must be done in a way that preserves data confidentiality and user/subscriber anonymity (where required). If this is done through the use if public key techniques, then the PKI requirements must be managed in such a way that the public key certificates do not them selves compromise anonymity.

Requirement 40:

The subscriber requests that he can authorise network access and grant access to individual services and resources on a per user base for all users within his administrative (contractual) domain.

This authorisation requirement can be provided by use of an appropriate PKI. One way this could be achieved is through the use of attribute certificates, where the user attribute certificates contain the privileges the subscriber grants the user. The attribute certificate may be issued by the subscriber or by a trusted party, obviously the communication service provider.

Requirement 41:

The user requests that his personal data (like address, accounting information, current location information while using a mobile service, ...) is used in a confidential way unless public use is explicitly permitted by the user.

Same comment applies as for Requirement 37.

Requirement 44:

The user requests that his communication data, as well as the communication associated data (e.g. CDRs), are transmitted, stored and processed in a secure way to ensure the privacy and integrity of his communication. Therefore, eavesdropping or traffic analysis of user traffic, signalling and / or data shall not be possible.

Same comment applies as for Requirement 37.

Requirement 45:

The user requests that no other user is able to perform operations or access services in place of him and at the user's cost. Masquerading as another user shall be prevented by appropriate authentication measures.

User authentication can be supported through the use of digital signature based authentication protocols. Use of such protocols requires establishment of a PKI, or other means of reliably distributing public verification keys. See Section 4.1.

Requirement 46:

Foreign applications that need to run on the user's hardware shall be distributed by the application service provider in a reliable and secure way, that the application user can trust that the application is received complete and unmodified from the application provider.

Application integrity and authenticity can be supported through the use of digital signatures. Use of signatures requires establishment of a PKI, or other means of reliably distributing public verification keys. An example of how this can be done is provided by MExE, which uses public key methods.

Requirement 47:

The user requests that his application data is protected on the way to and from an application service provider against modification, copying and eavesdropping by others than those authorized by the user.

Same comment applies as for Requirements 6 and 37. This requirement relates to the security features authentication, integrity, confidentiality and authorisation, which are supported by public key methods as described in Chapter 4.

Requirement 48:

The user requests that applications provided or run by an application service provider actually perform only those operations that they are claiming to perform. No "hidden" transactions or operations shall be performed.

This requirement relates to guarantees about the functionality of applications. In that this is not primarily a cryptographic issue, then this is marked 'N/A'. However, if the scope of this requirement includes protecting the integrity of the software whilst stored or in transit, then this needs to be changed.

Requirement 49:

The user requires high availability of network connections and service access. Therefore, denial of service attacks which would prevent the user from accessing the network and/or services shall be prevented or adequately minimised.

The impact of some denial of service attacks can be reduced by careful design of the cryptographic protocols. This can include restricting use of (computationally intensive) PK techniques until certain threshold checks have been passed.

Requirement 56:

The LEA requests, based on a lawful authorisation, access to the stored user data of an operator or provider.

Same comment applies as for Requirement 5.

5.4 Conclusions

The method chosen for authentication, a shared secret or public-key based method mainly determines how authorisation is supported. In case authentication is based on the use of certificates, extensions in the identity certificate or associated attribute certificates can be used to provide authorisation.

The security services data origin authentication, confidentiality and integrity may be provided by symmetric cryptographic techniques. The shared secret is obtainable from public key based authentication coupled with a dynamic key management.

In case the authentication is based on symmetric methods, then the involvement of the home network to obtain the necessary keying material is needed. This implies there is a trust relation between the access network provider and the home network of the user. The use of certificates for authentication makes the involvement of the home network unnecessary.

The feasibility of defining a global PKI or the possibility for interoperability between different PKIs mainly determines the method for authentication. To avoid interoperability problems standardisation is needed, e.g. of certificate formats.

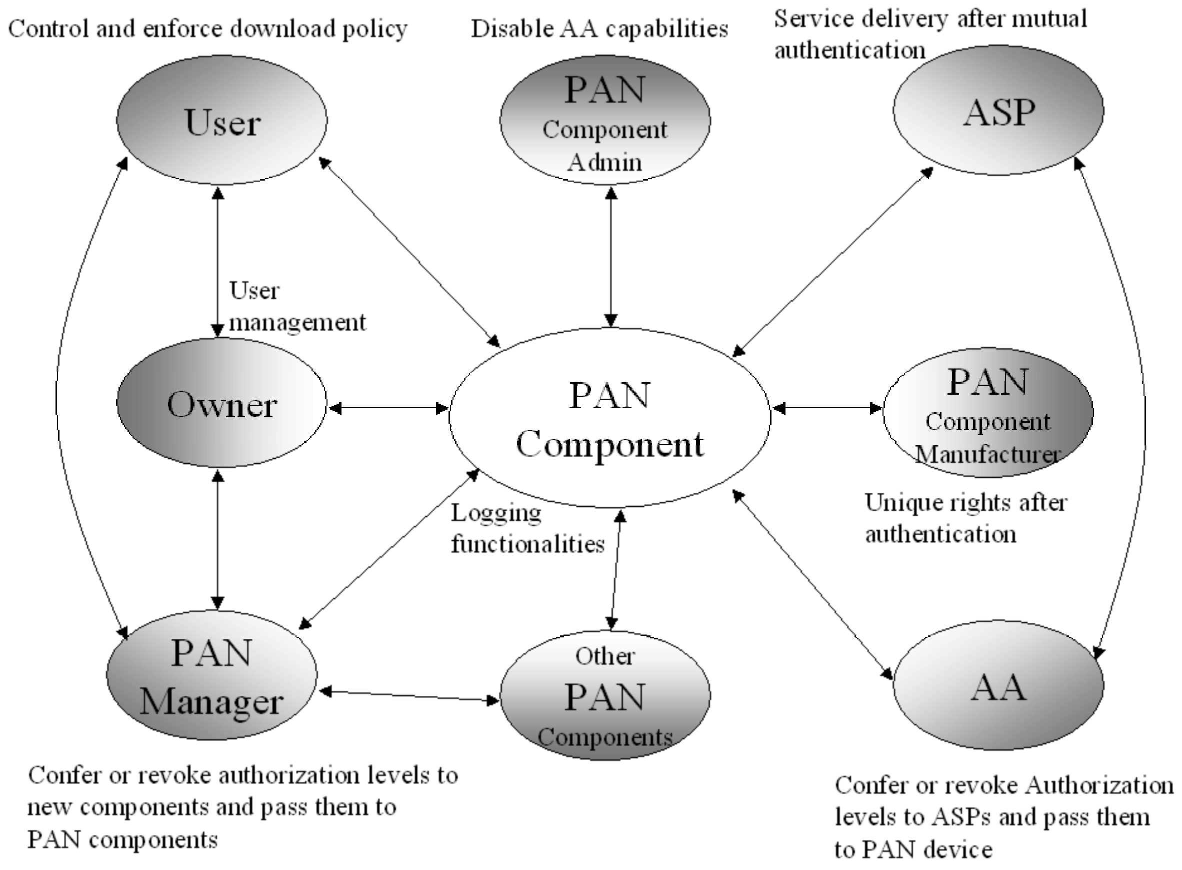
Another topic to consider in case of a public-key based method for the provision of security services are the constraints posed by the wireless technology. To cope with this the outsourcing of complicated functionality from mobile devices might be the appropriate way.

6 WP2 requirements for PKI

6.1 The security requirements from D03

This chapter refers to Shaman project deliverable D03 [71], which describes a security architecture for future terminals and applications. We will focus on the 34 requirements given in chapter 5 of D03 and analyse if these requirements can be fulfilled with PKIs and/or PK-applications.

As WP2s major focus is the mobile terminal all requirements are device-centric. Nevertheless nine roles of interacting parties have been specified and the requirements have been listed under these roles. The following diagram gives a rough overview about the roles and the relations. More detailed profiles are provided in D03.


Control and enforce download policy
Disable AA capabilities
Service delivery after mutual authentication
User
PAN Component Admin
ASP
User management
Owner
PAN Component
PAN Component Manufacturer
Logging functionalities
Unique rights after authentication
PAN Manager
Other PAN Components
AA
Confer or revoke authorization levels to new components and pass them to PAN components
Confer or revoke Authorization levels to ASPs and pass them to PAN device

As with the requirements from D02 (discussed in chapter 5), the PK-/PKI-relevance is strongly dependent on the mechanisms chosen for authentication, i.e. symmetric or asymmetric means, and on the availability of interoperable PKI-structures. Therefore all requirements that **may** be PK-/PKI-relevant will be discussed in the next sub-chapter.

6.2 Public key and PKI relevant requirements

The list of requirements is now mapped to features, where column two gives the information if the requirement is related to Public Key Cryptography or PKI and column three maps the relevant security features if possible.

require	Role	PK- or PKI	Related Sec.- (and other) features

ment		related (Yes/NO)	(Authent./Autho./Conf./Int./Non rep.)
1	User	Yes	Confidentiality
2	User	Yes	Integrity
3	User	Yes	Authentication
4	User	Yes	Authorisation
5	User	No	(Local Policy Enforcement)
6	User	No	(Device-specific Logging-features)
7	User	No	(Local Policy Enforcement)
8	User	No	(Local Policy Enforcement)
9	User	No	(Local Policy Enforcement)
10	PAN-Component-Owner	Yes (Local PAN-PKI?)	Authorisation
11	PAN-Component-Owner	???	Authorisation (Local Policy Enforcement)
12	PAN-Component-Owner	Yes (Local PAN-PKI?)	Authorisation
13	PAN-Component-Owner	Yes	(Local Policy Enforcement)
14	PAN-Component-Owner	Yes	Authorisation (Local Policy Enforcement)
15	ASP	Yes	Authentication
16	ASP	Yes	Authentication
17	ASP	Yes	Confidentiality
18	ASP	Yes	Integrity
19	ASP	Yes	Integrity (Transfer Security)
20	ASP	Yes	Confidentiality (Transfer Security)
21	PAN-Component-Manufacturer	Yes	Authentication
22	Authorisation-Auth.	Yes	Authentication
23	Authorisation-Auth.	Yes	Authorisation
24	Authorisation-Auth.	Yes	Integrity
25	Authorisation-Auth.	Yes	Confidentiality
26	Authorisation-Auth.	Yes	Authentication (Local Policy Enforcement)
27a revoke	Authorisation-Auth.	Yes	Authentication
27b communicate	Authorisation-Auth.	No	(Distribution Mechanism)
28	PAN-Manager	Yes	Authentication

29	PAN-Manager	Yes (Local PAN-PKI?)	Authorisation
30	PAN-Manager	Yes	Integrity (Transport Security)
31	PAN-Manager	Yes	Confidentiality (Transport Security)
32	PAN-Manager	Yes (Local PAN-PKI?)	Authentication (Local Policy Enforcement)
33a revoke	PAN-Manager	Yes (Local PAN-PKI?)	Authorisation
33b communicate	PAN-Manager	No	(Distribution Mechanism)
34	PAN-Manager	Yes	Authorisation (Local Policy Enforcement)

6.3 Derived PKI requirements

The PK- or PKI-relevant requirements are now described and evaluated in detail.

Requirements 1 and 2:

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

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.

These issues may be addressed by the bearer network (e.g. UMTS-security, ...) which does not at present use PK techniques for security, but a there might be use of PKI techniques if devices connect spontaneously over open networks.

Requirement 3:

It shall be possible to provide authentication of the source of applications downloaded by the user.

This requirement can be fulfilled by signed code. The signature is verified by the client. There may be the involvement of other entities if an online status request for the certificate used to sign the code is required, or if the client uses a trusted PKI service provider to provide certificate chain building facilities (using DPD or SCVP) or if the client delegates all the certificate processing to a trusted PKI service provider using XKMS or SCVP.

Requirement 4:

It shall be possible for the user to determine whether an authenticated source is an authorised source of applications or not.

To meet this requirement local authorisation structures have to be implemented. The most obvious concept to deal with this issue at the moment is MExE, offering a different classes of implicit trust that may be configured by the user to a certain extend. The direct involvement of a PKI may be necessary if the client delegates all its authorisation decisions to a trusted authorisation provider using a protocol such as SAML.

Requirement 10:

It shall be possible for the PAN component owner to only grant access to use the PAN component to authorised parties.

This issue could be addressed by local authentication mechanisms, such as PINs and biometrics. Where the owner is granting access remotely, the issue may be addressed by PKI-concepts. To decide whether an entity should get access, it must be authenticated first before the owner can decide about the authorisation and the authorisation level. This authentication may be done via certificates from a PKI. If the authorization is done via PKI-mechanisms as well, e.g. attribute certificates one can think of two solutions: on the one hand a general PKI could be asked to generate such a certificate or on the other hand a certificate could be generated in a local PKI of the PAN. Attribute certificates are a sophisticated mechanism though and might be seen as too complex for a local PAN, if the local PAN is the only user of attribute certificates. “Ordinary” certificates could also be used, with appropriate extensions, for authorisation.

Requirement 11:

It shall be possible for the PAN component owner to only grant access to accrue charges to authorised parties.

This requirement is only hard to handle as there are different scenarios how charges can arise. Possible services which may cost money are:

- Subscription-based-services: here an authentication mechanism will be used by the service provider. So this mechanisms could be projected from the owner (who in this case might be the payer) to the user of the devices. PK-/PKI-mechanisms can be included (depending on the authentication mechanisms).
- More difficult are the services where the user can agree spontaneously about a less obvious or indirect payment, e.g. by using high-cost phone-lines.

The requirement is closely related to requirement 12.

Requirement 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.

Again, the requirement may make use of PK and PKI-mechanisms as described, e.g. in requirement 10. This PKI could be locally based only, or a global PKI, with its ready made facilities may also be used.

Requirement 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.

Part one of this requirement refers again to local policy management. This can be done, e.g. by concepts like MExE, which are using PK-techniques, , in this case, signed code. The negotiation with the ASP has to be specified in more detail to evaluate the PKI-relevance of this issue. In this context one can think of PKI-related protocols for the negotiation itself or of negotiation-protocols secured by PK-means. The security policy may be contained or referred to in certificates for the ASP and PAN component.

Requirement 15:

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

This requirement may certainly be fulfilled by PK- and PKI-mechanisms. The PKI will be involved first in issuing certificates to the client and then perhaps in the provision of revocation-lists or answers to online status requests. Alternatively the client could be authenticated to the ASP using symmetric approaches, for instance, using the cellular bearer security mechanisms, with the client’s mobile operator passing the authentication information on to the ASP.

Requirement 16:

It shall be possible for the client to authenticate the source (by definition, the ASP) of a downloaded executable.

This requirement seems to be closely related to requirement 3.

Requirements 17 and 18:

It shall be possible for the downloaded executables to be sent from the ASP to the client with confidentiality.

It shall be possible for the downloaded executables to be sent from the ASP to the client with integrity.

It is possible to secure the transmission of downloads with PK-mechanisms to assure confidentiality and integrity. Ideally this will be provided end to end.

Requirements 19 and 20:

It shall be possible for the ASP to obtain the clients terminal capabilities with integrity.

It shall be possible for the ASP to obtain the client terminal capabilities with confidentiality

Some existing security protocols (e.g. TLS) provide as secure means to obtain the client’s security capabilities with integrity. Provision with confidentiality of security capabilities is more difficult but is generally not necessary. Where other client capabilities have to be provided with confidentiality, this can be done using existing mechanism such as WAP UAProf and http headers, but over secure sessions such as TLS.

Requirement 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.

This is an essential requirement that has to be addressed very thoroughly as the PAN-Component-Manufacturer has the most privileges concerning the upload of new software to the device. Certainly PKI-mechanisms can be used for this authentication.

Requirement 22:

It shall be possible for the AA to authenticate ASPs requesting authorisation.

PK-/PKI-mechanisms can be used, but this is more likely to be done using out of band mechanism as this authentication is done prior to the enabling of the ASP with PKI capability, so existing PKI capability at the ASP cannot be relied upon.

Requirement 23:

It shall be possible for the AA to confer variable authorisation levels on ASPs.

To fulfil this requirement you have to identify the authorisation-mechanism first. The choice will not only depend on the capabilities of the user’s device but also on the granularity of the authorisation-levels required. When you think of PKI-related approaches for this task attribute certificates are one possible solution. A CA (and therefore a PKI) would be directly involved by issuing theses certificates. This concept of certifying attributes instead of keys needs clients capable of dealing with such certificates. Another potentially easier approach could be to use identity x.509 certificates (as is done in MExE) and associate certificate extensions or particular roots on the clients with particular authorisation levels so that the CA signing an ASPs certificate is related to one of these levels.

Requirements 24 and 25:

It shall be possible for the authorisation level conferred on an ASP to be passed with integrity to the client.

It shall be possible for the authorisation level conferred to an ASP to be passed with confidentiality to the client

Integrity protection must be provided end to end between the client and the ASP, so is best done using application layer signatures. Confidentiality is not so crucial, and may therefore be provided by session or network layer mechanisms, which may not necessarily be end to end.

Requirement 26:

It shall be possible for the client to obey (or to enforce) the authorisation level received.*

* One can look at this requirement from two different views. The term 'to obey' may be suitable in certain scenarios, where the device is considered as potentially malicious. On the other hand, when you suppose the device to be 'well-behaved', it has to be assured that an application with a certain authorisation level really gets only the resources it was indirectly granted by the AA. Here the wording 'to enforce' would fit better.

Requirement 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.

This requirement is has to be looked at from two different angles. The first part demands the possibility to withdraw authorisation levels. This can be done depending on the mechanisms used to confer the levels to the ASP. If these mechanisms are certificate-based (and therefore PKI-related) a revocation-management has to be implemented. The second feature demanded is the information of the terminals about the withdrawal. This part is far more difficult to achieve as it is formulated at the moment. A better wording would be *'[...] and ensure that the information about the withdrawal is made available for the terminals'*. Then (in the case of a certificate-based solution) the usual revocation-mechanisms could be used.

Requirement 28:

It shall be possible for the PM to authenticate PAN component users requesting authorisation.

PK-/PKI-mechanisms could be used, probably at layers higher than that on which the inter-PAN communication will take place (layer 2). Alternatively, layer 2 security mechanisms (most likely symmetric) could be used. (See also requirement 15).

Requirement 29:

It shall be possible for the PM to confer variable authorisation levels on PAN component users.

Authorisation can be conferred using PK(I) techniques as described in Requirement 23. This issue deals with local structures. If PKI is used it could be a local PKI in the PAN or the facilities of a global PKI might also be used.

Requirements 30 and 31:

It shall be possible for the authorisation level conferred on a PAN component user to be passed with integrity to other PAN component users.

It shall be possible for the authorisation level conferred to a PAN component user to be passed with confidentiality to other PAN component users.

(See requirements 17 and 18)

Requirement 32:

It shall be possible for the other PAN users to obey (or to enforce) the authorization level granted to the new PAN component user.*

* Again two different points of view exist when you look at this requirement. Here, to obey can be interpreted as 'really giving the new PAN-component the rights it was granted by the PM'. 'To enforce' is more related to assuring the restrictions of the new component related to its authorisation level.

Requirement 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.

Closely related to requirement 27. The major difference to this requirement is, that now everything happens in a local structure, the PAN, which means, that pushing the information about a withdrawal may be a lot easier.

Requirement 34:

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

This requirement again is about local policy management. AAs will possibly be identified and approved by certificates, so the PM has to get the right to change trust levels for certain parties in the local certificate database of the Pan-Components.

6.4 Conclusions

If one has a closer look at the requirements, it becomes obvious that they can be categorised, at least to a certain level. The following issues occur repeatedly:

- **Authentication of one party against another:**

  Authentication is a major issue concerning the requirements from D03. It is not only a central issue but also a crucial one, as the solutions for many other requirements will depend directly on the mechanisms that are chosen to provide authenticity. Symmetric solutions may be chosen, when there are relationships between the proving party and the verifier based on subscription-like concepts. For instance a provider of a GSM-network today has symmetric keys with each customer anyway. Nevertheless the approach of spontaneous networking will make such symmetric means appear inefficient and unmanageable. Here PK-/PKI-based services come into consideration. The difficult aspect of PKI-based-applications on mobile devices is to implement them transparently and efficiently. So outsourcing complicated functionality like verification of certificate- and CRL-chains to trusted servers might be the appropriate way.

- **Transport security, i.e. confidentiality and integrity:**

  It may be necessary that different requirements need different mechanisms to achieve the confidentiality they require. Confidentiality is likely to be provided at different network layers.

- **Authorisation of parties:**

  Authorisation is needed in different scenarios, and a secure authorisation mechanism is essential when one entity is accessing significant functionality of another. If "authorisation" is seen as the transmission of messages (of an agreed format) with authentication and integrity, then it can be seen that appropriate mechanisms for authentication and integrity must be provided. Authentication has been discussed above. Integrity is also likely to rely on public key techniques so that integrity protection can be provided *end to end*, and between parties where secret keys cannot easily be established.

  - **PAN-internal authorisation issues:**

    To manage authorisation levels in PANs one can consider different mechanisms. One solution can be to introduce a PAN-internal PKI so that the PAN-manager may certify certain attributes to PAN-components.

  - **'Global' authorisation issues:**

    In contrast to the local scenario, not only PAN-components are involved but also external parties like AAs or ASPs. PKI surely can be introduced to fulfil these requirements.

- **Withdrawal of authorisation:**

  - **PAN-internal withdrawal:**

    The case that certificates get revoked in this context is less problematic than in 'global scenarios' as the PAN-manager will always be able to push this information to the PAN-components that are online. However the question does arise as to whether a PAN component should authenticate itself to the PAN (or MAN manager) every time it joins the PAN or whether the authentication can be valid for a series of sessions.

  - **'Global' withdrawal:**

Here the problem of a proper revocation-management is obviously harder as in local environments as a push-service may be much more difficult to implement an the different involved parties act independently of each other. The AA for example may have revoked certain rights for an ASP but is probably not able to distribute this information to all clients relying on the AA-generated authorisation-'tokens'. Therefore it might be possible for a malicious ASP to use his authorisation-token after the AA has revoked it.

- **Local policy enforcement:**

  A number of requirements can be described as local policy enforcement. That means that certain authorisation- and device-management-issues can be configured by the owner or the user locally. To achieve to intended policy PK-/PKI-mechanisms may be used. But the real action is not performed by the PKI itself but by the owner/user. An example for this kind of requirement is MExE, where PK-/PKI-mechanisms are used to project authorisation-levels to signed applications but the user may configure what kind of alert-handling he prefers: e.g. to be asked every time an application from a certain provider arrives, to be asked once a session or just to be asked once.

7 Organisational issues

7.1 Key management issues

7.1.1 Key pair generation

There are a number of different scenarios for the generation of key-pairs to be used in Public Key Infrastructures.

- Distributed generation of the key pair in the mobile device by the user.

  As the public key has to be certified after generation there are a few points to consider. How is the authentication of the user requesting the certification done? How is the public key to be certified? Is the sent key a cryptographically 'good' key? Do we need a proof of possession of the private counterpart of the public key to be certified?

  As the user generates the key pair, the certifying party has no control over the private part, so that it is not possible to back up a private key. This means that lawful interception requirement can not be shifted to the responsibility of the PKI-provider.

  As the key is not generated on-card it will be stored in software, which makes it vulnerable for an attacker. Refer also to the requirements for WTLS where the private key MUST be stored on a tamper resistant device. Therefore PSE-security (Personal Security Environment), i.e. the different tokens and security-mechanisms for storing sensitive key material, has to be discussed thoroughly at that point.

  Furthermore the trustworthiness of the mobile device is still an open point, since a trust model for post-3G systems has not yet been specified. It has to be discussed how the trust status of the mobile affects the key generation and the acceptance of the key by the CA.

- Distributed generation of the key pair on the SIM by the user.

  This scenario is very similar to the first one. The same requirements arise and the same advantages concerning lawful interception can be referred to.

  The major advantage is the secure storage of private keys on the SIM (see also 7.1.2).

- Central generation of the key pair by the manufacturer of the device or the owner of the SIM, before delivering the device or SIM.

  Here we have the advantage, that the manufacturer or SIM card owners can generate the key pair in a secure manner and that this generation process can guarantee 'cryptographically good' keys.

  In both cases, a general key of the manufacturer/provider can sign the keys, so that they can later be recognised as genuine keys in a certification process.

  As the provider has access to the private key that is generated, lawful interception issues may arise. That means that, provided that law demands this, a provider may have to backup all private keys.

- Centrally organised generation of keys on request.

  In contrast to the central generation of keys before delivering the device or the SIM-Card there is a possible scenario where a user requests a certificate when he already possesses the device/SIM-card but is not able to generate the asymmetric keys himself. Again the advantage arises, that the manufacture/provider is assured of the quality of the generated key-material. Nevertheless one big issue arises: the problem of an authentic private key distribution. As there might be no anchor of trust yet, the user might not be sure about the party he is really communicating with. Additionally, the sending of private key material over the net has to be secured properly, which raises further key management problems.

7.1.2 Key storage

Private cryptographic keys may be stored in different places that differ in their access-properties, the security and the flexibility. Sensitive data may be stored:

- On the SIM

  This means that the key-pairs have to be generated by the network-provider and have to be put onto the SIM before it is transferred to the customer or, via still to be specified mechanisms, when a subscriber already uses the SIM. The key-pairs can also be generated and put on the SIM by the user himself, using appropriate key generation tools (see 7.1.1). The public part of a key pair could be certified, when the SIM is assigned to a client via the information available in the concluded contract. To be sure that the public key to be certified is a genuine key, it may be signed by a general certification key of the provider, before it is put onto the SIM. In case the key-pair is generated by the user himself, certifying of the public key is an open issue (see 7.1.1). This solution does not offer the possibility to export the users private key and therefore to use it outside the SIM. This feature may be favourable in certain scenarios, but it may also be seen as a restriction in others.

- On an additional smart card

  When third parties want to act as PKI-service-provider the scenario may be considered where the third party issues its own smart cards containing the private pre-generated key-material or even the final certificate. These cases require the use of special terminals where secondary smart cards are addressed via a second slot in the device or local radio interface-solutions like Bluetooth. This solution does not offer the possibility to export the users private key and therefore to use it outside the additional smart card either. This feature may be favourable in certain scenarios but it may also be seen as a restriction in others.

- In Software

  The storage of private key-material on the device in software is always a higher risk than storing the material on hardware. Several security-issues arise: an attacker may grab a software-based PSE so that it can be compromised by long-term attacks. Additionally the risk of manipulating the PSE without the recognition of the owner should not be neglected. On the other hand software-based PSEs offer greater flexibility, because they enable the user to back up the whole PSE and to export the keys and then use them on several machines in parallel. Additionally one can think of scenarios with central key stores, i.e. a central storage of PSEs, from where the user can temporarily download its credentials and use them.

7.1.3 Key backup, key recovery

Obviously key-backup is only reasonable for encryption keys. Signature or authentication keys must never be backed up as there is no reason to do so.

As described in 7.1.1 the feasibility of backup-mechanisms depends primarily on the key-generation-process. To be able, to back-up key-pairs, the central generation approach has to be chosen. Otherwise, the private keys shall never leave the computing environment of the user and therefore will never be accessible by a third party, e.g. the network-provider or a third-party provider. Certainly, the key-backup discussion is one of the most sensitive issues in PKI. Therefore it is important to investigate this subject thoroughly and to define the scenarios where back up is necessary and to consider the related key recovery issues.

7.1.4 Key Update

When a certificate expires, a new certificate is usually generated for the certificate-owner. The new certificate might contain the old public key signed again by the issuing authority, or the issuing authority may sign the public part of a freshly generated key pair. Whether the first or second approach is chosen depends on the circumstances under which the keys are stored on the user’s device, the key length, and the current security of the algorithms used.

If the secret key is stored on a hardware-token it is certainly more secure against long-term attacks than a key stored only in a software-PSE that can be copied without the users knowledge. So the re-use of old key material might be appropriate when the private key is stored in hardware, whereas the use of software-storage might require an update of the key material.

7.1.5 Key history and archive

The key-history feature enables a user to deal with data encrypted under an old key, i.e. the user is still able to decrypt content with an old key at a point of time where he is already using another (new) key. To implement this functionality, old keys must be available for the user, i.e. in most cases be stored in his PSE as well.

It has to be discussed in which scenarios key history is necessary at all. The requirement for this service is mainly for confidentiality keys. If one thinks of environments where encryption is not used to store data securely in a permanent way, key history need not be implemented. E.g. you can look at e-mail-services as a mechanism that assures confidentiality during the transport of information only; (unfortunately, many people use e-mail-accounts as a permanent storage-medium too).

The same applies for key archiving, which is the long-term storage of keying material. Key archive differs from key history in the sense that archival can be used for audit purposes as well as to help resolve disputes.

If these services are required, it has to be decided where the key information is to be stored, and which key retrieval method(s) to use.

7.2 Certificate management issues

7.2.1 Registration

Registration is a crucial point in the certification process. As a certificate usually binds identity-related information to a cryptographic key, a certificate is only as good as the identification of the owner is done. Different scenarios therefore need different registration-procedures. Besides the ‘quality of trust’ a certificate has, due to the mechanism that was used to identify the user, one has to consider the effort one has to spend in certification. Most users won’t accept too complex or time-consuming registration-procedures. Therefore the certifying party has to choose its registration-mechanism so that the trade-off between effort and thoroughness of identification leads to a satisfying result for the user and the parties that rely on the certificate later. Furthermore the quality of the registration process affects the legal validity of the signatures.

Moreover one has to look at the different parties who might offer the certification-services.

The network provider who has already a contract with the customer can provide certificates with a relatively high trust-level without complex request- and identification-mechanisms as an identification has already be performed by the initial subscription. After this initial subscription the user can be related for instance to a mobile phone number so that the provider has a secondary means of identification.

Other service-providers not owning a customer-database are in a more unfavourable position as they can not rely on an already available customer-database. A registration that uses an equal mechanism as in the last example might therefore lead to certificates with a lower level of trust unless the third party provider co-operates with the network-provider when doing the identification.

Possible mechanisms for registration:

- The user has already subscribed for a mobile service. Therefore the network provider has already done an initial identification. In the case, that a network-provider is the PKI-provider at the same time or that the PKI is hosted FOR the network-provider, the identification already performed can be the basis for a certification-request. An example for this scenario (used in fixed networks) is the registration-process T-Online (the ISP of Deutsche Telekom) uses for offering secureMailcertificates to their users. T-Online uses the temporary IP-address given to a customer to get its personal data via an inverse RADIUS-query. The certified personal data is thus verified as authentic.

- If no former relationship existed between the PKI-provider and the user, the identification is more difficult to perform in a trustworthy way, especially as in 4th generation networks a subscription of the user cannot necessarily be assumed. Therefore the situation may be equivalent to today's use of the Internet in an anonymous way. If the requester is not already known to the issuer the following request-processes may be considered:
    - It is only checked, whether the reply-address in the request can be reached and therefore exists. The feasibility of this approach depends on the personal data to be included in the certificate. Perhaps something like a permanent address does not exist at all.
    - An out of band mechanism is used for the registration process (phone call, mail, etc.).
    - The requester has to register in person at a local RA.

7.2.2 Certificate generation and distribution

The distribution of freshly created certificates can be done via different mechanisms. Generally one has to differentiate between the pushing of certificates and the pulling of certificates.

The different request-mechanisms will influence the way the user will get his/her certified key. If the request-procedure is very quick it is possible that the requested certificate might be generated on the fly and that it is sent to the user during the same session. If the identification takes place with out-of-band mechanisms this concept will not work out. Then one has to specify whether the certificate is sent to the user (pushed) or retrieved by the user (pulled).

To send a certificate to a requester, the requester certainly has to be related to a fixed account or a fixed device, for instance a mobile phone account that can be reached. If the requester is only spontaneously connecting to a network, push-services are not feasible. In these scenarios the user has to know, where he might be able to fetch the certificate. This directory-information might be public or it can be transmitted during the request-process.

7.2.3 Certificate dissemination and retrieval

The dissemination of certificates can be done via two different means:

- The certificates are made public by pushing them into public directories where users can look for them (retrieval).
- The certificates (optionally the certificate-chains) are requested at the beginning of a session, are sent and may be stored locally at the receivers device or not (dissemination).
- The certificates (optionally the certificate-chains) are sent categorically during the establishment of a secure session (dissemination).

The storage of certificates on mobile devices should not be a problem regarding storage space. Typically certificates consist only of a few thousand bytes.

7.2.4 Certificate validation

The validation of certificates is a very delicate issue. It has to be done accurately to assure that the right result concerning the validity of signatures or the authenticity of certain sources is delivered. At the current state two major approaches are used: the validation of certificates via CRL-checking (Certificate Revocation Lists) along the certificate-path, or online-validation i.e. asking a trusted party about the current status of a certificate (for further information see Section 4.7). In the following paragraphs the proposed mechanisms will be looked at from the mobile device's point of view.

7.2.4.1 The use of CRLs in mobile devices

The Checking of CRLs, which have a certain validity period (typically relatively short lived), can be done offline, if the relevant CRLs are already cached on the users device. Unfortunately this advantage has to be looked at in more detail, to see what a user has to pay for it. Especially in the mobile environment there are certain technology constraints concerning storage space, processing power and bandwidth (see 7.3).

Storage space on mobile devices can be very limited if you look e.g. at mobile phones. However CRLs can get very large. This means the caching-capacities of mobile devices for CRLs may be insufficient to store all relevant CRLs, so that these lists have to be retrieved several times even throughout the period in which they are valid. Therefore the advantage of offline checking may vanish partly on devices with only little storage space. Additionally the download of CRLs may be a bottleneck for devices, which are only connected over a narrow bandwidth. Of course, the bandwidth problem should get more and more negligible with the introduction of new technologies like GPRS or UMTS. As already implicitly said this checking has to be done for the whole certificate path that may consist of several levels. These problems have already been addressed in the fixed network PKI standardisation. The concepts of delta-CRLs and distributed CRLs have been proposed. On one hand these concepts reduce the amount data that has to be downloaded in order to check a certificate but on the other hand they require an additional logic in the client device to deal with such CRLs. The storage of the certificates themselves should not be a major problem, as they usually do not exceed a size of 2-3k and do not need a really trustworthy environment to be stored in. That means that certificates of the people your are communicating with won't be stored on hardware-token.

On-line validation has been proposed to reduce the amount of data that has to be downloaded and stored by the client who wants to validate certificates, e.g. using the OCSP protocol (for more details see Section 4.7.1.2). The main validation activities have been shifted to a trusted party. A client only requests a status of a certain certificate and waits for the signed answer. Therefore the client has not to store any revocation-information as in the CRL-scenario but has to be online every time a certificate-check has to be performed. To provide such a service in a scalable way, the trusted party has to be able to process a multitude of requests at a time.

On-line verification seems more favourable for the use of PKI-functionality in mobile devices as no bandwidth problems arise. On the other hand this kind of validation is only possible if the device is online, whereas checking via CRLs can be done off-line (at least to a certain extent, as explained in the second paragraph).

7.2.4.2 The use of short-lived certificates

At this point one has to look at another solution that is currently used in WAP-scenarios. Here so-called short-lived certificates are generated. That means certificates, which are only valid for a very short time, e.g. a day. In this approach no real revocation management is necessary as a compromised certificate reaches its validity-limit after a short period of time anyway. In the case of a compromised key the consequence in this scenario is that no new (short-lived) certificate is generated for the concerned party unless a new key is used. The striking advantage is that the client devices do not have to perform CRL-checks or online-validation-requests. They only check the validity-period of the certificates in the certificate-chain. Of course this solution results in an overhead at the certifying party, who has to generate lots of certificates every day. Moreover local certificate-caching is more difficult, as they change on a short term basis.

7.3 Technology constraints

The development of a public-key infrastructure supporting the provision of the security features for next generation networks has to take into account the *bandwidth*, *storage* and *processing* constraints, imposed by future mobile environments.

Mobile environments have less *bandwidth* compared to wired networks. The hand-held wireless devices have a constrained computing environment, due to limitations of power and form-factor. They tend to have less *processing* and *storage* capacity. The same applies to the smart card, a tamper-resistant device, which is used in performing security functions, and especially, to store and process information needed for user identification and authentication. On the other hand, these boundary conditions change at a rapid speed. The best example is the development of handheld computers, i.e. PDAs. Most Palms had no more than 2 Mbyte of storage-space a year ago (currently 8 Mbyte) whereas the COMPAQ I-Pac has come over these restrictions with it’s 64Mbyte of available memory.

The following sub-sections describe the involvement of the PKI components with each of these constraints and the possible effect on network performance, call set-up. Restrictions are identified according to the characteristics of certain devices as well as to the actions that have to be performed in a PKI-environment.

7.3.1 Storage constraints

7.3.1.1 Devices

The most difficult environment with regard to storage space is certainly the personal hardware token, i.e. smart cards, USB tokens, etc. These tokens offer only very limited physical memory and processing capability. Therefore, only the most critical operations (regarding security) should be performed on these devices. All other operations might be ‘outsourced’ to the hosting device, i.e. a smart card-reader or the computing environment in the background (mobile device, PC, ...). Current smart cards typically offer the following resources and processing power: ROM 64K, RAM 4K, EEPROM 32K and CPU 8bit / 5-15MHz.

The other group of devices to be considered includes handheld-computers and mobile phones. The problem with this group is that the range of devices, and therefore the diversity of their features, grows every day. The convergence of these devices, already started, is subject to different approaches and will be difficult to achieve in the near future due to, e.g., different operating systems. Therefore proprietary solutions will arise. To cope with this, minimal storage requirements on mobile devices have to be defined to make PKI services available for a broad range of customers. Certainly, these minimum storage requirements will not be sufficient for all services. *So a service with sophisticated features offered by a provider might only be available for customers that use the most up-to-date devices.*

7.3.1.2 Relevant PKI-processes:

Storage constraints may be a main factor in selecting a certificate validation mechanism, as explained in section 7.2.4.1.

7.3.2 Processing constraints

7.3.2.1 Devices:

The observations about the type of device used in a PKI scenario concerning the storage restrictions can be transferred to the restrictions given by less powerful computing environments. Therefore only those operations that involve the private key information should be performed on hardware-tokens (if available at all). Again, all other operations should be delegated to the host.

7.3.2.2 Relevant PKI-processes:

All operations done with the public or the private key in PKIs are obviously asymmetric cryptographic operations. These kinds of operations are a lot more costly than symmetric cryptographic functions. Moreover costs of such asymmetric operations depend on the key-size. Relevant operations in this context are decryption and digital signing. To reduce the complexity of these operations confidentiality is mostly provided by hybrid cryptosystems, where the content is encrypted with symmetric means and only the symmetric session key is encrypted with the public part of the asymmetric key-pair. A somewhat similar mechanism is used to reduce the costs for digital signing. The data itself is not signed with the private part of the asymmetric key pair of the user, but instead the asymmetric signature function is applied to a hash-code, computed using a cryptographically secure hash-function. One has to consider, especially when hardware-token like smart cards are involved, what operations under which key-sizes are feasible.

7.3.3 Bandwidth constraints

In this paragraph one has to consider two possible limitations:

- Bandwidth constraints that may arise due to limited radio interfaces;
- Bandwidth constraints that may appear between a host device and a security token.

7.3.3.1 Radio interface

While bandwidth is a mayor problem with 2G networks this kind of restriction will get less and less important as the 3G and certainly the 4G mobile networks will offer broad bandwidth-capacities that will allow a fast and effective processing of PKI-processes. The only problem in the PKI-context that could arise as an exception to the rule is again the download of large CRLs, especially if no real caching mechanism is implemented and the operation is time-critical.

7.3.3.2 Hosted Security Token

The bandwidth-restrictions between a security token and it's host (e.g. a mobile phone) can be more limiting, especially if one thinks of scenarios like the decryption of large amounts of content, say movies, where the encryption is done block-wise with many different session keys. The limited channel between a smart card and the reader could be such a bottleneck. New kinds of token, like in the fixed network environment USB-token, will offer better capabilities.

7.3.3.3 Relevant PKI-Processes

As already mentioned in section 7.2.4, certificate validation will be the crucial operation to consider. Two general approaches will have to be taken into account: the use of CRLs and online verification.

CRLs: Revocation-Lists can grow very large. Therefore the time to download them for the verification of a certificate, especially in long certificate-chains, may take too long for the client. Possible measures to reduce this problem could include the following:

- *Limit the size of CRLs* (if they exceed a certain size the CA-Keys have to be revoked and all currently valid user-certificates have to be exchanged. This solution obviously contains considerable organisational overhead).
- *Introduction of CRL-Caching-Mechanisms in the devices*. If a client uses certain CRLs frequently, caching them will offer a major advantage.
- *Introduction of distributed CRLs*. A PKI-provider will not only host one CRL but several CRLs, e.g. an own CRL for all certificates within a certain range of serial numbers. Of course, a reference of the appropriate CRL has to be given in the certificate.

Online Verification: Online Verification has been a topic for standardisation work for several years. IETF PKIX-RFC 2560 [26] specifies the Online Certificate Status Protocol (OCSP). The big advantage of this approach is that CRLs are no longer necessary, as the principle of OCSP is to contact a trustworthy entity every time a certificate has to be verified. Clearly, this kind of verification is more up-to-date than verification with CRLs. The disadvantages are that the user always has to be online to verify a certificate, and the CA may have an enormous overhead in signing all the responses it has to generate. Online verification has not yet achieved market penetration.

7.4 Interoperability issues

Interoperability of PKI solutions involves the following issues [12]:

- The ability for the verifier of an electronic signature to judge the conditions under which the signature was created.
- The ability for applications to exchange and operate with each other's certificates.
- In scenarios involving multiple PKIs, the use of cross certification and/or user trust lists.
- The use of certificate revocation solutions.

The first issue is addressed in the following section “Signature Policies”, whereas the other issues are raised in the section “Certificate Translation”.

7.4.1 Signature Policies

In case of electronic signatures, it is necessary to always use the same set of rules when evaluating an electronic signature. That means, that the conditions agreed by the signer at the time of signing have to be indicated to the verifier and to any arbitrator. These conditions or rules are captured in the *signature policy*, which must be unambiguously available to all parties. Hence, the following requirements must be fulfilled:

- A clear definition of the signature policy must be available to all parties
- The signature policy must be unambiguously recognisable by all parties.

A signature policy may be issued, for example, by a party relying on the electronic signatures and selected by the signer for use with that party. Alternatively, a signature policy may be established through an electronic trading association for use amongst its members. In any case, both signer and verifier must always use the same signature policy.

Before signing a document, a signer must make sure that he is using the right signature policy. Equally, when verifying an electronic signature, a verifier must make sure that he is applying the right signature policy. A specific *security service* could provide this assurance as a combination of two basic security services, namely a data origin authentication service and an integrity service.

A signature policy must include the following fields:

- A *unique identifier* or unambiguous identification of the signature policy.
- The name of the *signature policy issuer.*
- The *issuing date* of the signature policy.
- The *field of application* of the signature policy (expected application and corresponding conditions).
- A signature validation policy (technical rules in order to process the electronic signature).

For more information on security policies, see [67].

7.4.2 Certificate Translation

Digital certificates play a major role in any PKI. There will be situations where it is desirable to use a certificate in a different context. However, the corresponding certificate formats are often not compatible. For example:

- There are many different types of certificate, e.g. X.509, WTLS, ANSI X9.68, and SPKI certificates (see Annex). Applications designed to use one type of certificate usually do not work very well with a different type of certificate. However, if certificates are used for the same purpose, it would be desirable to use the certificate of one application in another.
- Although many PKIs use standard formats such as X.509, the supported extension fields may not be compatible. Different CAs may issue certificates carrying extensions with the same purpose but with different names, or certificates carrying extensions with the same name but with different purposes.

In such cases, it would be useful to make changes to existing certificates employing a *certificate translation service*, which makes changes to the structure and/or content of a certificate. The value of the public key is the only field in a certificate that may not be altered. A *certificate translation server* (CTS) is a server that offers a certificate translation service to clients. For more information on certificate translation, see [6].

Note that there are a number of open questions regarding the feasibility and use of certificate translation services. Perhaps the most problematic is the trust model to be used. Where the two certificates use compatible policy identifiers, trust issues may be relatively simple to deal with, but in many cases different means will be used to indicate the policy under which the certificate is issued (including implicit policies). Hence the provider of the CTS may need to do more than examine the fields of a certificate. For example, the CTS may also need to consider the CPS (Certification Practice Statement) of the CA which issued the certificate, and find ways to encode relevant information from this CPS into the translated certificate.

8 Requirements on PKI-involved parties

The aim of this chapter is to identify the relevant parties involved in the provision of PKI-services as well as the parties using these services, and to define their requirements and their roles. Therefore we will match the parties with the corresponding roles in the PKI-environment and evaluate the impacts on the requirements from the previous sections of this document.

8.1 'Mobile Network' Operators

The network operator plays a central role in the m-commerce PKI-scenarios and services. There are several reasons why this is the case:

- The network operator has the contracts with the customers and therefore a database with authentic customer-data. For data privacy reasons only he is able to use this database in PKI-scenarios. So registration in PKI-processes may be a lot easier for him, than for other parties.
- The network operator bills his customers and can so handle the billing for third parties as well.
- In 2G and 3G systems, the network operator distributes the SIM (Subscriber Identity Module) and has therefore authentic and unique identification-mechanisms by making use of individual (symmetric) keys stored on the SIM.
- Because the subscribers and mobile network operators share a secret key, and also because the subscribers have to trust their operator to produce correct bills, there is already a special and far-reaching trust relationship between operators and customers, which could be utilised also in mobile commerce scenarios.

8.2 Manufacturers of mobile devices

Another major role for PKI in m-commerce is played by the manufacturers of mobile devices.

- They can influence the capabilities of the devices concerning the use of PKI-mechanisms and formats.
- The manufacturers are necessary to introduce root-instances to the devices, so that users can work with PKI-mechanisms in an easy way.

PKI-structures to handle local execution environments, like MExE have to be implemented by the manufacturers.

8.3 Trust-service providers

Third-party trust-service-provider may be integrated in m-commerce scenarios.

- They may host PKI-solutions for network-provider. In this constellation roots of the trust-service-provider may be pre-installed, so that authentic distribution of the roots is not a problem. The trust-service-provider may use the network-provider's database to authenticate the user more easily. Lawful aspects about the protection of data have to be considered here.
- They may offer independent PKI-services. Initial registration in this scenario is obviously more difficult, as the provider and the user usually have no former relationship, so that the trust-service-provider has to identify the user from scratch. If the trust-service-provider is completely independent from the network-provider and the manufacturer another question arises: how will he get his roots distributed authentically?
- Besides the generation of certificates the trust-service-provider may also offer additional services like timestamping, certificate checking, etc.

8.4 The user

The user plays the central role in PKI-scenarios in m-commerce as he is the party that decides about the rise or the fall of such services. The following arguments underline this statement:

- The user is the one who buys the mobile devices. Therefore the critical mass of users to make mobile PKI-services succeed will only be reached if the users accept the devices of a certain kind.
- PKI-services themselves will mostly be liable to costs. That means that a user must see the added value when he is using PKI-services.
- The user not only decides which technical solutions he will accept, but, more importantly, he decides which issuing party he will rely on.

8.5 Third party application providers

The importance of third party providers is not quite clear at the moment. It depends on what services the network-providers will offer. If they cover a broad range, the importance may be low; on the other hand one can imagine scenarios, where a PKI-service is offered completely by a third party, that means independent from the network provider. There may also be certain scenarios where network operators act as trusted mediators between users and third party application providers.

8.6 Provider for mobile services (not owning a network)

The role of this party is still to be clarified. As such a provider is not the SIM-issuing party it is certainly less powerful than a ‘full’ provider.

9 PKI issues for future mobile systems

The purpose of this final section of D04 is to highlight those areas of PKI technology and application which, whilst being of importance to future mobile systems, are not well developed. We start by briefly reviewing those areas of PKI technology that would appear to be relatively mature, and hence not requiring major additional research. We then go on to consider the more problematic areas, in which new research is of great importance to future mobile systems.

9.1 PKI technologies not requiring major new research

Probably the most fundamental part of any PKI is the public key certificate. Public key certificates were first standardised as long ago as 1988 in the first version of ITU-T X.509 (=ISO/IEC 9594-8). Whilst a number of shortcomings of this original certificate format have been identified, these have been rectified in subsequent revisions of the standard. Hence it is seems reasonable to assert that the issue of certificate formats is not the most pressing research issue relating to the use and management of PKIs. Of course, issues do remain with certificate formats, notably issues related to the size of X.509 certificates, but with the inevitable increase in storage and processing power of all computing devices (including personal tokens such as smart cards) such issues are probably only of short term significance.

Similar remarks can be made about the format of other data structures associated with public key certificates, notably Certificate Revocation Lists (CRLs). Whilst the distribution of such lists remains a potential issue, in particular because of bandwidth and timeliness issues, the format of such lists has evolved in parallel with the development in the certificate format itself, and this area is probably unlikely to see any major new developments. Similar remarks also apply to the certificate status protocols such as OCSP. While the exact form of such protocols remains a subject of development work in the relevant standards bodies, the basic concepts now seem relatively mature. Again, where the problems really lie (as discussed below) are in managing the complexity and trust issues that arise.

The third and final area we mention here relates to the cryptographic primitives underlying PKI, namely asymmetric cryptographic techniques (and supporting algorithms such as hash-functions). Whilst new techniques continue to be developed, there is already a well-established and widely used set of primitives, many of which are the subject of international standards (see, for example, [38], [41], [48], [49], [50], [65], [66], [68]). This means that, whilst research in this area will continue, its success is not critical for the management and use of PKIs.

9.2 Problematic areas

9.2.1 Complexity of PKI for limited devices and for small trust models

A global PKI such as that provided by a large public CA such as VeriSign potentially allows two entities from anywhere in the world, who do not have a pre-existing trust relationship, to establish security services such as an SSL session, digitally signed and encrypted e-mail transfer, or the transmission of digitally signed executable code. This is valuable security connectivity, but the connectivity that public key cryptography can potentially provide comes at a price – complexity. Aspects of this complexity are examined below.

Below we refer to an entity using the public keys of other entities to verify communications from that other entity, as the *verifying entity*.

PK techniques are computationally more intensive than secret key techniques (especially hashing), even purportedly more efficient techniques such as those based on the use of ECC.

If a public key, which might be used to establish the authenticity and integrity of communication is to be transmitted over open networks, this itself must be transmitted with authentication and integrity. Some form of authentication is required to enable authentication!

If certificates are used to transmit the public key to the verifying entity, then the following requirements apply:

- the provision of root and intermediate certificates at the entity;
- a reliable, resilient (and ideally, one that cannot be changed by the user) time source at the verifying entity to verify that the certificate has not expired;
- in some cases, long certificate chains are needed, which require extra bandwidth and processing;
- the client should ideally have access to revocation status information on the certificate.

These requirements are all significant for limited devices.

Further, an entity seeking to obtain a certificate for itself must go through a reasonably intensive registration procedures in order to gain a certificate from a public CA. This is because it is seeking to authenticate itself to an entity with which it has no existing trust relationship, that is, the RA of the public CA.

These requirements occur because the PKI must support communications between, potentially, any two entities in the world. It may be possible that some or all of these requirements could be dispensed with for the use of PK within a smaller environment, such as a PAN or ad hoc network or small community of users that know each other well. However, it is interesting to note that:

- A PAN or ad hoc network is composed of a small number of entities in close physical proximity. Therefore the features of PK which allow an entity to authenticate itself to effectively anyone in the entire world, over open networks, do not seem to be necessary.
- A PAN or ad hoc network may be newly formed, and the components will therefore may not have any pre-existing trust relationships with each other. In this case, the fact that PK allows authentication to entities which one is not in an existing trust relationship with is of advantage.

There may be some way of balancing these two facts, and this will be examined in further deliverables.

9.2.2 The collision of trust models and commercial models

It is also interesting to note that one key advantage of PK techniques, namely of being able to provide security services to entities with whom there is no existing trust relationship, could in certain circumstances actually be a **disadvantage**. One example might be in the context of authorisation of significant functionality (e.g. downloaded code, or access rights to a house). Where PK is being used for securing an SSL session to a web site that one might never visit again, the easy connectivity is an advantage. However, where the functionality being enabled by PK is significant, it would make sense that this **cannot** be given to a potentially large number of entities with which there is no existing trust relationship but in fact should only be possible for a small number of already trusted entities. This tension is being seen in the standardisation of MExE and signed content in WAP – PK, with all its open and easy authorisation is the base technology, but concerned parties such as operators, are attempting to **cut down** the number of entities that can send authorisation messages.

What is occurring here is a collision between trust and commercial models. Public CAs want to issue as many legitimate entities as possible. However, users and perhaps entities that like to think they “own” the user, such as their operator, may want to limit the number of entities that are issued certificates given significant functionality with regard to the user.

9.2.3 Revocation of public keys and public key certificates

Whilst the use of CRLs is a well-established means to distribute revocation information, and the format of CRLs is not a major issue, the use of CRLs gives rise to major issues. These issues mainly relate to bandwidth and timeliness.

- *Bandwidth:* If a CA issues a large number of certificates, then even if a relatively small percentage of certificates are revoked CRLs can become quite large. The routine distribution of such CRLs to all clients of a PKI can become a formidable task, especially in bandwidth-limited environments.
- *Timeliness:* The major limitation of the use of CRLs is that, if a new CRL is issued every *t* seconds, then a newly revoked certificate may not be in the current CRL for up to *t* seconds. Hence *t* should ideally be as small as possible; however, for bandwidth and complexity reasons, it is typically infeasible to make *t* very small, which means that there is always a window of opportunity for fraudulent behaviour.

These issues have given rise to a number of alternative approaches, which we can divide into two main categories. (For a more detailed discussion see Section 4.7).

- *CRL enhancements*. This refers to modifications that can be made to try and ameliorate the practical difficulties encountered with the use of CRLs. One example is provided by the use of 'delta CRLs', i.e. a CRL consisting only of new additions to a previously issued CRL. Typically a series of delta CRLs will be issued, followed by a new complete CRL at less frequent intervals.
- *CRL alternatives*. This refers to approaches that do not involve CRLs at all. The foremost example are the certificate status protocols, where a Trusted Third Party will provide online information regarding the current validity of a certificate.

Both approaches have significant problems associated with them, and it is not clear exactly how this technology will evolve. For example, it is by no means clear which technology is most appropriate for the future mobile environment, and indeed whether new technology is required.

9.2.4 Interoperability of public key certificates and PKIs

In homogeneous environments, e.g. closed networks devised for specific purposes such as certain banking networks, use of PKI becomes relatively simple, since rules can be issued limiting certificates to a single type and profile, and defining a single policy for their use. However, in large heterogeneous environments, different CAs may issue certificate in different formats and/or conforming to different policies and profiles. Users may nevertheless need to verify certificates in a form different to that created by their 'own' CA(s).

Finding solutions to this problem is a major challenge, particularly with regard to the use of different profiles for different X.509 certificates (see also Section 9.2.6). One possible solution involves certificate translation services – see Section 7.4.2.

9.2.5 PKI-based authorisation issues

A further issue closely related to that of interoperability concerns the issue of the acceptability of certificates. In heterogeneous environments, even if the certificate profile is widely agreed, relying parties may be obliged to process certificates using a variety of policy identifiers. Of course, one way of handling this is by simply equipping relying parties with tables of which policy identifiers are acceptable for which purposes. However, such an approach will inevitably give rise to compatibility problems as new CAs with new policies come into the picture.

One way in which this problem might be solved is through the automatic processing of policy information, either be relying parties or by TTPs offering such services. For further discussion of this see the paper of Pinkas, [67].

9.2.6 The use of X.509 extensions

There are a number of major disadvantages with using PK that relate to the use of extensions. These are given below:

1. If, through desire for increased support, an attempt is made to standardise a new extension, global, multipurpose standards (e.g. [20], [58]) must be changed to meet the requirements of potentially a single purpose application.

2. Private extensions cannot be easily defined. Public key specifications ([20], [58]) state that private extensions can be used but in practice this is frowned upon, as it is said to be a barrier to interoperability, in spite of the fact that the criticality functionality should allow interoperability with applications that do not understand the new extension.
3. If attempts are made to standardise the use of existing extensions, a standard extension could end up having different meanings in different contexts, which can cause confusion for standards representatives and among implementers with consequent inter-operability problems. Alternatively, the existing meaning is used, with a consequent constraint put on the options for those specifying functions. In all cases, standards personnel in one forum require knowledge of how the extension is used in all forums, and such personnel are often few in number.
4. There are a large number of extensions, and again, there is the issue of the lack of people who understand all the extensions and how they should be used, and therefore how interoperable products supporting extension really are to be implemented.
5. Extensions, ironically, do not provide an easily extensible framework, as (see (1) above) global standards must either be changed (if a new extension is standardised) or inter-operability risked or new functionality constrained (if an existing standard extension is used).

   It should be noted though, that even for techniques with extensibility built in, for instance XML, extensibility is not guaranteed, as existing implementations will not, unless action is taken, understand a new "extension" and some sort of upgrade will be required.

   This upgrade could be by done by defining the extension's significance using an already existing "functionality language" present in all clients. Taking this method to its logical conclusion, the existing "functionality language" could be that of a programming language itself and new software to interpret the extension would be downloaded. This, of course, requires an already existing way to download new code that must be in all clients. Java is a language that seems to lend itself to the addition of functionality and meaning in small amounts as required here.

   At the very least, the extensibility mechanism should be used such that existing clients will not malfunction if an unknown "extension" is encountered. This is the intention behind the criticality setting in [20] and [55].
6. If the authorisation information is left outside the certificate, as an alternative to using a new extension, then there is the danger that this is separated from the certificate, and so loses its significance and security. Further, the possibility of using the defined method of propagating privileges and authorisation down a certificate chain that PK provides, is lost.
7. In order to avoid intermediate CA allocating new authorisation privileges to end entities, some implementations (whose designers believe intermediate CAs cannot be wholly trusted) sometimes require that an EE certificate can only contain an extension if the extension is contained in certificates higher up the certificate chain. The specification of a new extension may therefore require the re-issuing of intermediate CA certificates, or at worst case, the re-issuing and re-provisioning of a root CA certificate.

Extensions were defined to limit the purposes for which a certificate may be used. This they have done to some extent but have also been a frequent source of inter-operability problems. What is required is a framework for defining certificate purposes that is truly extensible and interoperable.

9.2.7 Non-compliant providers and users of PKI

Because use of PKI technology has often preceded the development of the necessary standards, and also because of the lack of desire by some manufacturers to adhere to standards, there are a number of major providers of PKI technology who use non-standard features. One example is discussed in Section 9.2.8 immediately below. Where such technology is commonplace (e.g. in web browsers), other PKI technology suppliers may be obliged to adopt these non-standard features. This can then give rise to major interoperability problems.

9.2.8 The effect of certificate lifetimes

The interaction of profiles and client implementations with certificate lifetimes is illustrated by SSL. The SSL v3.0 specification required use of X.509 certificates but did not specify a profile. It seems that many certificates, which were of course not compliant to [20], were issued. In particular, private extensions were introduced to allow use of 128 bit cipher algorithms. As many of these certificates have lifetimes considerably longer than product lifecycles, certificate processing applications released after approval of [20] must still support the older, non-compliant certificates or, as is more often the case, just say that certain extensions are “unknown”.

A more general example is that, [20] defines that certificate processing applications must still support names in TeletexString format even though it is, relatively speaking, an old and defunct format, because, it is believed, there are still certificates with names in this format. Therefore, any specification of certificate processing must take into account the fact that certificates generated in accordance with the specification may be in existence for a considerable period of time.

Annex A Certificate types

A.1 The development of public key certificate standards

The first work on developing PKI standards predates the term PKI by some years. What has become known as the X.509 certificate format was first standardised in 1988, [56], as part of the first edition of the ITU-T X.500 directory services recommendations (note that they were then referred to as CCITT recommendations). The first edition of an aligned ISO/IEC standard, ISO 9594-8, was published a couple of years later. Two subsequent versions of aligned ITU-T recommendations and ISO/IEC standards have been subsequently published, [57], [58], and to avoid confusion the three different certificate formats defined in these documents are referred to as X.509 versions 1, 2 and 3 certificates.

The original work on X.509 was performed as part of the development of the X.500 directory series recommendations. The main initial ‘customer’ for the standardised public key certificates were the parallel X.400 series of recommendations specifying the operation of an email system. The 1988 version of the X.400 standards incorporated a large range of security features, the key management for which was based round the use of X.509 certificates. Interestingly, while the X.400 recommendations have hardly set the world alight, the X.509 public key certificate format dominates the field.

After the publication of the first edition of the X.509 recommendation, the next main customer for the X.509 certificate format was again a secure email system – this time the Internet Privacy Enhanced Mail (PEM) system, [15]. This system again used X.509 certificates as the basis of its key management – however, a number of additional certificate features were required by PEM which were incorporated into the X.509 version 2 certificate format, [57]. Subsequent growing interest in deploying X.509 PKIs revealed the need for further additions to the certificate format, and these have been incorporated into the version 3 certificate format, [37], [58].

A.2 X.509 certificates

We now consider in a little more detail the main elements in a certificate constructed according to the X.509 version 3 standard, [37], [58]. The X.509 certificate format is specified in a language called Abstract Syntax Notation One (ASN.1), [59], [60], [61], [62]. ASN.1 is widely used for the specification of ITU-T and ISO (and other standard) communication protocols. The purpose of ASN.1 is to have a standardised and platform independent language with which to express data structures, and to have a standardised set of rules for the transformation of values of a defined type into a stream of bytes. This stream of bytes can then be sent on a communication channel set up by the lower layers in the stack of communication protocols, e.g. TCP/IP, or encapsulated within UDP packets. As a result, two different applications written in two completely different programming languages running on different computers with different internal representations of data can exchange instances of structured data types. This frees the programmer from a great deal of work, since no code has to be written to process the transport format of the data.

When the first ITU-T recommendation on ASN.1 was released in 1988, it was accompanied by the *Basic Encoding Rules* (BER) as the only option for encoding. BER is a somewhat verbose protocol. It adopts a so-called TLV (type, length, value) approach to encoding, in which every element of the encoding carries some type information, some length information and then the value of that element. Where the element is itself structured, then the Value part of the element is itself a series of embedded TLV components, to whatever depth is necessary. In summary BER is not a compact encoding but is fairly fast and easy to produce.

The Basic Encoding Rules come in three variants: BER – which allows options for the encoder, DER (*Distinguished Encoding Rules*) – which resolves all options in a particular direction, and CER (*Canonical Encoding Rules*) – which resolves all options in the other direction, [63]. That is DER and CER are unambiguous, since there are no encoding options.

A more compact encoding is achieved with the *Packed Encoding Rules* (PER), [64], which were introduced with the revised ASN.1 recommendation in 1994. PER takes a rather different approach from that taken by BER. The first difference is that the T (Type) part is omitted from the encodings, and any tags in the notation are completely ignored. A second difference is that PER takes full account of the sub-typing information while BER completely ignores it. PER uses the sub-typing information, for example, to omit length fields whenever possible. In summary, use of PER results in compact encodings that require much more computation to produce than does BER.

The ‘top level’ X.509 v3 certificate syntax is as below (note that this syntax is the same for all three versions of the X.509 certificate). For signature calculation, the certificate is encoded using the ‘tag, length, value’ ASN.1 distinguished encoding rules (DER), [63].

```
Certificate  ::=  SEQUENCE  {
        tbsCertificate       TBSCertificate,
        signatureAlgorithm   AlgorithmIdentifier,
        signatureValue       BIT STRING  }
```

The `signatureValue` field contains a digital signature computed upon the ASN.1 DER-encoded `tbsCertificate`. That is, the ASN.1 DER encoded `tbsCertificate` is used as the input to the signature function. This signature value is then ASN.1-encoded as a `BIT STRING` and included in the Certificate' s signature field.

The `signatureAlgorithm` field contains the identifier for the cryptographic algorithm used by the CA to sign this certificate. An algorithm identifier is defined by the following ASN.1 structure.

```
AlgorithmIdentifier  ::=  SEQUENCE  {
      algorithm         OBJECT IDENTIFIER,
      parameters        ANY DEFINED BY
            algorithm OPTIONAL  }
```

The algorithm identifier is used to identify a cryptographic algorithm. The `OBJECT IDENTIFIER` component identifies the algorithm (such as DSA with SHA-1). The contents of the optional parameters field will vary according to the algorithm identified. This field must contain the same algorithm identifier as the signature field in the sequence `tbsCertificate`.

The `tbsCertificate` field contains the names of the subject and issuer, a public key associated with the subject, a validity period, and other associated information. The `tbsCertificate` may also include extensions (the name given to the additional fields introduced into version 3 X.509 certificates). The ASN.1 structure is as follows.

```
TBSCertificate  ::=  SEQUENCE  {
  version         [0]  EXPLICIT Version DEFAULT v1,
  serialNumber         CertificateSerialNumber,
  signature            AlgorithmIdentifier,
  issuer               Name,
  validity             Validity,
  subject              Name,
  subjectPublicKeyInfo SubjectPublicKeyInfo,
  issuerUniqueID  [1]  IMPLICIT UniqueIdentifier
      OPTIONAL,  -- If present, must be v2 or v3
  subjectUniqueID [2]  IMPLICIT UniqueIdentifier
      OPTIONAL,  -- If present, must be v2 or v3
  extensions      [3]  EXPLICIT Extensions OPTIONAL
                     -- If present, must be v3
  }
```

The `Version` field describes the version of the encoded certificate. The default value is v1 (in which case this field can be omitted).

The serial number is an integer assigned by the CA to each certificate. It must be unique for each certificate issued by a given CA (i.e., the issuer name and serial number identify a unique certificate). This is especially useful when constructing Certificate Revocation Lists (CRLs), where the serial number can be used to identify the certificate being revoked. The syntax of CRLs is also defined in X.509.

```
      CertificateSerialNumber  ::=  INTEGER
```

The certificate validity period is the time interval during which the CA warrants that it will maintain information about the status of the certificate. The field is represented as a `SEQUENCE` of two dates: the date on which the certificate validity period begins (`notBefore`), and the date on which the certificate validity period ends (`notAfter`).

```
      Validity ::= SEQUENCE {
              notBefore      Time,
              notAfter       Time }
```

The `SubjectPublicKeyInfo` field is used to carry the public key and identify the algorithm with which the key is used. The algorithm is identified using the `AlgorithmIdentifier` structure.

```
      SubjectPublicKeyInfo  ::=  SEQUENCE  {
              algorithm            AlgorithmIdentifier,
              subjectPublicKey     BIT STRING  }
```

The `issuerUniqueID` and `subjectUniqueID` fields may only appear if the X.509 certificate version is 2 or 3. They are present in the certificate to handle the possibility of reuse of subject and/or issuer names over time.

The Extensions field may only appear if the X.509 certificate version is 3. If present, this field is a `SEQUENCE` of one or more certificate extensions. The extensions allow the encoding of policy information within a certificate.

```
      Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension
      Extension   ::=  SEQUENCE  {
              extnID      OBJECT IDENTIFIER,
              critical    BOOLEAN DEFAULT FALSE,
              extnValue   OCTET STRING  }
```

There are a large number of standardised extensions. The standard also allows implementers to define their own extensions. Some of the more important standardised extensions are as follows.

- **Key usage**. The key usage extension defines the purpose (e.g., encipherment, signature, certificate signing) of the key contained in the certificate. The usage restriction might be employed when a key that could be used for more than one operation is to be restricted.
- **Certificate policies**. A *certificate policy* is a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements. For example, a particular certificate policy might indicate applicability of a type of certificate to the authentication of electronic data interchange transactions for the trading of goods within a given price range.
- **Subject alternative name**. The subject alternative names extension allows additional identities to be bound to the subject of the certificate. Defined options include an Internet electronic mail address, a DNS name, an IP address, and a uniform resource identifier (URI). Other options exist, including completely local definitions. Multiple name forms, and multiple instances of each name form, may be included. Whenever such identities are to be bound into a certificate, the subject alternative name (or issuer alternative name) extension must be used.

A.3 The PKIX certificate profile

As should be clear from the description above, the X.509 certificate extensions are very broad in their applicability. In order to develop interoperable implementations of X.509 v3 systems for Internet use, it is necessary to specify a profile for use of the X.509 v3 extensions tailored for the Internet.

The goal of RFC 2459 [20], [21] is to facilitate the use of X.509 certificates within Internet applications for those communities wishing to make use of X.509 technology. Such applications may include the WWW, electronic mail, user authentication, and IPsec. In order to remove some of the obstacles to using X.509 certificates, RFC 2459 defines a profile to promote the development of certificate management systems, the development of application tools, and interoperability determined by policy. As such, RFC 2459 addresses the following issues:

- Format and semantics of certificates and certificate revocation lists for the Internet PKI,
- Procedures for processing of certification paths,
- Encoding rules for popular cryptographic algorithms.

Some communities will need to supplement, or possibly replace, this profile in order to meet the requirements of specialised application domains or environments with additional authorization, assurance, or operational requirements. However, for basic applications, common representations of frequently used attributes are defined so that application developers can obtain necessary information without regard to the issuer of a particular certificate or certificate revocation list (CRL).

The PKIX certificate profile will operate in a wide range of communication topologies, and will therefore support users without high bandwidth, real-time IP connectivity, or high connection availability. In other words, the profile recognises the limitations of the platforms these users employ.

In terms of certificate revocation, the PKIX certificate profile specifies the X.509 v2 CRL (certificate revocation list) format in order to facilitate interoperable implementations from multiple vendors.

A new version of RFC 2459 [21] was published in July 2001 and replaces the old version [20] of January 1999. Major changes are as follows:

- Chapter 6, Certificate Path Validation, has been extended. The new version contains a very elaborate description of “Basic Path Validation”, “Extending Path Validation” and “CRL Validation”.
- The chapter on Algorithm Support (Chapter 7 in [20]) has been excluded from the document. The procedures for identification and encoding of public keys are now defined in [4].
- The 1994 ASN.1 syntax is not supported anymore in the Appendix.

The extensions defined for X.509 v3 certificates provide methods for associating additional attributes with users or public keys and for managing the certification hierarchy. The X.509 v3 certificate format also allows communities to define private extensions to carry information unique to those communities. Each extension in a certificate is designated as either *critical* or *non-critical*. A certificate using system must reject the certificate if it encounters a critical extension it does not recognise; however, a non-critical extension may be ignored if it is not recognised.

RFC 2459 identifies a large number of standard certificate extensions:

- *Authority key identifier extension* – provides a means of identifying the public key corresponding to the private key used to sign a certificate. This extension is used where an issuer has multiple signing keys (either due to multiple concurrent key pairs or due to changeover).
- *Subject key identifier* – provides a means for identifying certificates containing the particular public key used in an application. Where an end entity has obtained multiple certificates, especially from multiple CAs, the subject key identifier provides a means to quickly identify the set of certificates containing a particular public key.
- *Key usage* - defines the purpose (e.g., encipherment, signature, certificate signing) of the key contained in the certificate. The usage restriction might be employed when a key that could be used for more than one operation is to be restricted. For example, when an RSA key should be used only to verify signatures on objects other than public key certificates and CRLs, the digitalSignature and/or nonRepudiation bits would be asserted. Likewise, when an RSA key should be used only for key management, the keyEncipherment bit would be asserted.
- *Private key usage period* – allows the certificate issuer to specify a different validity period for the private key than the certificate. This extension is intended for use with digital signature

keys. This profile recommends against the use of this extension. CAs conforming to this profile must not generate certificates that include a critical private key usage period extension.

- *Certificate policies* – contains a sequence of one or more policy information terms. In an end entity certificate, these policy information terms indicate the policy under which the certificate has been issued and the purposes for which the certificate may be used.
- *Policy mappings* – is used in CA certificates. It lists one or more pairs of OIDs; each pair includes an issuerDomainPolicy and a subjectDomainPolicy. The pairing indicates the issuing CA considers its issuerDomainPolicy equivalent to the subject CA's subjectDomainPolicy.
- *Subject alternative name* – allows additional identities to be bound to the subject of the certificate. Defined options include an Internet electronic mail address, a DNS name, an IP address, and a uniform resource identifier (URI).
- *Issuer alternative name* – associate Internet style identities with the certificate issuer.
- *Subject directory attributes* – is used to convey identification attributes (e.g., nationality) of the subject.
- *Basic constraints* – identifies whether the subject of the certificate is a CA and the maximum depth of valid certification paths that include this certificate.
- *Name constraints* – indicates a name space within which all subject names in subsequent certificates in a certification path must be located (only in CA certificates). Restrictions apply to the subject distinguished name and apply to subject alternative names.
- *Policy constraints* – can be used in certificates issued to CAs. The policy constraints extension constrains path validation in two ways. It can be used to prohibit policy mapping or require that each certificate in a path contain an acceptable policy identifier.
- *Extended key usage field* – indicates one or more purposes for which the certified public key may be used, in addition to or in place of the basic purposes indicated in the key usage extension field.
- *CRL distribution points* – identifies how CRL information is obtained.
- *Inhibit any-policy* – can be used in certificates issued to CAs. The value indicates the number of additional certificates that may appear in the path before any-policy is no longer permitted. For example, a value of one indicates that any-policy may be processed in certificates issued by the subject of this certificate, but not in additional certificates in the path.
- *Freshest CRL* – identifies how delta CRL information is obtained.

In addition, there are two extensions for use in the Internet Public Key Infrastructure. These extensions may be used to direct applications to on-line information about the issuing CA or the subject.

- *Authority information access* – indicates how to access CA information and services for the issuer of the certificate in which the extension appears. Information and services may include on-line validation services and CA policy data.
- *Subject information access* – indicates how to access information and services for the subject of the certificate in which the extension appears. When the subject is a CA, information and services may include certificate validation services and CA policy data. When the subject is an end entity, the information describes the type of services offered and how to access them.

A.4 WTLS certificates

This certificate format is used by the WTLS security protocol for authentication of WAP client and server during the establishment of a secure transport end-to-end communication session.

The server certificates authenticate the identity of a WAP server to visiting WAP clients in mobile user equipment, while the client certificates authenticate the identity of a visiting WAP client to a WAP server.

The WTLS certificate has the advantage of being very compact, easily implemented in code, and easily parsed.

The top level WTLS certificate syntax is as follows:

```
struct {
      ToBeSignedCertificate to_be_signed_certificate;
      Signature             signature;
} WTLSCertificate;
```

The signature field contains the digital signature computed upon the ToBeSignedCertificate and the signature algorithm. The signature algorithm can have the values ecdsa_sha, rsa_sha and anonymous, used to compute the digital signature. The value anonymous for the signature algorithm means the certificate is used to distribute the public key uncertified to setup a secure session without authentication.

```
struct {
      uint8                 certificate_version;
      SignatureAlgorithm    signature_algorithm;
      Identifier            issuer;
      uint32                valid_not_before;
      uint32                valid_not_after;
      Identifier            subject;
      PublicKeyType         public_key_type;
      ParameterSpecifier    parameter_specifier;
      PublicKey             public_key;
} ToBeSignedCertificate;
```

The parameter_specifier field specifies parameters relevant for the public key. The meaning of the other fields maps this of the X.509 certificate.

A.5 SPKI certificates

Simple Public Key Infrastructure (SPKI) is the product of the merging of the Simple Distributed Security Infrastructure (SDSI), proposed by Lampson and Rivest in 1996, and the Simple Public Key Infrastructure (SPKI), proposed by Ellison et al. Its main purpose is to provide authorisation rather than authentication. The sources of information for the present description are RFC 2693 [29], and the paper by Clarke et al. [7].

Simple Public Key Infrastructure (SPKI) works by authorising keys, *not* names from a fixed global name space. In SPKI there are two types of certificates: name certificates (name certs), and authorisation certificates (auth certs). A central concept in SPKI is that of a *local name*. It consists of a public key K and an identifier A. We will denote a local name by K A. The identifier A is not global; that is, for a public key K' different from K, the local names K A and K' A are distinct. We say that the local name K A is in the name space of K. There is also the notion of extended names in SPKI, which we omit in this description, since it will only complicate things. We also define a *term* to be either a public key or a local name.

A *name certificate* is a 4-tuple (K,A,S,V), where

- The issuer K is a public key. The certificate is signed by the secret key associated with K.
- The identifier A defines the local name K A.
- The subject S is a term
- The validity specification V provides information about the validity of the name cert. Typically, it contains a time period during which the certificate is valid.

In the discussion to follow, we will always assume that the certificates are valid. Non-valid certificates are ignored.

In order to give a meaning to name certs, we define the notion of the value (or meaning) of a term. The value of a term is a set of public keys. If S is simply a public key K then we define its value v(K) to be the singleton set containing K. If S is the local name K A, then given the certificate C=(K,A,S,V), we pose the condition v(K A) $\supset$ v(S), i.e., the certificate C contributes to the value of K A all the keys in v(S). Given now a term S, and a set of certificates **C**, the value v(S) is defined to be the smallest set of public keys, so that the above condition is satisfied. Intuitively, each term S is a "name" for a group of public keys v(S). It important to note, that the values of terms relative to a set of certificates can be computed efficiently by an algorithm given in [7].

An *authorisation certificate* is 5-tuple (K,S,D,T,V), where

- The issuer K is a public key. The certificate is signed by the secret key associated with K.
- The subject S is a term. Authorisation is granted to public keys in v(S).
- D is the delegation bit. When turned on, it gives permission to public keys in v(S) to further delegate to others the authorisation it is receiving via this certificate.
- The authorisation specification T, gives the specific permissions being granted.
- The validity specification V is the same as in a name cert.

In a system with authorisation control, access will be given to groups, and those groups are specified by "names". An Access Control List (ACL) can be thought of as one or more auth certs. If K_{issuer} denotes the public key of the issuer of the ACL, and the ACL consists of the auth cert (K_{issuer} , S, D, T, V), then any request for access that is signed be a private key corresponding to a public key in v(S) will be honoured. If furthermore, the delegation bit D is turned on, then any auth cert issued by a key K in v(S) will be treated as if it was an original auth cert in ACL issued by K_{issuer} .

Here is an example. Suppose that RHUL has a resource, which has access control: only partners in the SHAMAN project may access it. Let K_{RHUL} denote RHUL's public key. Then access control can be implemented as follows. RHUL issues one name certificate (K_{RHUL}, SHAMAN, K_P,V) for each partner. Here K_P is the public key of partner P, and V denotes the time span of the project. The certificates are then distributed to the partners. Then access control list consists of the auth cert (K_{RHUL} , K_{RHUL} SHAMAN, 0, T, V), where T specifies the type of access that is allowed. When partner P requests access to the resource, he/she submits the name cert (K_{RHUL}, SHAMAN, K_P,V) and signs the request by the secret key corresponding to K_P. According to the definition of the scheme, access will be granted to P if K_P is in v(K_{RHUL} SHAMAN), where the value is computed given the auth cert and the submitted name certs. Indeed, given the name certificate (K_{RHUL}, SHAMAN, K_P,V), the key K_P is in v(K_{RHUL} SHAMAN) and therefore access is granted. We note that in more complicated cases, the user may have to submit more than one name certs that he/she possesses.

A.6 Proprietary certificate formats

Apart from the international standard certificate formats, there are a number of other certificate formats defined for use in specific application domains. It is outside the scope of this report to list them all, but we mention one that is of some practical importance.

This is the 'EMV certificate', [8], defined in the Europay-MasterCard-Visa (EMV) standards governing communications between a payment smart card (e.g. a credit or debit card) and a merchant terminal. The main reason that EMV certificates were developed (as opposed to adopting X.509 certificates) was the need to minimise the length of certificates. In the card/terminal environment, both storage space and communications bandwidth are in short supply.

EMV certificate are encoded using a Tag-Length-Value technique, much as for BER; however, the number of fields is much less than for X.509. Moreover, the signature algorithm employed (following

ISO/IEC 9796-2, [38]) minimises data storage/bandwidth requirements by enabling as much data as possible to be recovered from the signature.

A.7 Underlying technology

When discussing standards governing the creation and format of certificates, it is also worth briefly mentioning standards for the cryptographic techniques employed as part of the certificate creation and verification processes. Fundamental to the creation of a public key certificate are two types of cryptographic function:

- a digital signature scheme, and
- a cryptographic hash-function (used almost invariably as part of a signature scheme).

Digital signature schemes are specified in a number of different standards, notably in IEEE P1363, [14], ISO/IEC 9796, [38], ISO/IEC 14888, [48], [49], [50], NIST FIPS PUB 186-2, [66], and PKCS #1, [68]. Hash-functions are also specified in a number of standards, including ISO/IEC 10118, [39], [40], [41], [42], and NIST FIPS PUB 180-1, [65].

References

[1] 3GPP TS 23.057 v4.1.0 (2001-03), Mobile Execution Environment (MExE); Functional Description Stage 2 (Release 4).

[2] ABA, *Digital signature guidelines: Legal infrastructure for Certification Authorities and electronic commerce*. American Bar Association, 1995.

[3] Baltimore Learning Centre, http://www.baltimore.com/library/whitepapers/acswp-hm.html.

[4] L. Bassham, R. Housley, and W. Polk, *Internet X.509 Public Key Infrastructure Representation of Public Keys and Digital Signatures*, draft-ietf-pkix-ipki-pkalgs-02.txt, March 2001.

[5] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelson, and T. Wright, *TLS Extensions*, draft-ietf-tls-extensions-00, June 2001.

[6] N. Borselius and C. Mitchell, *Certificate Translation*, in: Proceedings of NORDSEC 2000 - 5th Nordic Workshop on Secure IT Systems, Reykjavik, Iceland, 12/13 October 2000, pp.289-300.

[7] D. Clarke et al., *Certificate Chain Discovery in SPKI/SDSI*, Draft Paper, Nov 1999.

[8] EMV '96, *Integrated Circuit Card Specification for Payment Systems*. Version 3.1.1, May 31, 1998.

[9] ETSI TS 101 733, *Electronic Signatures Format*, V1.2.2, October 2000.

[10] S. Farrell, *An Internet Attribute Certificate Profile for Authorization*, IETF Internet Draft `draft-ietf-tls-ac509prof-00.txt`, January 2000.

[11] A. Frier, P. Karlton and P. Kocher, *SSL Protocol, version 3.0*, draft-freier-ssl-version3-02.txt, November 18, 1999. Available at http://home.netscape.com/eng/ssl3/draft302.txt.

[12] M. Getliffe, WP2N005 – *Overview of PKI Interoperability Issues*, EEMA pki Challenge, February 2001.

[13] V. Hassler, 'X.500 and LDAP security: A comparative overview'. *IEEE Network* Vol. 13 no. 6 (November/December 1999) pp. 54-64.

[14] IEEE P1363, *Standard Specifications For Public Key Cryptography*, 2000.

[15] Internet RFC 1422, *Privacy enhancement for Internet electronic mail, Part II: certificate-based key management* (by S. Kent), February 1993.

[16] Internet RFC 1777, *Lightweight Directory Access Protocol* (by Y. Yeong, T. Howes and S. Kille), March 1995.

[17] Internet RFC 1778, *The string representation of standard attribute syntaxes* (by T. Howes, S. Kille, W. Yeong, and C. Robins), March 1995.

[18] Internet RFC 2401: *Security Architecture for the Internet protocol*, November 1998.

[19] Internet RFC 2409: *The Internet Key Exchange*, November 1998.

[20] Internet RFC 2459, *Internet X.509 public key infrastructure – Certificate and CRL profile* (by R. Housley, W. Ford, W. Polk and D. Solo), January 1999.

[21] Internet RFC 2459 (revision), *Internet X.509 public key infrastructure – Certificate and CRL profile* (by R. Housley, W. Ford, W. Polk and D. Solo), eighth draft, July 2001.

[22] Internet RFC 2510, *Internet X.509 public key infrastructure – Certificate management protocols* (by C. Adams and S. Farrell), March 1999.

[23] Internet RFC 2511, *Internet X.509 certificate request message format* (by M. Myers, C. Adams, D. Solo and D. Kemp), March 1999.

[24] Internet RFC 2527, *Internet X.509 public key infrastructure – Certificate policy and certification practices framework* (by S. Chokhani and W. Ford), March 1999.

[25] Internet FFC 2559, *Internet X.509 public key infrastructure – Operational protocols – LDAPv2* (by S. Boeyen, T. Howes and P. Richard), April 1999.

[26] Internet RFC 2560, *X.509 internet public key infrastructure – Online certificate status protocol (OCSP)* (by M. Myers, R. Ankney, A. Malpani, S. Galperin and C. Adams), June 1999.

[27] Internet RFC 2587, *Internet X.509 public key infrastructure – LDAPv2 schema* (by S. Boeyen, T. Howes and P. Richard), June 1999.

[28] Internet RFC 2630, *Cryptographic Message Syntax* (by R. Housely), June 1999.

[29] Internet RFC 2693, *SPKI Certificate Theory*, (by C. Ellison et al.), September 1999.

[30] Internet RFC 2903, *Generic AAA Architecture*, August 2000.

[31] Internet RFC 2904, *AAA Authorisation Framework*, August 2000.

[32] Internet RFC 2906, *AAA Authorisation Requirements*, August 2000.

[33] Internet draft, *A PKIX Profile for IKE*, http://community.roxen.com/developers/idocs/drafts/draft-ietf-ipsec-pki-req-05.html.

[34] Internet draft, *Internet X.509 public key infrastructure – PKIX roadmap* (by A. Arsenault and S. Turner), March 2000.

[35] Internet draft, *Simple Certificate Validation Protocol (SCVP)* (by A. Malpani and P. Hoffman), June 2000.

[36] Internet draft, *Internet X.509 public key infrastructure – Time Stamp Protocol (TSP),* (by C. Adams, P. Cain, D. Pinkas, and R. Zuccherato), October 2000.

[37] ISO/IEC 9594-8: 1998 (3rd edition), *Information technology – Open Systems Interconnection – The Directory: Authentication framework*.

[38] ISO/IEC 9796-2: 1997, *Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Mechanisms using a hash-function*.

[39] ISO/IEC 10118-1: 2000 (2nd edition), *Information technology – Security techniques – Hash-functions – Part 1: General*.

[40] ISO/IEC 10118-2: 2000 (2nd edition), *Information technology – Security techniques – Hash-functions – Part 2: Hash-functions using an n-bit block cipher algorithm.*

[41] ISO/IEC 10118-3: 1998, *Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions*.

[42] ISO/IEC 10118-4: 1998, *Information technology – Security techniques – Hash-functions – Part 4: Hash-functions using modular arithmetic*.

[43] ISO/IEC 10181-2, *Information Technology – Open Systems Interconnection – Security Frameworks in Open Systems, Part 2: Authentication framework*.

[44] ISO/IEC 11770-1: 1996, *Information technology – Security techniques – Key management – Part 1: Framework*.

[45] ISO/IEC 13888-1: 1997, *Information technology – Security techniques – Non-repudiation – Part 1: General*.

[46] ISO/IEC 13888-3: 1997, *Information technology – Security techniques – Non-repudiation – Part 3: Mechanisms using asymmetric techniques*.

[47] ISO/IEC TR 14516: 2000, *Information technology – Security techniques – Guidelines on the use and management of Trusted Third Party services*.

[48] ISO/IEC 14888-1: 1998, *Information technology – Security techniques – Digital signatures with appendix – Part 1: General*.

[49] ISO/IEC 14888-2: 1999, *Information technology – Security techniques – Digital signatures with appendix – Part 2: Identity-based mechanisms*.

[50] ISO/IEC 14888-3: 1998, *Information technology – Security techniques – Digital signatures with appendix – Part 3: Certificate-based mechanisms*.

[51] ISO/IEC FDIS 15945, *Information technology – Security techniques – Specification of TTP services to support the application of digital signatures*, October 2000.

[52] ISO/IEC FCD 18014-1, *Information technology – Security techniques – Time stamping services – Part 1: Framework*, May 2001.

[53] ISO/IEC CD 18014-2, *Information technology – Security techniques – Time stamping services – Part 2: Mechanisms producing independent tokens*, June 2001.

[54] ISO/IEC WD 18014-3, *Information technology – Security techniques – Time stamping services – Part 3: Mechanisms producing linked tokens*, January 2001.

[55] ISO/IEC JTC1/SC29/WG11 N3943, *Intellectual Property Management and Protection in MPEG Standards*, January 2001, Pisa.

[56] ITU-T Recommendation X.509 (1988), *Information technology – Open Systems Interconnection – The Directory: Authentication framework*.

[57] ITU-T Recommendation X.509 (11/93), *Information technology – Open Systems Interconnection – The Directory: Authentication framework*.

[58] ITU-T Recommendation X.509 (08/97), *Information technology – Open Systems Interconnection – The Directory: Authentication framework*.

[59] ITU-T Recommendation X.680 (12/97), *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation*.

[60] ITU-T Recommendation X.681 (12/97), *Information technology - Abstract Syntax Notation One (ASN.1): Information object specification*.

[61] ITU-T Recommendation X.682 (12/97), *Information technology - Abstract Syntax Notation One (ASN.1): Constraint specification*.

[62] ITU-T Recommendation X.683 (12/97), *Information technology - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*.

[63] ITU-T Recommendation X.690 (12/97), *Information technology - ASN.1 encoding rules - Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*.

[64] ITU-T Recommendation X.691 (12/97), *Information technology - ASN.1 encoding rules - Specification of Packed Encoding Rules (PER)*.

[65] NIST FIPS PUB 180-1, *Secure hash standard*, April 1995.

[66] NIST PIPS PUB 186-2, *Digital signature standard*, January 2000.

[67] Denis Pinkas, *Using signature policies to verify e-signatures*. Presented at the ISSE 2000 Conference, Barcelona, 2000.

[68] PKCS #1, *RSA cryptography standard*, Version 2.1 (draft), September 1999.

[69] E. Rescorla, *SSL and TLS*, Addison Wesley, 2000.

[70] Shaman Deliverable D02, Version 1.0, 2001.

[71] Shaman Deliverable D03, Version 1.0, 2001.

[72] TeSSA, http://www.tml.hut.fi/Research/TeSSA/

[73] Internet draft: Wireless Extensions to TLS (17-Nov-2000): Draft-ietf-tls-wireless-00.txt.

[74] *WAP Signed Content*, WAP-233-SCONT-20010531-p, Prototype Version 31-5-2001, WAP Signed Content, Version 31-5-2001.

[75] WAP Transport Layer E2E Security Specification (Version 11-July-2000): WAP-187-TranportE2ESec-20000711-a.pdf.

[76] WAP WIM (Version 18-feb-2000), part security: WAP-198-WIM-20000218-a.pdf.

[77] WAP-161-WMLScriptCrypto-20010620-a, *WMLScript Crypto Library*.

[78] WAP WTLS (Version 18-feb-2000): WAP-199-WTLS-20000218-a.pdf

[79] Mobility Support in IPv6 (December 1998), Draft-ietf-mobileip-ipv6-13.txt.

[80] E-mail sent to WAP Security Group mailing list, 2000.

[81] *White Paper on the Secure Digital Music Initiative SDMI*, March 1999, Erlangen.

[82] 21.133, *Security Threats and Requirements*.