IST-2000-25350 - SHAMAN

Deliverable Number	**D07**
Deliverable Title	**Intermediate specification of PKI for heterogeneous roaming and distributed terminals**
Date of delivery	
Document Reference	**SHA/DOC/VOD/WP3/D07/1.0**

Contractual Delivery Date	**1-Mar-2002**
Actual Delivery Date of this version	**1-Mar-2002**
Editor	**Tim Wright, Vodafone**
Participant(s):	**Vodafone, Royal Holloway, Siemens ATEA, T-Nova, Nokia**
Workpackage	**WP3**
Est. person months	
Security	
Nature	
Version	**1.0**
Total number of pages	**51**

Abstract:

- The purpose of this report is to specify PKI architectures and techniques that can and, it is expected, will be used by other workpackages within SHAMAN in their final security architectures. Solutions to issues raised in previous PKI studies within SHAMAN are provided.

Keyword list:

PKI, mobile telecommunications

Executive summary

This document contains an analysis of PKI issues for which an investigation was explicitly requested by other Shaman workpackages (WPs) and also analyses that WP3 undertook on its own initiative.

With regard to the PKI issues for WP1 (secure heterogeneous access from mobile terminals to core networks) there is a major issue of mobile node (MN) to access network (AN) authentication. If this uses asymmetric methods, then the resulting requirements for PKI are simplest if the MN's certificate is issued by its "home domain". The PKI requirements are more complex if the AN issues the MN certificate, particularly if AN providers do not cooperate with each other. Further, issuance by the home domain potentially provides the MN with greater identity privacy towards the AN provider.

PKI can be used for payment mechanisms for AN access, but these do not provide methods for the establishment of a security association which is seen as a necessity. Therefore it is concluded that, until a final decision is made on the method of security association establishment to be used, analysis of PKI-based payment mechanisms must wait.

However, WP3 believe that WP1 PKI issues would benefit from further analysis and this will be contained in the final technical Shaman deliverable, D13, [1].

WP2 have requested WP3 to examine the provision of CA functions within a PAN, and without reference to a "global" PKI over a long range interface. The intra-PAN CA function has been termed the "personal CA". Two methods of providing personal CA functionality are given:

- "Traditional PKI", where identities are associated with randomly generated public keys in X.509 or similar certificates;
- Identity based PKI, where the public key of a device is the e-mail address or another identity by which the device can be publicly addressed.

The two methods are compared and our conclusion is that on the criteria used for comparison, the two schemes are approximately equal.

WP2 requirements for PKI for secure execution environments are examined. It is seen that there are outstanding problems here that require further analysis, but also that solutions, some of which are contractual and legal, are available. The problems mainly relate to the fact that it is felt that the party that will suffer most from unreliable authorisation of providers of executable code, that is, the network operator, has no control of which parties are so authorised, as this is done by manufacturers and third party Certification Authorities (CAs).

Generic PKI issues, not related to a particular WP, are also examined. We conclude that the Online Certificate Status Protocol (OCSP) is the best choice if client certificate revocation checking is required. We examine generic issues related to the use of asymmetric cryptographic techniques for authentication and authorisation. We further examine the pros and cons of linking authentication and authorisation, and conclude that although new standards work may be required, we should seek to promote methods where authentication and authorisation are provided using separate mechanisms, and not jointly using X.509 certificates with extensions, as is done presently. Finally, we consider issues associated with the implementation of PKI on devices with limited computational and communications capabilities, such as are likely to exist within the PAN.

Table of contents

List of contributors

Name		Affiliation	Email	Phone
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
Knospe	Heiko	T-Systems Nova GmbH (TNO)	heiko.knospe@t-systems.com	+49 6151 83 2033
Mitchell	Chris	Royal Holloway, University of London (RHUL)	c.mitchell@rhul.ac.uk	+44 1784 443423
Nyberg	Kaisa	Nokia	Kaisa.nyberg@nokia.com	+358 40 7038169
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
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 intermediate report is to specify PKI architectures and techniques that can, and are expected to, be used by other workpackages within SHAMAN in their final security architectures. Solutions to issues raised in previous PKI studies within SHAMAN are also provided. This deliverable presents the current state of WP3 analysis on the issues covered. There may be further work on these issues and this deliverable should be therefore be considered "intermediate", as the title suggests.

1.2 Contents of this report

In the previous WP3 deliverable, D4 [2], the security requirements listed by WPs 1 and 2 in D2 and D3 respectively were analysed for potential satisfaction of the requirements using asymmetric cryptographic techniques. Further, D4 identified a number of general PKI issues, for example, client revocation checking, which WP3 believed required analysis.

This deliverable is a development of the work done in D4 and in other workpackages. Specifically, there are three sources for the material in this deliverable:

- Analysis which has been done by WP3 in response to specific requests for asymmetric functionality in another workpackage. Section 3 is an example of this, being a response to a specific request from WP2.
- Analysis which is not in response to a specific request from another workpackage but where WP3 believes that analysis of WP1 and 2 requirements in D4 would benefit from some further analysis in D7. An example of this would be section 4, "Secure Execution Environment".
- Further analysis of the general PKI issues raised in D4, for instance, section 5.3, "Authorisation".

The sections are as follows:

Section 2 analyses the PKI requirements of WP1

Section 3 analyses the PKI requirements of WP2 in relation to the concept of a "personal CA", that is, a device within a Personal Area Network (PAN) that acts as a CA for the rest of the PAN. The provision of a personal CA by both "traditional" and identity-based schemes is examined and comparisons between the two made.

Section 4 analyses the PKI requirements of WP2 in relation to secure execution environments.

Section 5 contains WP3's "proactive" analysis of general PKI issues not conducted in response to a particular request from another Shaman WP. Client revocation checking, authentication, authorisation, and PKI issues for limited devices are examined.

1.3 List of abbreviations

Table 1 – List of abbreviations

Term	Definition
ASN.1	Abstract Syntax Notation 1
ASP	Application Service Provider
CA	Certification Authority
CCM	Certificate Configuration Message
CPS	Certification Practice Statement
CRL	Certificate Revocation List
DNS	Domain Name Service
DPD	Delegated Path Discovery
DPV	Delegated Path Validation
GPRS	General Packet Radio Service
HTTP	HyperText Transfer Protocol
IETF	Internet Engineering Task Force
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
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
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 Verification Protocol
SHA-1	Secure Hash Algorithm revision 1 – a standardised cryptographic hash-function
SIM	Subscriber Identity Module
SMS	Short Message Service

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 PKI for mobile access security

In this section we consider PKI requirements arising out of the work of Shaman/WP1. In particular we consider PKI requirements arising from possible public key based approaches to the provision of mobile access security. The analysis of Shaman/WP1 requirements is based on an analysis of [3].

The main issue considered in Section 3 of [3] is authentication between a Mobile Node (MN) and an Access Network (AN). Two main approaches are considered:

- subscription-based access, and
- access based on alternative means.

In both cases, the use of public key techniques has been considered, which brings associated PKI requirements. We review both applications of public key techniques, and consider the associated PKI requirements. We conclude with a summary of the main directions for future research to address the potential PKI requirements.

2.1 Subscription-based access

2.1.1 Possible use of public key techniques

There are two different public-key-relevant topics addressed within [3] which fall under this heading.

- In Section 3.1.1 of [3], the use of secret key techniques to provide security for BRAIN mobility is considered. Whilst most of this discussion is not public key relevant, subsection 3.1.1.8 on 'micromobility' considers securing communications within the BRAIN access network (BAN). Providing this communications security (using IPsec) will require key management, and it is suggested that this can be achieved using a 'BAN-local' PKI, enabling nodes to use certificates in IKE to negotiate security associations.
- Section 3.1.2 of [3] briefly reviews three public key based authentication protocols, designed to operate between an MN and an Access Network. In every case it is assumed that the MN has a long-term security context pre-established with a 'Home Domain' (and hence they are all 'subscription-based'). This security context could involve the production of public key certificates for the MN by the Home Network acting as a CA, or a CA operating with the Home Network. The three protocols have potentially different PKI requirements.

  Note that the choice of authentication protocol for Mobile Node/Access Network authentication is inextricably bound up with access privacy for the MN. Section 4.2 of [3] discussed security of the initial communication between the MN and the AR, in order to establish a security association. Subsection 4.2.3 is concerned with providing access privacy for the MN. Two options are considered for providing this privacy, both based on public key techniques.

  - The first option (see Section 4.2.3.1.1) provides MN identity privacy over the link between MN and AR. The key used for this encryption is derived from an asymmetric key agreement protocol based on the public key of the Access Network.
  - A second possibility (see Section 4.2.3.1.2) provides MN identity privacy from all parties including the Access Network. This is supported by encrypting the MN identity under the public key of the MN's Home Network. This relates closely to some of the issues considered in Section 2.1 above.

  Both options relate closely to the choice and design of the authentication protocol to be used between the MN and the AN.

We consider the PKI requirements associated with these two topics in the next two subsections.

2.1.2 PKI requirements for the BRAIN access network

To meet the potential BAN requirements for public key cryptography will require the distribution and maintenance of public key certificates within the BAN. The appropriate approach to dealing with this requirement will depend on the nature of the BAN. For example, is the BAN likely to consist of a large number of relatively small 'closed' groups of closely-related entities? Alternatively, will the BAN contain large collections of very loosely-connected entities who need to interact?

In the first case the problem will be relatively simple, since a small 'local CA' could probably be set up without too much difficulty. If the entities involved are closely related then they are likely to find it simple to agree on a single security policy governing key management. Indeed, if the number of entities involved is really small, then keys could be exchanged on a bilateral basis with no need for a PKI.

However, the second case has significant PKI implementation problems. If there is not a close connection between the various entities then it is likely that multiple security policies will be in force, and the levels of trust between entities will vary widely. This means that the PKI may become rather complex to operate and manage. Issues likely to cause problems include interoperability of certificates produced by different CAs under different policies, obtaining certificate chains, and obtaining certificate status information.

2.1.3 PKI requirements for MN/AN authentication

The PKI requirements for the authentication protocol will, to some extent, vary, depending on the authentication protocol in use. In particular, whether or not the AN is required to exchange certificates with the MN without mediation by the 'home network' (AAAH) of the MN has a major impact on the PKI design. We can therefore divide this discussion into two main cases, as follows.

- **Case 1**. In this case we assume that public key cryptography is primarily used between MN and its AAAH. This case is relatively simple since the AAAH can provide the AN and the MN with the certificates (and certificate chains) they require. In addition, this probably avoids most of the certificate revocation issues.
- **Case 2**. In this case we suppose that the MN and AN may need to conduct a public key authentication exchange without any assistance from the AAAH. In this case there needs to be a mechanism for both parties to be provided with certificate chains (and certificate status information). In such a case the PKI complexity is likely to be much higher, and many of the PKI issues discussed elsewhere in this report are relevant.

2.2 Access based on alternative means

2.2.1 Possible use of public key techniques

In Section 3.2 of [3] a variety of other methods are considered which could provide a basis for a secure payment for service between an MN and an Access Network. Some of those considered depend on public key techniques, and hence on a PKI. Some examples of public key based techniques are as follows.

- SET ('Secure Electronic Transaction') – a public key based protocol for securing e-commerce payments across the Internet;
- eCash – an electronic money scheme;
- MilliCent – another electronic money scheme;
- M-payment – a card-based m-commerce scheme.

However, it is not yet clear how effective these techniques can be in providing a cost-effective and efficient authentication service for the MN. The PKI issues associated with their use will help determine the practicability of their deployment.

Prior to use of a payment protocol, it will be necessary for the MN and the Access Network to set up a secure association (involving some type of authentication protocol). A key (or keys) will be needed to set up the security association, and these keys could be provided through the use of public key techniques, backed up by some kind of PKI. (These issues have been covered in the previous section).

2.2.2 PKI requirements

Each of the identified payment techniques have associated PKI requirements. Only when appropriate candidate mechanisms have been identified can the PKI requirements can be properly analysed.

2.3 Other issues

Before proceeding we note certain other areas of WP1 work which potentially involve the use of public key cryptography. However we do not explore these areas further here.

2.3.1 Network layer protection

The possible use of IPsec is discussed in Section 4.4 of [3]. Whilst IPsec is not intrinsically public key based, key management for IPsec could be based on the public key cryptography based IKE protocol, and hence have PKI requirements. However, the current Shaman proposal is not to follow this route and instead use the key established across the first hop.

2.3.2 Security for QoS control

Security issues for QoS control are discussed in Section 5 of [3]. However, no detailed discussion of security mechanisms has yet been provided, and so it is not clear what, if any, PKI requirements exist.

2.3.3 BRAN security

As described in Section 6.4 of [3], ETSI-BRAN is currently considering a number of options for authentication mechanisms for HIPERLAN/2 networks for use when interworking with 3G networks. All three alternatives currently being considered for the ‘Extensible Authentication Protocol’ (EAP) are public key based.

2.4 Areas for further research

We can identify the following major PKI-relevant issues arising from the above discussions.

- The nature of the BRAN Access Network needs to be better understood in order to determine whether using a PKI within this network is likely to present major problems.
- A major issue relates to MN/AN authentication. As discussed above, the choice of authentication protocol has a potentially large impact on the complexity of PKI provision. Hence the final choice for the protocol may be a complex trade off between complexity of the protocol itself, the level of MN privacy offered, and the complexity of the PKI to support the protocol.
- When selecting possible alternative payment methods for network access, the complexity of the associated PKI will need to be carefully considered.

These areas merit further study within WP1 and WP3.

3 PKI for securing local interfaces – the "personal PKI"

Shaman/WP2 has requested Shaman/WP3 to develop methods for the use of PKI to secure communications between devices in a PAN. Specifically, with regard to the pairing of "second party components" (as defined in the PAN reference model within [4]), a method for two PAN components to securely exchange their public keys is required. It is assumed that the two devices cannot rely on either existing symmetric shared keys or connection to a global PKI that both devices trust.

Two methods have been identified by WP3 to provide the functionality requested by WP2:

- "traditional" PKI, but adapted for local use (and where there is no interface to a global PKI);
- "identity based" cryptographic schemes.

In this section, these two schemes are described, analysed and compared and a recommendation made to WP2.

First, the requirements of the personal PKI are identified and clarified.

3.1 Personal PKI Requirements

The underlying requirement is for two devices, which do not share any pre-existing symmetric keys or root certificates, to be able to securely exchange public keys which each device can verify. In this section we identify the requirements that arise of a 'conventional' PKI solution is followed, albeit adapted to a PAN environment. In such a case, one of the devices within the PAN is defined as the "personal CA" and is responsible for issuing public key certificates to other devices.

The following functional requirements therefore result (many are taken from [4], section 3.2.2):

a) the personal CA key pair can be securely generated within the device, or securely generated and transferred to the device at manufacture, and (in both cases) the private key securely stored when on the device;

b) the root public key of the personal CA can be securely transferred to those devices that will have to verify certificates issued by the personal CA;

c) the personal CA can generate public key certificates for mobile devices (and in such a way that the security of the personal CA private key is not endangered);

d) mobile devices can verify certificates issued by the personal CA, and can check certificate validity and revocation status where appropriate.

The general security requirements applying to methods used in the personal PKI are:

f) no third party passive interceptor of communications can learn any secret information;

g) no third party active interceptor of communications can manipulate the exchanges between mobile device and personal CA so that a public key certificate is created for the incorrect device or that contains incorrect data (e.g. a public key other than that created by the mobile device);

h) For securing the transfer of the personal CA root certificate from the personal CA device to another mobile device, the interaction between a mobile device and personal CA shall use at least a 'weak' shared secret, e.g. a shared password or PIN, and the method of this use should be capable of resisting 'brute force' attacks on the shared secret; that is, one of the secure passkey protected mechanisms listed in section 3.2.1 of [4], or a method of equivalent strength, should be used.

Additional and optional functional requirements are:

i) the security-critical personal CA functionality (including key generation and storage functions) should preferably be removable, personal and transferable;

j) the security-critical personal CA functionality can be directly verified and readily enabled/disabled from a single gateway and/or master user.

3.2 Local PKI, using traditional PKI methods

As defined above, a Personal CA provides CA functionality within a Personal Area Network (PAN). Such a device is able to issue certificates to all other personal devices (in the category of first party devices).

Since all the personal devices for a particular user will be equipped with certificates issued by the same CA, i.e., the personal CA, they will all share a common root public key. Consequently, the public keys in the certificates can be used to exchange session keys or authenticate first party (and second party) components in PANs.

Further investigation will need to be performed on the issue of certifying the devices of a second party, i.e. the certification of devices from a different owner.

3.2.1 Operation of a personal CA

In this section we describe the operational processes of a personal CA.

3.2.1.1 CA initialisation

Before use, the personal CA must be initialised. This involves generating a signature key pair for the personal CA. The personal CA will therefore need to incorporate means for generating sufficient random material to enable it to securely generate a signature key pair.

The requirements for the personal CA functionality listed in Section 3.1 point towards the use of a smart card or other portable tamper-resistant device. Particular advantages could be obtained by combining this device with a device already used for global network access, e.g. a GSM/UMTS *SIM device.

3.2.1.2 Device initialisation

This will require a mobile device to perform the following steps – not necessarily in the order specified. (Note that some of these steps may be combined).

- The mobile device will generate any necessary key pairs (signature keys, encryption keys, etc.).
- At some point in this process the mobile device must import authentication material from its owner. As discussed below, for a variety of reasons this should require the minimum number of keystrokes by the user, i.e. it should be a ‘weak’ passkey.
- The mobile device will be informed of which other device is the personal CA, or will have to ‘discover’ this device across the PAN.
- The personal CA root public key will be passed to the mobile device. This must be done in such a way that the mobile device can verify the integrity and origin of the CA public key.
- The mobile device will provide its public key(s) to the personal CA. This must be done in such a way that the personal CA can verify the integrity and origin of the public key(s) before it generates any public key certificates.
- The personal CA will generate a public key certificate for the mobile device.
- The newly created public key certificate will be passed to the mobile device. (The mobile device can verify the certificate using the CA root public key).

3.2.1.3 Candidate mechanisms for password-based initialisation

There exists a considerable literature on protocols designed to enable two entities who share a password (a ‘weak key’) to use it to authenticate one another and (possibly) establish a shared secret key. A number of protocols of this type are known that are resistant to off-line searching attacks for

the weak key, even if the attacker participates in the authentication protocol. A short discussion of such schemes can be found in Section 3.3 of [4].

What is required here is slightly different, in that we wish to have a means for two entities to exchange public keys in an authenticated way, based on a weak (short) shared secret. Of course, one approach would be to first establish a shared secret key (as above) and then use this to establish an authenticated channel. However, other possibilities, if they exist, would also be of interest. In fact, the use of passwords for the PKI registration process is an issue of much more general application than for Personal Area Networks.

A possible candidate mechanism for password-based initialisation is discussed in Section 3.2.2 below.

3.2.1.4 Public key status management

Once a mobile device has performed the exchange of public keys with the personal CA, the issue remains of managing the status of public keys, and disseminating public key status information. Specifically, if a public key is compromised, or suspicion of a possible compromise arises, how is this information disseminated to parties within the PAN? The issue of revocation and certificate status management is discussed in more detail in Section 5.1, but solutions discussed there, e.g. OCSP, may not be appropriate within the PAN environment.

This issue is discussed further in Section 3.2.3 below.

3.2.2 A protocol for device initialisation

The security requirements for the device initialisation process have been listed in Section 3.1 above. This issue has been considered within WP2 of Shaman, and a protocol has been proposed within WP2 to meet the identified requirements – note also that this protocol has been previously described in [5]. We sketch this protocol below (this protocol will eventually be specified in a Shaman/WP2 deliverable).

Before giving this protocol observe that, in order to operate successfully, the mobile device and CA must meet certain minimum requirements.

- The personal CA must be equipped with a display and a simple input device for giving it commands.
- The mobile device must possess a moderately sophisticated user interface – that is it must possess both the means for a user to input a sequence of digits (e.g. a numeric keypad or at least two buttons to insert a sequence of zeros and ones), and a simple output device, e.g. an audio output, to indicate success or failure of the initialisation process.

The question of how to perform the initialisation process for mobile devices which do not possess a numeric keypad (or similar) is an issue for further study (see also Section 3.2.4).

Finally note that we also assume that the mobile device and personal CA can communicate via a wireless interface.

3.2.2.1 Protocol specification

The protocol operates as follows.

1. The Personal CA must be reliably informed of the identifier for the mobile device. This could, for example, be achieved by the user typing the identifier for the mobile device into the keyboard of the Personal CA. However, it could also be achieved as part of the protocol itself (see below).
2. The Personal CA sends its public key P_{CA} to the mobile device, and the mobile device sends its public key P_M to the personal CA. This transfer is assumed to take place via the wireless interface. Along with P_M, the mobile device can send any other information it wishes to have included in the public key certificate which the personal CA will generate (again via the wireless interface). This could, for example, include the identifier for the mobile device.

3. The Personal CA now generates a random key *K*, where *K* is suitable for use with a MAC function shared by the Personal CA and the mobile device. Using this key *K*, the Personal CA computes a MAC as a function of P_{CA}, P_M and any other data supplied by the mobile device. The MAC and the key *K* are then output by the personal CA (e.g. via a display attached to the personal CA).
4. The user now types the MAC and key *K* into the mobile device, which uses the key *K* to recompute the MAC value (using its stored versions of the public keys and associated data). If the two values agree then the mobile device gives a success signal to the user. Otherwise it gives a failure signal.
5. If (and only if) the mobile device emits a success indication, the user instructs the personal CA to generate an appropriate public key certificate. This certificate generation must only take place **after** the mobile device has given the required positive indication. This certificate can then be sent (unprotected) to the mobile device via the wireless interface.
6. The mobile device now performs two checks before accepting the certificate. Firstly the mobile device checks the signature using the personal CA's public key (P_{CA}). Secondly the mobile device verifies that the data fields within the certificate (including the public key P_M and the identifier for the mobile device) are all as expected. The protocol is now complete.

3.2.2.2 Implementation considerations

Apart from meeting the security objectives of the initialisation process, a further primary objective for the design process is to minimise the length of the data strings that the user has to type into the mobile device. This is important for several reasons.

- Firstly, the user will wish the initialisation process to be as quick and simple as possible, arguing in favour of the minimum number of required keystrokes. This is accentuated by the fact that the keypad on the mobile device may be rather small and awkward to use for large strings of data (notwithstanding the ability of many users of existing mobile devices to send text messages using small numeric-only keypads).
- Secondly, the initialisation process should have a high probability of successful completion. This will clearly not be the case if the user is required to enter a large number of digits, especially using a small keypad and/or with a small or non-existent display to give feedback.
- Thirdly, if typing in long data strings is necessitated by the scheme, then it might be just as simple to type in the respective public keys, thus avoiding the threats that arise from use of the wireless interface.

In the protocol specified in Section 3.2.2.1, this minimisation of data entry can be achieved by using a very short key *K* and a very short MAC. For example, if the key and MAC both contain 4 decimal digits, then the probability that an attacker can successfully manipulate any of the information protected by the MAC is very small. (The precise effects of particular parameter choices on the security level of the protocol are discussed in more detail in Section 3.2.2.4 below).

3.2.2.3 Proof of possession requirements

In some circumstances, before generating a certificate, it is necessary for a CA to ensure that the requester of a public key certificate knows the private key corresponding to the submitted public key. To provide this service, the mobile device could supply a 'proof of possession of the private key in step (2) of the protocol specified in Section 3.2.2.1 above.

The nature of this proof of possession will vary depending on the 'type' of the mobile device's public/private key pair. For example, if it is a signature key pair, then the private key can be used to create a 'self-signed certificate', i.e. a signature generated using the mobile device's private key on a string containing the mobile device public key and the mobile device's identifier.

The nature of a proof of possession for a private decryption key or a private key agreement key is *for further study*.

3.2.2.4 Analysis of protocol

The purpose of the protocol described in Section 3.2.2.1 is to transfer the public keys and other data needed for production of the certificate. All data to be transferred is assumed to be public. Therefore the security goal is to protect the integrity of the data, not the confidentiality. The necessary integrity protection is performed using the MAC-based checking procedure in steps 3 and 4 of the protocol.

The security threat against the protocol is an active adversary who by any possible means tries to modify the data exchanged between the CA and the mobile device in step 2. If such a modification, insertion of new data or deletion of data takes place on the wireless communication between the devices then the data sent by one party will be different from the data received by the other party.

The adversary is successful, if the integrity protection method fails to detect modification of data. In what follows the probability of failure is determined.

For the security analysis of the protocol it is essential to observe that the communication channel used for the checking procedure in steps 3 and 4 is completely independent of the wireless communication channel used for other exchanges of data in the protocol.

Also different instances of the protocol are independent. This is due to the fact that for each protocol instance the key *K* is randomly generated. The key is generated independently for each protocol instance and for each MAC computation. This means, in particular, that even if the data between two protocol instances are strongly related, the respective MAC values computed using different keys are independent. To achieve this randomisation property of MAC the length of the key should be larger than or equal to the length of the MAC value.

Let *m* be the bit length of the MAC and *k* the bit length of the key. Then the adversary is successful either if he guesses the key *K* correctly, or if the guess for the key is not correct, but the MAC values for the different data happen to be the same. Hence the probability of success is

$$\frac{1}{2^k}+\left(1-\frac{1}{2^k}\right)\times\frac{1}{2^m}=\frac{2^m+2^k-1}{2^{m+k}}.$$

For a fixed total length of the bit string to be entered to the mobile device, this probability is minimised if the lengths of the MAC and the key *K* are equal, that is, if $m = k$, in which case the success probability for an adversary is approximately equal to 2^{1-k}.

3.2.3 Techniques for public key management

As mentioned in Section 3.2.1.4 above, once a mobile device has been initialised, there is a need for ongoing management of keys pairs and certificates. We now consider these issues in detail.

3.2.3.1 Issues

We start by listing the various aspects of public key management for which solutions need to be found.

- *Certificate and key pair update*. The public key certificates issued by the Personal CA will (almost certainly) have a specified expiry date. Once this date is reached the mobile device will need to be equipped with a new certificate. This may be for the sane key pair or for a new key pair.
- *Key status management*. At any time a mobile device's private key (or the mobile device itself) may be compromised or stolen. In such an event, all entities within the PAN will need to be informed that the public key certificate(s) assigned to this device should be revoked (i.e. no longer considered valid). In a similar way, the Personal CA may itself be compromised or stolen, in which case the Personal CA root key needs to be revoked. Information on which keys have been revoked will need to be distributed to mobile devices in a timely and efficient way.
- *Trust management*. The relationship between the mobile device and the personal CA will need to be managed, including CA (root) key update and the possible replacement of personal CA devices, especially in the event of lost or stolen personal CA devices.

3.2.3.2 Certificate and key pair update

If the mobile device merely wishes to obtain a new certificate for an existing public key, then because of the scale of the personal PKI a simple solution is possible. Given that the total number of personal devices will be small it is likely to be possible for the personal CA to securely retain a copy of all public keys for which it generates certificates. It could even routinely check the certificates to see if any of them have expired. Once the need for a new certificate has been determined, the personal CA device simply asks the user if the existing key pair should be renewed. Once the user has agreed, a new certificate can be generated and passed to the device concerned across the wireless interface at the next opportunity.

Even if storing all public keys at the personal CA is not feasible, in certain cases it may be possible to use a relatively simplify certificate renewal process. The mobile device requiring a new certificate could pass the expired certificate to the personal CA which would then pass the relevant information to the user for a decision. If the user agrees a new certificate can be generated.

If a new key pair is to be assigned to the mobile device, then the renewal process becomes more difficult. In some cases it may be possible to use the old key pair to establish a secure exchange between personal CA and mobile device – however, if the key pair is still trusted to secure this process then it is not clear why it would need to be changed. Indeed, the default for many inexpensive mobile devices may simply be to use the same key pair indefinitely.

However, if a new key pair is definitely required, and if the old key pair cannot be used to secure the necessary interactions between personal CA and mobile device, then a new imprinting process will probably be necessary. However, given that this will involve relatively few user keystrokes, and given also that this will probably be a rare event, this should not present a huge practical problem for the user.

3.2.3.3 Key status management

We consider two different ways in which certificate status information can be disseminated to mobile devices. The choice between the two approaches depends on the online availability of the personal CA.

3.2.3.3.1 Online status dissemination

The first approach we call *online status dissemination*. This is designed for use in the case where the personal CA is available online to every mobile device either permanently or at least at frequent intervals. In the case where the personal CA is permanently online then an online status query protocol could be used, e.g. a protocol along the lines of the Online Certificate Status Protocol (OCSP). However, because of the small scale and relatively closed nature of the personal PKI it may be possible to use a simplified version of OCSP.

In the case where the personal CA is not always online, but is nevertheless online at frequent regular intervals, the use of routinely distributed Certificate Revocation Lists (CRLs) – see, for example, X.509 – would appear to be appropriate. In this approach the personal CA generates new CRLs at regular intervals and distributes them automatically to all mobile devices. Whilst the personal CA is not online permanently, and neither are all mobile devices, this approach will be appropriate in cases where the personal CA is online sufficiently often that the chances of every mobile device having the latest CRL is very high.

3.2.3.3.2 Ad hoc status dissemination

The second case we call *ad hoc status dissemination*. This is designed for use when the personal CA may only be online intermittently or rarely. In such a case, a mobile device may not be online at the same time as the personal CA very often, in which case directly distributed CRLs no longer appear appropriate. Thus an alternative means for distributing CRLs appears to be necessary.

As in the previous case we assume that the personal CA generates CRLs at regular intervals. We now suppose that the personal CA is online sufficiently often that it can distribute the latest CRL to at least one mobile device (if not then there is clearly no way of distributing timely status information). Subsequent distribution of CRLs is then assumed to occur in an ad hoc fashion between mobile

devices. That is, whenever mobile devices communicate, they exchange the serial number of the CRLs they possess. If one device has a higher serial number than the other then it passes the latest CRL to the other device. Thus the latest CRL should disseminate across the PAN very rapidly, and without requiring any active support from the personal CA. Such an approach may even be appropriate in other networks, although that is outside the scope of this discussion.

3.2.3.4 *Trust management*

We first consider the routine updating of root keys, i.e. when an existing personal CA wishes to update its key pair. If the old root public key has not been revoked, then this could be achieved by distributing a certificate for the new root public key signed using the old CA private key. Whilst this approach has dangers, it may be sufficiently secure for use in a PAN environment. The only alternative would appear to be to engage in a new imprinting process with all mobile devices, which could be a rather onerous process for the user.

The case of a compromised or stolen personal CA is rather more difficult. In such a case there is a need to inform all mobile devices of this in a timely way. Of course, once the root key has been revoked, then secure communications between devices will become impossible unless another root key (and a certificate signed using this key) is available. There would appear to be two main approaches to dealing with this issue.

The first approach is to use multiple personal CAs. In this case every device will have multiple root keys and multiple certificates for their public key(s). If two or more Personal CAs are available at the time a mobile device is imprinted, then it should be possible to devise a special version of the imprinting protocol given in Section 3.2.2.1 to enable simultaneous registration and certificate generation. When one CA root public key is to be revoked, then the mobile devices can be informed by the remaining personal CAs, using the same mechanism as is used to disseminate revocation information for other mobile devices.

The second approach is to re-imprint every device with a replacement personal CA as soon as possible after the loss of the old personal CA. Such a process can be designed to simultaneously revoke the old CA and register with the new CA. An appropriately modified version of the imprinting protocol described in Section 3.2.2.1 above will need to be used.

3.2.4 Issues for further research

The above discussion indicates that the following issues will need to be addressed on ongoing research within Shaman WP3.

- First and foremost, more research is required on initialisation methods for the reliable exchange of public keys between a mobile device and a Personal CA, particularly in the case where the mobile device does not possess a numeric keypad or other means of user input. The security of the initialisation process is particularly important for the 'personal CA', since there will be no manual intervention in the public key exchange, making 'man in the middle' attacks a genuine threat. This work should include the examination of manual proximity security procedures, and methods based on passwords/passkeys. In particular, it would be of interest to see if existing protocols (which typically achieve entity authentication and/or session key establishment) could be modified to achieve reliable public key exchange. In particular it is worth noting that, whilst a session key may be useful in performing this reliable exchange of public keys, it is not required for its own sake.

  A further threat to this process could arise from 'false' personal CAs. The initial exchange must therefore provide a measure of mutual entity authentication.

- Methods for proof of possession which are appropriate to this environment and which are also appropriate for private decryption keys and private key establishment keys need to be considered.

- Possible methods for revocation which are appropriate to the PAN environment need to be examined. This examination should include a consideration of the impact of the loss or compromise of the personal CA device.

- The use of multiple personal CA devices should be analysed, in particular as a possible way of dealing with the potential loss and/or compromise of a single personal CA device. In general personal CAs will be much more prone to such threats than CAs in a more conventional PKI environment. This should include an examination of ways in which the imprinting protocol of Section 3.2.2 can be adapted to allow simultaneous imprinting using multiple personal CAs.
- The choice of certificate formats for a local PKI should be considered. If the local PKI will operate autonomously of other PKIs, then there are relatively little advantages to be gained from employing a 'standard' certificate format, and significant possible disadvantages (notably in terms of certificate size, and the complexity of certificate generation and verification).

Finally, in parallel with this ongoing research, assessments should also be made of which parts of the functionality ought to be included in the WP5 demonstrator.

3.3 Identity Based Cryptographic Schemes

The use of ID-based cryptography presents an interesting alternative to the use of a conventional PKI solution within a PAN environment. Whilst ID-based cryptography will not avert the need for a secure initialisation process, involving a trusted exchange between a mobile device and a Trusted Third Party, it will remove the need for any subsequent exchanges of public key certificates between mobile devices. Moreover, whilst the trust model for an ID-based system may not always be appropriate, in a PAN environment the requirement for all devices to strongly trust one entity does not seem likely to present a major problem.

3.3.1 Background

The origin of ID-based cryptography goes back to 1984, when Shamir described the potential utility of an encryption scheme in which the public key can be an arbitrary string. The original motivation is to simplify certificate management in e-mail systems. In such a scheme, the public key is derived (using a publicly known function) from the identity of owner, e.g., from the owner's e-mail address. According to the definition in the Handbook of Applied Cryptography [6], an ID-based cryptographic system is an asymmetric system wherein an entity's public identification information plays the role of its public key, and is used as input by a trusted authority (along with the authority's private key) to compute the entity's private key.

3.3.1.1 ID-based encryption

In an ID-based encryption scheme, the encryption and decryption functions are the same as in a traditional scheme. The only difference is in the key management. The scheme is best illustrated by an example.

Suppose that Alice wants to send a message to Bob. She first encrypts it using the string bob@hostname.net. There is no need for Alice to obtain Bob's public key certificate, thus simplifying the certificate management. The role of the CA, however, is not eliminated. When Bob receives the encrypted message, he has to contact a trusted third party, the Private Key Generator (PKG), authenticates himself and obtains his private key. The private key that he obtains is valid as long as his public key is valid. Methods for key revocation are discussed later. Similar to the notion of ID-based encryption schemes are those of ID-based authentication, and signature schemes.

Until recently there was a lack of practical and secure ID-based encryption schemes. However, in the last couple of years, two promising ID-based encryption schemes have been proposed, by Boneh and Franklin [7] and Cocks [8].

3.3.1.2 ID-based signatures

ID-based signature schemes are the ID-based natural analogues of traditional signature schemes. As expected, if Bob wants to generate the signature first contacts the PKG, authenticates itself and obtains the private key. This is then used as in a traditional scheme to generate a signature. When Alice

receives a signed message from Bob, Alice can verify the signature in a traditional way using Bob's identity information as his public key, thus avoiding the use of any certificates.

Satisfactory ID-based signature schemes have been known since 1986. See for instance, [9], [10] and [11].

3.3.1.3 ID-based key establishment

ID-based key establishment schemes are a further class of ID-based cryptographic schemes. As in the ID-based encryption and signature schemes, each public key is a function of the user's identity. These public keys can then be used in key establishment protocols (involving pairs of users or 'conferences' of more than two users) without use of public key certificates.

3.3.2 Practical issues

We next briefly consider certain practical issues which arise in the use of ID-based schemes.

3.3.2.1 Key revocation

Key revocation in ID-based systems can be done in a very efficient way by limiting the lifetime of public keys. This can be achieved by defining the public key to consist not only of the identity of the owner, but the identity with a date appended to it. Continuing the previous example, if the public key of Bob is to be renewed once a year, then his public key for the year 2001 would be bob@hostname.net2001, and he would have to obtain a fresh private key once a year. Note again, that Alice does not need Bob's certificate in order to obtain his public key. Also, one could add more granularity to the revocation system, by simply adopting a different convention for the public key (e.g., instead of the year append the month).

However, unless the lifetime of public keys is made very short (with a consequent overhead relating to the need for new private keys to be distributed very regularly) one cannot completely avoid CRLs. If a public key is revoked due to compromise of the corresponding private key, then the public key, i.e., the public identity, has to be added to a CRL. One may be able to avoid CRLs in a very constrained system. This idea will be described in the context of PANs.

3.3.2.2 Private key distribution

This may in general be a problem, as the private key is communicated from the PKG to Bob via a secure and authenticated channel. However, the same problem exists with traditional PKIs, where the key pair is generated by the CA. Furthermore, for the applications in SHAMAN, and more specifically PANs, this may not be a major obstacle: it is conceivable that the device that plays the role of the PKG will be physically close to the "client" device, so that the transition of the private key (which is done only once at the initialisation phase), can be considered secure.

3.3.3 Advantages and disadvantages of ID-based schemes

Given the above considerations, an ID-based system in general requires less communication. For example, consider a system with n parties, where all parties want to communicate with everyone else. Using traditional PKI, each party would have to retrieve everyone else's certificate, i.e., $n(n\text{-}1)$ messages in total. In an ID-based system, each party would have to contact the PKG and retrieve its private key, i.e., n messages in total. In practice, there are ways of offsetting the certificate delivery in a traditional scheme. For interactive communication there will be the need for a handshake, so that the certificate can be added at this point to an existing message. In other types of communication, such as messaging or e-mail, when user A requests user B's certificate in order to encrypt a message, user A can append his/her certificate to the request message, which then can be used by user B in a reply message. Thus in practical terms, the number of messages need not be significantly different in the two schemes. However, there is an overhead in bandwidth due to the presence of certificates in the traditional scheme.

The main disadvantage is the need of a secure channel for the distribution of private keys. Another issue is that all the parties in the system need to know the global parameters of the system, e.g., the method that transforms the public identities to public keys. Furthermore, an ID-based system relies on

the functionality of a PKG, which must be trusted. These considerations make ID-based systems suitable for rather constrained environments, where global parameters, and key distribution is manageable, and a trusted third party is a reasonable assumption.

3.3.4 ID-based systems in the PAN

Personal Area Networks (PANs) are very constrained environments: few components exist, and they are located physically close to each other. These characteristics make the concept of ID-based cryptography an attractive solution to the security requirements posed by PANs. In the following discussion, we make the following assumption: in the PAN, there is a pre-specified device that has the functionality of the PKG. This device needs to have a reasonable amount of processing and storage capabilities, to be able to carry out the PKG functionality. Furthermore, it has to be trusted by every other device, as it will have access to every private key in the system. In the context of the PAN both assumptions seem reasonable: the processing and storage capacities needed are not extraordinary for a mobile device, and the trust requirement is met, if one thinks of the devices of a PAN as owned by one person.

3.3.4.1 Private key generation

As indicated in the previous paragraph, the private keys are generated by a pre-specified device, which acts as a PKG. In the process of generation of a private key, the PKG uses the identity of the device whose private key is generated. This is the same identity that is used as a public key of the device, and must therefore be unique. One can imagine several naming schemes, such as serial numbers, strings consisting of device owner and device name, etc.

3.3.4.2 Distribution of system wide parameters

In an ID-based system, e.g., an ID-based signature scheme, besides the algorithm used, several other parameters have to be known by every participating member. These parameters should be specified by the PKG, and be made available to all other devices prior to the generation or verification of any signature.

We should emphasize that the system parameters are not secret, and therefore need not be encrypted when transmitted. However, they should be sent over an authenticated channel, in order to protect against active attacks. One possible way to do this is by establishing an authenticated channel using the ideas of Section 3.3 in M2.2. More specifically, when a mobile device joins the PAN, it executes (together with the device that acts as a PKG) a protocol that establishes an authenticated channel based on a weak secret (such as a passkey). This channel is subsequently used by the PKG to transmit the system parameters to the new device.

Another possibility for exchanging parameters in an authenticated way would be as follows (this method was proposed within the context of Shaman/WP2 work, and will be described in more detail in a WP2 deliverable). The device acting as the PKG sends the parameters to the client device. Then it generates a string a at random, and uses this string to compute the MAC d of the parameters. The pair (a,d) is then transmitted (unprotected) to the client device. The device now uses a to compute the MAC δ of the received parameters. The user (same for both devices) checks that the two pairs (a,d) and (a,d) are the same. If they are the same then the received values are indeed those transmitted. If not, the process is repeated.

3.3.4.3 Distribution of private keys

As part of the ID-based scheme, a private key must be transmitted from the PKG to a mobile device over a private and authenticated channel. One solution would be to assume that private keys are distributed only during an initialisation phase, which is physically secure. In many cases, however, distribution of all private keys during initialisation may not be a realistic assumption. Furthermore, if a private key is compromised between initialisations, the corresponding device is left without “public key capabilities” until the next initialisation. In what follows, we explore alternative methods of private key distribution, which do not require a physically secure environment. These methods allow re-keying and re-distribution of system wide parameters at any time. These features make the use of the system realistic, and may also simplify key revocation.

3.3.4.3.1 Password based shared secrets

The application of password based techniques for establishing an authenticated and private channel to this situation deserves more research. The basic protocols are described in Section 3.3 of M2.2.

3.3.4.3.2 Authenticated Diffie-Hellman

Another possibility would be to establish a shared secret between the PKG and the client device, and then use this as a key for a symmetric encryption scheme. The basic Diffie-Hellman protocol does not provide authentication, which allows for active attacks. One possible way of authenticating the Diffie-Hellman key exchange would be to authenticate the received values in the way described in the last paragraph of Section 3.3.4.2. We note that even if the keying material for the MAC (the random string *a*) and the MAC itself may be very short, this does not compromise the security of the system. The reason is that those values are only used to authenticate the exchange of the "Diffie-Hellman" values. Thus, an active attacker would have to "break" the MAC in a *very* limited amount of time (the time needed by the user to compute the MAC and check that the two pairs are the same). From this point on any communication is protected by proper keys.

3.3.4.4 Key revocation

As explained in Section 3.3.2, the lifetime of a public key can easily be encoded in the public key itself. However, no general solution of key revocation is given: one still has to consult a CRL for the revocation status of a public key. The situation poses a further disadvantage: once a private key is compromised, the corresponding party cannot obtain a new private key given the global system parameters, unless its identity changes. A remedy for the last problem would be the following. Instead of CRL, the system makes use of a Current Identity List (CIL). Once the private key of a device is compromised, the device notifies the PKG, which in turn makes a slight change to the identity of the device (e.g., by appending a number to the fixed identity), generates the new private key, and transmits it to the device using the secure channel established using the techniques of Section 3.3.4.3.1. Then, it changes the entry in the CIL corresponding to that particular device and transmits the list to every other device in the system. Now, the compromised key has been revoked, the affected device has a new private key, and all devices know the new public key of the affected device.

Yet another solution to the problem of key revocation might be the following. Upon compromise of a private key, the PKG is notified, which in turn changes the global parameters of the system, generates new private keys, and transmits to each device in the system the global parameters along with its new private key using the secure channel established using the techniques of Section 3.3.4.3.1. Now the compromised private key has been revoked, and all devices know the new system parameters and their private keys.

A comparison of the two techniques is in order. The first technique is more efficient, as only one new private key is generated. It does however make necessary the use of CILs. The second solution seems to be an over-kill: if one private key is compromised, the whole system is re-initialised. The advantage of this solution is that no CRL or CIL is needed. The choice between the two solutions might be based on the following consideration: if one expects private keys to be compromised only rarely, e.g., if the lifetime of each key is a few days only, then one might choose the second solution, as the overhead of CILs is avoided. If on the other hand, one expects private keys to be attacked frequently, one should use the first solution, as the revocation of each individual private key is done more efficiently.

3.4 Comparison of methods

In this section an initial comparison of the two approaches to providing the 'personal PKI' is provided. This comparison is inevitably provisional since investigations of appropriate protocols for 'pairing' two devices (which could be used for a simultaneous exchange of public keys) are at an early stage. Hence this comparison is based on working assumptions about the properties of protocols to support and manage the personal PKI.

We divide this discussion into the following sub-topics, covering particular areas of activity for a mobile device:

- *Initial exchanges*, i.e. the exchanges between a mobile device and a personal CA or personal PKG necessary when the mobile device is first added to the personal network);
- *Key use*, i.e. the computations and communications necessary when a public/private key pair is used);
- *Key updates*, i.e. the computations and communications necessary when a private key is updated;
- *Key status management*, i.e. computations and communications necessary in order to establish the status of a public key.

3.4.1 Criteria for comparison

We use the following measures to compare the two approaches.

- *Communications complexity*, i.e. the number and length of messages exchanged between a mobile device and the personal CA/PKG, and/or between a pair of mobile devices.
- *Computational complexity*, i.e. the amount of computation that the various parties need to perform.
- *Management complexity*, i.e. the management overhead for the particular operations.
- *Overall security level*, i.e. the strength and trust properties of the security mechanisms.

We discuss the performance of the two approaches with respect to each of these measures. In a concluding subsection we give a brief summary of our findings.

3.4.2 Initial exchanges

As discussed in Sections 3.2 and 3.3 above, the following tasks will need to be performed by the mobile device and the CA/PKG at the time a mobile device is first introduced within a personal network. Note that we ignore the initialisation tasks that need to be performed by the CA or PKG itself. These tasks are not trivial; however, since they are a one off overhead, a comparison of these two tasks is probably not particularly relevant here.

- The **Personal CA approach** involves the following steps:
  1. the mobile device needs to establish an identity,
  2. the mobile device needs to generate a key pair,
  3. the CA and mobile device need to exchange public keys in a reliable way,
  4. the CA needs to generate a public key certificate for the mobile device,
  5. the CA must send the newly generated certificate to the mobile device, and
  6. the mobile device must verify the newly received certificate (using the CA public key).
- The **ID-based approach** involves the following steps:
  1. the mobile device needs to establish an identity,
  2. the PKG must send the public domain parameters to the mobile device in a reliable way,
  3. the PKG must generate a private key for the mobile device,
  4. the PKG must send the newly generated private key to the mobile device in a way which preserves the confidentiality of the private key, and
  5. the mobile device must verify the newly received private key (using the public domain parameters).

We now compare these two processes with respect to the criteria defined above.

- *Communications complexity*. The main differences are as follows. The personal CA approach requires the reliable transfer of a public key from the mobile device to the CA, whereas no such transfer is required for the ID-based approach. The ID-based approach requires the confidential transfer of a private key from the PKG to the mobile device, whereas no such transfer is required for the ID-based approach.
- *Computational complexity*. The main difference in terms of computational complexity is that for the personal CA approach the mobile device generates its private key whereas for the ID-based approach the private key for the mobile device is generated by the PKG. That is, the ID-based scheme involves a transfer of effort from the mobile device to the TTP. The relative amounts of effort required in the two cases will be scheme dependent.
- *Management complexity*. There do not appear to be any significant differences in this respect. In both cases the TTP (CA or PKG) needs to retain a record of the initial transaction. e.g. for certificate status management.
- *Overall security level*. The main difference here is that the level of trust required in the TTP is greater for the ID-based approach. This is because, in this approach, the TTP has access to all the mobile device private keys.

3.4.3 Key use

After the initialisation phase, all the devices involved in the system are assumed to poses the appropriate cryptographic key. That is, each device possesses its own private key, as well as the public key of the device acting as the personal PKI (if this is the case). We compare the two different approaches according to the above criteria considering (when necessary) the purpose of the key (i.e., encrypting or signing).

- *Communications complexity.* For digital signatures the main differences are as follows. In the personal CA approach, when the device sending the signed message normally attaches a copy of its public key certificate. This is an overhead compared to the ID-based approach, where the public key of the sender is already known to everyone. In the case of encryption, the sender needs the public key of the receiver in advance. This seems to imply that the public key certificates are obtained once by each device, and stored for future use.
- *Computational complexity.* Signature generation and verification require the same amount of computing for both approaches. The case may be different in encryption. ID-based schemes such as the one of Boneh and Franklin, require slightly more computation to achieve the same level of security as traditional schemes.
- *Management complexity.* ID-based schemes may be easier to manage, as the public keys (for signature verification, or encryption) are simply the identities of devices that have to be looked up, but not verified.
- *Overall security level.* The level of security is the same for both approaches (assuming that the cryptographic keys are of “equivalent” size).

3.4.4 Key updates

We now consider the task of updating the cryptographic keys.

- The **Personal CA approach** involves the following steps:
  1. The mobile device needs to generate the new pair of keys.
  2. The mobile device needs to send the new public key to the CA in a reliable way.
  3. The CA needs to generate a new public key certificate for the mobile device.
  4. The CA needs to send the newly generated certificate to the mobile device.

5. The mobile device needs to verify the received certificate.

- **The ID-based approach** involves the following steps:

1. The mobile device needs to establish a new identity and communicate it to the PKG (alternatively, this new identity can be established by the PKG according to a pre-specified rule).
2. The PKG needs to generate the new private key for the mobile device.
3. The PKG needs to send the newly generated private key to the mobile device in a way that ensures integrity and confidentiality.
4. The mobile device needs to verify the received key.

We compare the two processes according to the criteria defined above.

- *Communications complexity*. There are no major differences in the communications complexity of the two approaches. The need for a private channel in the case of the ID-based approach seems to imply more message exchanges. However, since the mobile device already possesses the public key(s) of the PKG, establishing the channel is done in a straightforward manner, without additional messages. The fact that the identity of a mobile device is usually short, suggests that the first message in each approach (step 2 in the personal CA; step 2 in the ID-based) is shorter for the ID-based approach.
- *Computational complexity*. The complexity of creating the keys is virtually the same for both approaches. A characteristic of the ID-based approach is that the computation is done by the PKG (and not the mobile device), which might be an advantage, as the "master" device is assumed to have considerable computational power, whereas the mobile devices may be very constrained.
- *Management complexity*. There do not appear to be any significant differences in the management of the keys. Both devices need to keep records of the transaction.
- *Overall security level*. The security level is again similar. One disadvantage of the ID-based approach is that the PGK has access to all private keys, and therefore must be trusted.

3.4.5 Certificate status management

In both the personal CA and Id-based approaches, before a mobile device uses a public key, it has to establish its validity. This involves two things: Check that the key has not expired, and check that the key has not been revoked. In the ID-based approach, the expiration check can be done automatically, since the time interval during which a key is valid can be encoded in the key itself. In the personal CA approach, the check is performed using public key certificates. Revocation status is done similarly in both cases: A list of valid (or invalid) public keys has to be available (alternatively a OCSP-like service should exist).

The problem of making the public keys available to every device is an issue that should be discussed here.

- **In the personal CA approach**, this is done by distributing public key certificates by request, or attaching them at some message (e.g., a signed message).
- **In the ID-based approach**, this can be combined with "certificate status lists". Given that the number of devices in the PAN is relatively small, one could use Current Identity Lists (CIL) that serve two purposes at the same time: they contain the current identities of the devices, and the missing identities are exactly those revoked.

We compare the two approaches according to the usual criteria.

- *Communications complexity*. In the personal CA approach both certificates and certificate revocation lists have to be distributed. In ID-based schemes, the same tasks can be accomplished using CILs only. This seems to lead to less messages.

- *Computational complexity.* The computational complexity of the two approaches seems to be the same.
- *Management complexity.* There do not appear to be any significant differences in the management of the status of keys. One slight advantage of the ID-based approach seems to be that the mobile device needs to maintain only one list, as opposed to one list and a list of certificates in the personal CA approach.
- *Overall security level.* The security level is again similar.

3.4.6 Summary and conclusions

In this section, we summarise the conclusions of the comparison between the two approaches studied above. We give our conclusions for each of the established criteria.

- *Communications complexity.* The personal PKI approach requires more bandwidth, as public key certificates have to be distributed to the members of the PAN. For interactive communications, this distribution may not require more messages (certificates may be attached to other messages) they certainly make some messages lengthier. For connectionless communications, such as email, the advantage offered by ID-based schemes is even more significant, as the sender needs to obtain the certificate of the receiver in advance (which implies one request and one reply message). On the other hand, the distribution of private keys by the PKG in the ID-based approach requires an authenticated and private channel (as opposed to simply authenticated channel in the traditional approach). This clearly complicates matters.
- *Computational complexity.* The main computations in both schemes come from key generation, and computations involving the key, e.g., signature generation, signature verification, and encryption. Key generation is of approximately the same complexity in both schemes. One possible advantage of the ID-based scheme is that all the keys are generated by the PKG, which may be considerably more powerful than the other devices in the PAN. Once the keys have been generated and distributed to the interested parties, the computations are again of approximately the same complexity. One exception is encryption, which is slightly slower for ID-based schemes.
- *Management complexity.* In both schemes there is a need for maintaining keys (preserving their integrity and/or privacy), and checking for the validity of public keys before using them. A solution such as certificate revocation lists is therefore unavoidable in both cases. In the ID-based approach (in principle) one can avoid public key certificates, thus reducing the complexity of managing the system. Even in ID-based systems, however, a device still has to know the "current" identity of the device it wants to communicate with. Depending on how this problem is solved, the above observation is or more or less significance.
- *Overall security level.* The cryptographic primitives in both schemes provide the same security level. A disadvantage of the ID-based encryption scheme of Boneh and Franklin (the only practical such scheme), is that it has been around for only a few years, and the underlying assumptions have not yet been sufficiently tested. Therefore, it is possible that the scheme is not as strong as initially though to be. Interestingly, ID-based digital signature schemes have been known for almost as long as traditional schemes. Another issue with ID-based schemes is that of the trust placed in the PKG. The device acting as a PKG has access to all private keys, and therefore has to be trusted by everyone.

3.5 WP3 recommendation for WP2

The issue of study in Section 3 was how to provide the devices in a PAN with public key capabilities, and the relevant public key management issues. Two different techniques were proposed as possible solutions:

1. A traditional PKI, where a specified device plays the role of the CA in the system, called the personal CA.

2. An identity-based system, where a specified device plays the role of the PKG.

Both solutions were described, and in particular it was analysed how they can be applied to the PAN. Based on this analysis, some conclusions were drawn in Section 3.4.6. In fact, in many aspects the two techniques are very similar. For instance, the use of the public and private keys is almost identical. The differences come in the key management techniques, as well in issues of trust. ID-based techniques require less “material” for management (no certificates needed), saving mainly in bandwidth, and therefore appear to be preferable in this aspect. On the other hand, the trust that needs to be placed in the PKG, as opposed to the traditional PKI approach, where the personal CA need not be trusted, is a clear disadvantage. Which solution should be adapted, depends on the needs and assumptions of WP2.

4 PKI for Secure Execution Environments

This section will examine those WP2 requirements which were both listed in [12] and considered, in [2], as requirements which can be satisfied using asymmetric techniques. In particular, those requirements relating to secure execution environments are considered here. WP3 believes that the use of asymmetric techniques to satisfy these requirements would benefit from further analysis by WP3.

The requirements relate to download of executable code from outside of the PAN, over a global interface. These requirements can be divided into:

- requirements of the PAN component (will be termed "the client") on the Application Service Provider (ASP) or Authorisation Authority (AA);
- requirements of the ASP on the client;
- requirements on the ASP to AA relationship;

Requirement numbers given below are those given in [12].

4.1 Client requirements on ASP and AA

The requirements relate to:

Authentication of executable source (Requirements 3, 16)

Authentication of PAN component manufacturer and recognition as PAN component manufacturer (Requirement 21)

Whether authenticated executable source is authorised or not (Requirement 4)

Possibility for the PAN component owner to define the access policy for applications and negotiate the security policy with the ASP (Requirement 14)[1]

Confidentiality and integrity of downloaded executables and authorisation level thereof (Requirements 17 and 18, 24 and 25)

Possibility for the PAN manager to disable the authorisation capability of AAs

4.2 ASP requirements on client

The requirements relate to:

Authentication of client by ASP (Requirement 15)

Possibility for ASP to receive client capabilities with confidentiality and integrity (Requirements 19 and 20)

4.3 ASP – AA requirements

The requirements relate to:

Authentication of the ASP by the AA (Requirement 22)

Possibility for the AA to confer variable authorisation levels on the ASP (Requirement 23)

Possibility for AA to withdraw authorisation from ASP and for the client to be informed (Requirement 27)

[1] WP2 have clarified that negotiation of security policy is no longer a requirement.

4.4 Should PKI be used to meet these requirements?

A preliminary analysis of whether these requirements could be met with asymmetric techniques or not was given in [2]. This analysis is extended here.

The client to ASP relationship is a many to one relationship at least and probably (assuming a client has a range of ASPs to choose from), a many to many relationship. The key management problems (generation of shared, authenticated secret keys across open networks, and the number of symmetric keys that the ASP must store) with the use of purely symmetric cryptography alone surely mean that PKI techniques must be used. Moreover, a direct security relationship between ASP and client may not always be appropriate given that code may be distributed in a connectionless way. For example, code may be passed to the client via a third party code supplier. In such a case, the use of public key techniques is almost inevitable.

ASP to AA communication is a many (ASPs) to few (AAs) problem and for this reason alone, PK techniques seem much better than symmetric techniques. Further, the ASP to AA relationship involves an authorisation of the ASP by the AA which must be visible to the client. There is effectively a transferred very many (client) to few (AA) relationship for ASP authorisation, again making PKI techniques appropriate.

4.5 Requirements that cannot be met with existing PKI standards and implementations

WP3 believes that some of the requirements cannot be satisfied using existing asymmetric standards and implementations. The following are not available or are not done well:

1. Possibility for the PAN manager to disable the authorisation capability of AAs

   MExE [13] provides the Certificate Configuration Message (CCM). This is a message which an "Administrator" can send to a terminal in order to disable specified third party roots o nthe terminal. However, the CCM does not use standard formats and is unpopular with terminal manufacturer. The mechanism for deciding who the Administrator (the Administrator serves within MExE a role similar to the PAN Manager) is, is also not well defined.

   The requirement arises fundamentally because the PAN manager (which in practice may be a network operator acting on behalf of its subscribers) feels it will suffer from poor authorisation decisions made by organisations it may have no control over, that is, poor decisions made by commercial AAs. This issue, and other SEE authorisation issues are examined in section 4.6.

2. Possibility for the AA to confer variable authorisation levels on the ASP (Requirement 23)

   This can be done within the WAP Signed Content specification [14], in that a Trusted CA (the term used in [14] for a CA that is authorised to award a particular privilege) can award the right to sign content of a particular type to a Trusted Content Provider or not. Attribute certificates could also be used, and this will be examined in section 5.2.

   Also, an ASP may choose to issue codes of various 'classes' (e.g. code suitable for safety-critical devices, which has been very carefully checked, or code for general use, which contains the usual number of defects). There does not appear to be an agreed syntax for the ASP to mark code in this way. Moreover, the client needs to know whether it is authorised to run codes of various types from an ASP, and hence the attribute certificate for the ASP will need to specify which categories of code from that ASP are suitable for execution in which classes of device. Hence the syntax of the attribute(s) needs to be defined.

3. Possibility for AA to withdraw authorisation from ASP and for the client to be informed (Requirement 27)

Revocation mechanisms are defined, but not as a mandatory part of any executable environment specification. There are issues surrounding revocation with particular relevance for wireless clients. Revocation for wireless clients will be examined in section 5.1.

4.6 Authorisation issues for Secure Execution Environments

A major issue with MExE [13], and also now with MIDP NG [15] is that one party, the operator, fears the effects of authorisations given by other parties, e.g. public CAs and manufacturers. For example, the operator fears that a CA, whose root is on a mobile device, may authorise a small company to send executable code to that device by issuing a signing (attribute) certificate to that company, indicating that it is authorised to provide signed code. This small company may turn out to be a rogue company which now sends or pushes signed, "trusted" viruses to many of the operator's subscribers. The virus perhaps causes phones to lock up or to make lots of calls that the affected subscribers did not request. It is widely believed that the operator will suffer most from this, both from lost calls and from customer care time.

This issue comes up frequently in discussions within standards bodies, and is usually accompanied by a lot of emotion and anxiety. This is partly because many parties feel that authorisation decisions are not being taken by the correct party. The following comments can be made:

a) The authorising party is not the party most affected if authorisations are unwisely given.

b) Who has authorised the authorising party (i.e. the CA)? In the case of pre-loaded roots, the manufacturer has authorised the authorising party by putting the root on the client. However, some might say that it would be more appropriate if the party most affected by unwise authorisations, the operator, was the one authorising the authorising party.

c) The authorising party (i.e. the CA) is not the verifying party (the client is the verifying party). Therefore there has been an implicit delegation of authorisation rights from the verifying party to the authorising party, but the verifying party was not consulted about this.

In response, the following comments can be made.

Operators would not suffer alone

In truth, both the manufacturer and the operator will suffer from unwise authorisations by the CA, so **both** should have a say in which CAs are authorised to become authorising parties, and not just the operator, nor just the manufacturer.

Further, it must also be said that a legitimate CA has much to lose from unwise authorisations. The core business of a CA is to issue trusted assertions about the identity of certain organisations, and in the case of signed content, the authorisation given to these organisations. If it turns out that these assertions cannot be trusted, the CA will be seen to be deficient in its **core** business. The operator may suffer some, or even significant losses but its core business reputation is not affected. However, if a CA is seen to issue trusted assertions which cannot actually be trusted, it will be thought to be unable to do the only job it has. Hence we can assume that a CA will, as far as possible, not act unwisely in the authorisations it gives.

The operator could also seek to ensure it did not suffer alone using legal means. The CA will have a contract with the ASPs it has authorised, covering behaviour on both sides. The CA will have a contract with the manufacturers which place its root on their phones. The operator, however, is only likely to have a contract with the manufacturer, and this only if it buys phones directly from the manufacturer. The fact that the operator has no contract with the CA is one of the reasons for unpopularity of the MExE CCM – it would be very difficult for a manufacturer to say to a CA, "We will install your root on our terminals for the price of x thousand dollars per year, but can delete it at any time if an operator orders us to". However, the operator can use the only contract it does possess, namely that with the manufacturer, to hold the manufacturer liable for damages caused by signed code verified by roots the manufacturer had put on the terminal. This is reasonable as the manufacturer has enabled the download of signed code to the terminal by enabling the basic capability and by putting

the CA root on the terminal. The manufacturer can, in turn, hold the CA liable if it issues certificates to parties who subsequently cause damage.

Withdrawal of authorisation

The CA must strike a balance between the amount (and therefore cost) of investigation it does into an organisation before issuing a trusted assertion to that organisation and the desire to maximise their profits, which requires that the price of trusted assertions is not prohibitive. It should be remembered that however much investigation the CA carries out, it has no guarantees that the to-be-authorised organisation will act responsibly for the duration of the authorisation. The past is a guide to the future, but not an infallible one.

In such a situation, and as it seems that all liabilities eventually fall with the CA, a method for a CA to withdraw an authorisation after it has been given is very valuable, as it would allow the CA to withdraw its authorisation from an organisation, that, in spite of the CA's investigations, turns out to be a rogue organisation, and thereby reduce damages and liabilities. Further, it allows the CA to (very carefully) reduce the amount of investigation it does before issuance of assertions. This should allow the CA to reduce the cost of assertions, so potentially increasing the number of authorised parties and so increasing the number and (hopefully, by competition) the quality of downloadable executables. All this can be done whilst being able to withdraw the authorisation if it turns out to have made a mistake.

It can be seen that client revocation checking benefits all the parties, in reducing the potential damage that would be inflicted either directly, or indirectly using legal means. Revocation is analysed from a technical perspective in section 5.1.

(Note however that online revocation checks will not work if the virus writer has written a trusted virus that behaves well for a while. Users will then not be alarmed when they see it and will agree to its installation. A revocation check may be done at this time, but as the virus has not been activated yet, the certificate will have not been revoked. The virus might be activated, or activate itself at a predetermined time, once installed in a significant number of phones.)

Authorisation by the Verifying Party – user authorisation

It was stated that the authorising party is not the verifying party, and that therefore there has been an implicit delegation of authorisation rights from the VP to the AP without the VP's consultation.

This can be corrected to some extent. Though it is asking far too much of users to be involved in every authorisation by the AA, the user should have the ability to authorise or not authorise the generation of chargeable events by applications resident on its terminal.

Further, the user should have the ability to delete root certificates on the terminal, and so effectively de-authorise the power of AA's to authorise parties with respect to that terminal. Indeed, it might be reasonable to enable future security modules to override completely any root keys loaded into devices at the time of manufacture (although such a capability would probably not be routinely included in modules provided to users, and would present a number of contractual issues for the manufacturers).

A further possibility might be to offer a range of different authorisation capabilities to users. For example, the user might choose between contracts offering no power over authorisations (and where responsibility is completely delegated to the operator and manufacturer), and contracts offering some or even all authorisation powers to the user him/herself. The former model might be appropriate for the vast majority of users who have no wish to becomes acquainted with the complexities of attribute certificates and root keys, and who simply wish to have a reliable and transparent means of downloading new capabilities to their devices. The latter model might be appropriate for 'power users' (and/or corporate users) who do have the knowledge and expertise to manage their own authorisation policies. Of course, in this latter case, it would need to be made absolutely clear to users that the manufacturer and operator took no responsibility for damage incurred by using inappropriate root certificates.

4.7 Practical effects of the roles of the involved parties

In this section we consider what actual organisations and organisation types might take some of the roles (AA, ASP etc) above and whether this could have any impact on the resulting architecture. In practice most of this section is to do with analysis of the operator taking the roles of ASP and AA.

The most obvious organisation to mention is the network operator. The operator might take the role of the ASP and/or AA. If the operator took the role of both (or if any type of organisation took the role of both), then this would mean that an open standardised interface would not be required for ASP-AA communications. This would make AA revocation of ASP certificates for internal administrative reasons not necessary as the ASP could just stop using the relevant certificate. However, if an ASP signing key had been compromised, there would still be a need for a revocation mechanism, even though the ASP and AA were the same organisation.

If the operator is the AA then there is the possibility of having the AA's root certificate on the (U)SIM or WIM of the devices which have a (U)ICC. In many cases, this is more secure (in terms of physical tamper resistance) than storage on non-smartcard devices, and gives the opportunity for the operator to update the AA root using symmetric SIM toolkit methods that are completely under the operator's control. For this reason, use of the (U)S(W)IM as a root storage location is specified in both the MExE [13] and WIM [16] specifications and PKI applications on terminals which could have a UICC or WIM should, it is argued, always be open to the storage of root certificates on the (U)SIM or WIM.

If the operator were the ASP and AA, there is a long lived relationship (with the possibility, in the SIM, of secure secret key distribution) between the ASP/AA and the user and symmetric methods could potentially be used. Indeed some way of using standard 3GPP authentication mechanisms is probably possible, though it might make sense to use a separate "master key" for executable authorisation. If symmetric methods were to be used, this would be outside the scope of WP3. (If, as has been suggested, inter-PAN authorisation is conducted using symmetric methods, this mechanism could be re-used for ASP to PAN component authorisation where there was a long lived relationship between the user and ASP).

Similarly, the manufacturer might securely provision the terminal with a secret key, and so do uploads using symmetric methods. However, this would require the secret key to be kept very securely, which is difficult without an explicitly tamperproof area. As the terminal will probably have certificate verification software in any case, the manufacturer would probably use asymmetric methods.

5 Analysis and recommendations of generic PKI issues

This section will extend the analysis of some of the generic PKI issues identified in [2]. Client revocation checking, authorisation, and issues of the complexity of PKI for limited devices are considered.

5.1 Client revocation checking

In D4, we examined the use of Certificate Revocation Lists (CRLs), the Online Certification Status Protocol (OCSP), the Simple Certificate Verification Protocol (SCVP) and the XML Key Management System (XKMS) by clients to check the revocation status of certificates. These four techniques/protocols were described but not compared to any great extent. In this document we take the two most promising of these four protocols, namely OCSP and XKMS, and subject them to further analysis, compare them and then make a recommendation as to which should be used for the applications under consideration within SHAMAN.

We exclude SCVP as progress on it within the relevant standards body (the IETF PKIX group) has stalled, and there is little support for its continuation.

A major fault with CRLs identified in [2] and elsewhere is that CRLs, in the mobile domain, cannot be used to provide up to date certificate revocation information, because their size means that mobile bandwidth considerations prevent updates of CRLs, and infrequent CRL update considerably reduces the effectiveness of CRL use.

Therefore we will take a closer look at OCSP and XKMS only.

To compare the two methods we need a set of criteria for the comparison. These criteria are:

- Security mechanisms (what security is provided and how?)
- Current support (standards and technologies implementing the method)
- Hardware and software support (can mobile handsets support the method? What are the implications?)
- Memory and bandwidth implications

5.1.1 OCSP

5.1.1.1 Configuration

OCSP, being an online revocation method, integrates the use of signed messages from the OCSP responder to the client (mobile handset). The purpose of OCSP is to provide revocation status and nothing else. Figure 1 and Figure 2 show the two ways that OCSP can be configured.

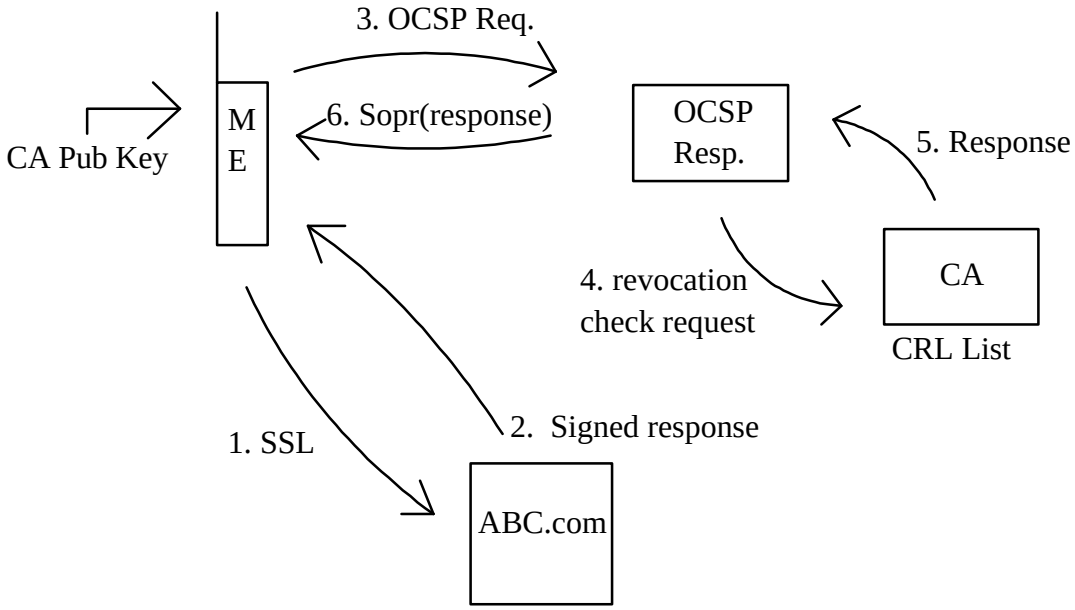


Figure 1. OCSP using issuing CA as OCSP responder

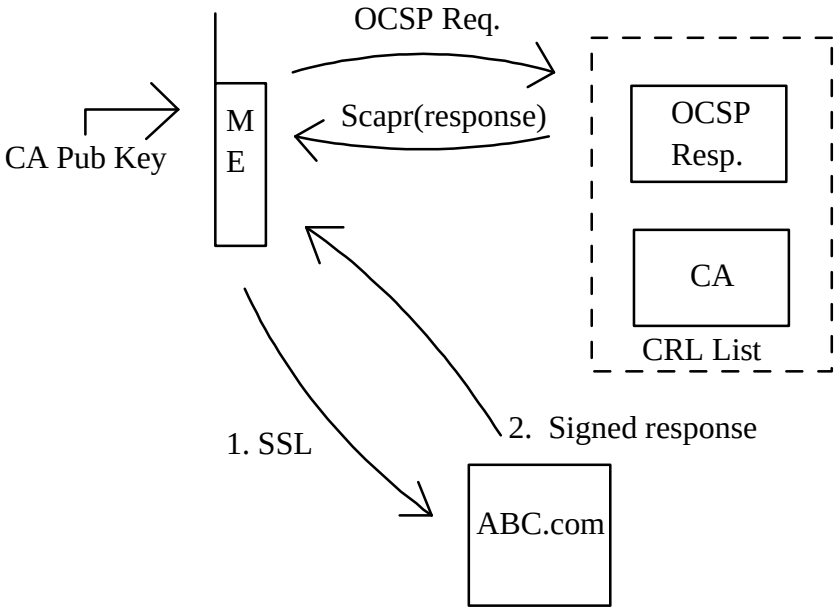


Figure 2. OCSP using delegated OCSP responder

Scenario 1 assumes that the OCSP responder is within the CA. This means that the CA itself signs all responses given by the OCSP. The CA will have an up to date record of the status of certificates it has issued (perhaps as a CRL or perhaps just in a database) stored. The OCSP received response is checked against the certificate status database or CRL. The response is then passed back to the client.

In scenario two, the OCSP responder is separate from the CA. The client now makes a request to the OCSP responder and not the CA. The client has to establish that it can trust the responder. The method for establishing this trust specified in [17] is that the CA issues the responder with a certificate

containing an extension indicating that the responder is authorised to issue OCSP response on behalf of the CA (the OID for id-kp-OCSPSigning must be present in the extendedKeyUsage extension of the responder's certificate). The AuthorityInformationAccess extension can be present in the end entity certificate which is being verified to indicate the OCSP responder that should be used by the client.

OCSP by itself does not offer validation of a full certification path/chain. If this functionality were to be provided, additions to OCSP such as DPV and DPD could be used.

The response will also give an indication of the duration of time for which the response is valid – the nextUpdate field gives the time, beyond which, the response should not be considered valid. Any response received by a client with a nextUpdate time earlier than the client local time should be considered unreliable by the client.

5.1.1.2 Security of the Mechanism

OCSP provides server authenticity, as in OCSP it is mandatory for all responses to be signed. OCSP also offers optional client authenticity, in that the client may sign OCSP requests. This could be used if the OCSP responder only wishes to give responses to authorised requesters.

OCSP offers protection against replay attacks by including a nonce within every message sent. The requester includes a randomly chosen nonce in his response, and the responder extracts this nonce and places it in the response. The requester then can check if the packet has been replayed by verifying that the nonce in the response is that sent in the request. The inclusion of the nonce is an optional feature. A replay attack is only of value if the status of the certificate to be checked has changed (e.g. from Good to Revoked) between the genuine generation of the response and its replay by an impostor. Note that the impostor must be able to conduct a man in the middle attack when the request is made.

Time validity in the OCSP responses rely on synchronised clocks between the requester and responder. If this is not the case, then requesters are vulnerable for accepting responses that will have information taken from an older CRL.

Therefore, it can be seen that OCSP overall is only "secure", i.e. that an authenticated certificate status is given by an authorised party if the client clock is "synchronised" to a reasonable degree with the clock of the OCSP responder and also that the nonce method of preventing replay attack is used.

5.1.1.3 Current Support/Hardware support

OSCP has been developed by the IETF PKIX group. Several vendors such as Baltimore, Valicert, VeriSign, Entrust, have implemented OCSP client and server implementations. Client implementations only exists a plug-ins to fixed Internet browsers – there are no mobile-specific implementations as yet.. OCSP requests and responses are encoded using ASN.1 DER encoding. Currently there is limited support for ASN.1 BER on mobile terminals. Support for DER notation, as oppose to BER, is not trivial but the requirement for OSCP is that only a subset of ASN.1 DER encoding is needed in-order to encode and decode OCSP requests and responses. Because terminal manufacturers have already got limited support for ASN.1 encoding it will be easier to implement further ASN.1 support than to support another protocol altogether.

5.1.1.4 Bandwidth considerations

From [18], OCSP uses ASN.1 DER encoding and a typical OCSP signed request has size 449 bytes and unsigned request has size 303 bytes. Responses have size 459 bytes.

5.1.2 XKMS

5.1.2.1 Configuration

XKMS, like OCSP, provides an online certificate revocation checking method. XKMS, however, offers more than just certificate revocation; it can all also check the certificate validity (as described in [2] section 4.9.1) and process a certificate chain path. It also allows for key registration. XKMS is specified in [19].

The setup of XKMS is similar to that of OCSP. It is recommended to use it in the same way as OCSP in scenario two as described above, i.e. with a server separate from the CA acting as the responder.

A client supporting XKMS will have to support the verification of XML digital signatures and will have to support XML. All XKMS responses are signed with XML digital signatures. The revocation status of the public key corresponding to the XKMS signed responses is ambiguous, as the specification does not define a way of validating the corresponding public key certificate, it is simply assumed to be trusted.

XKMS however provides the means of certificate chain processing defined in the XML signature specification; it is also able to retrieve and check revocation for these certificates. XKMS also supports the ability to verify the (duration of) validity of the certificate in question; this is useful, as relying on the client for this will not have to be necessary. Clients may have out of sync clocks or may potentially be attacked so that invalid certificate's may be validated successfully by a client, which has been compromised by a 'Trojan' program showing a false clock. It is recommended that a time stamping authority is used for the synchronisation of clocks, this may be because the terminal does not support a clock and because the clock may be able to be manipulated by a 'Trojan' program residing o the terminal.

5.1.2.2 Security of the Mechanism

XKMS protects against replay attacks by using a transaction ID in each request. The transaction ID is comparable to the nonce issued within OCSP. This is not a mandatory feature within XKMS. The transaction ID should be unique within a client with regard to a particular certificate. The use of the term "transaction ID" suggests that the client must use the transaction ID as a sequence number but in practice the client could just generate a nonce in each case.

There is not a proscribed method in XKMS for indicating to the client which XKMS servers are to be trusted – it is just assumed that the identity, or identities of trusted XKMS server(s) are securely provisioned within the client.

XKMS responses have a limited time validity. The responses is only valid at the client for times (as measured by the client) between the NotBefore and NotAfter fields of the ValidityInterval field in the response.

Hence it can be said that XKMS as a mechanism offers the same level of security as OCSP, i.e. authenticated responses from authorised sources on condition that the client clock is "synchronised" to a reasonable degree with the clock of the XKMS server, that the transaction ID is used to prevent replay attacks and that the "secure provisioning" of trusted XKMS server identities in the client is in fact secure.

5.1.2.3 Current Support/Hardware support

Compared to OCSP, XKMS is a fairly new specification. It has been published to the World Wide Consortium (W3C) as a "technical note", which means it is not a standard as yet.

Vendors such as Entrust and VeriSign offer XKMS prototype implementations where users can download Java toolkits to test the functionality of their XKMS support.

XKMS encoding is carried out using standard XML syntax, thus making it interoperable to all other XML protocols such as XML digital signatures [20]; they XML digital signatures are used within XKMS to sign all responses.

Mobile handset support for XML is non-existent as yet there has never been a requirement to have full support of XML in the wireless world, thus for a short-term solution OCSP maybe more favourable as limited support for ASN.1 encoding is already available on mobile handsets. WAP supporting handsets have support for WBXML, this is a 'compressed' version of XML where the XML is converted into binary digits. This however will not be useable for XKMS.

5.1.2.4 Memory and bandwidth issues

XKMS signed Validate requests have size 2,119 bytes and unsigned requests have size 682 bytes. XKMS provides certificate chain processing too, a locate request (to validate the other certificates in the chain) signed will be 1,988 bytes in size and an unsigned request is 551 bytes in size. Signed OCSP responses and requests are therefore seen to be nearly four timers shorter. The size differences in both XKMS and OCSP are purely based on the encoding and format of the two schemes and not depending on any extra or less security functionality offered. It is clear therefore that, on memory and bandwidth grounds, OCSP requests and responses are the more preferred in the wireless world as they use less bandwidth (typically all responses will be signed to authenticate the responders).

Size is particularly sensitive with regard to revocation checking as it is difficult to allocate the cost of the revocation checking except to the user (the CA will not want to pay as it has no control over how often requests for OCSP checks will be made. It is difficult to assign the cost to the owner of the certificate being checked as this will per transaction billing and this is expensive for the small transaction that is an OCSP check). However, most users have no understanding of certificate revocation checking and so will not want to pay for something they neither understand nor requested. In such a situation, where no party wishes to pay for the check, it is important to keep the costs of such an "unwanted" transaction to a minimum.

5.1.3 Comparison and Conclusion

Signed OCSP responses and requests are seen to be nearly four timers shorter than XKMS messages. The size differences between XKMS and OCSP are purely based on the encoding and format of the two schemes and do not depend on any differences in security functionality offered. It is clear therefore that, on memory and bandwidth grounds, OCSP requests and responses are the more preferred in the wireless world as they use less bandwidth (typically all responses will be signed to authenticate the responders).

(Size is particularly sensitive with regard to revocation checking as it is difficult to allocate the cost of the revocation checking except to the user (the CA will not want to pay as it has no control over how often requests for OCSP checks will be made. It is difficult to assign the cost to the owner of the certificate being checked as this will per transaction billing and this is expensive for the small transaction that is an OCSP check). However, most users have no understanding of certificate revocation checking and so will not want to pay for something they neither understand nor requested. In such a situation, where no party wishes to pay for the check, it is important to keep the costs of such an "unwanted" transaction to a minimum.)

The encoding method for OCSP messages is preferred to that of XKMS as the mobile world has already support for limited ASN.1 encoding support, whereas support for XML within the mobile world is still very scarce. This fact would enable OCSP implementation (particularly on the client side) to be developed more quickly than XKMS implementations.

However, XKMS provides more services than OCSP. OCSP only provides a certificate revocation service, where XKMS provides a whole certificate revocation, validation, key registration solution which will look more favourable as technology improves. The Ericsson report [18] does not take into account DPV and DPD as extensions for OCSP, this will increase the size in bytes for requests and responses over OCSP.

Overall, OCSP seems to be the preferable solution for certificate revocation in the short-term future due to the significant smaller size of its message compared to XKMS and the fact that it can be implemented more easily on the client side.

5.2 Authentication

The issue of authentication has already been discussed in a general way in D04. Authentication was identified as a security-feature generating a basis for many other mostly security-related functionality

such as authorisation or billing mechanisms. Different forms of authentication have been discussed in D04 and several mechanisms and protocols have been briefly introduced. Symmetric have been considered and certificate-based (i.e. PKI-related) formats have been proposed for more global and spontaneous authentication scenarios.

The requirements from WP1 and WP2 for PKI as evaluated in chapters 5 and 6 of D04 have shown the general need for authentication and the different scenarios this feature is needed for.

This subchapter of D07 will focus on the PK-based solutions and the challenges to meet in this context, such as interoperability or revocation-issues. Furthermore the difference between a client-authentication and a 'server'-authentication will be examined.

5.2.1 Non-PK related authentication

Classic authentication procedures involved symmetric, password-based mechanisms. These mechanisms were sufficient as long as the group of users in a system was not too big to manage the keys agreed upon with the peers and as long as no spontaneous communication without meeting the peer in advance was demanded. Many protocols these days still use such symmetric mechanisms. Examples are the PAP/CHAP protocols to dial in at your ISP or even newer protocols as IKE (where pre-shared-secret mode is included beside modes using PK-authentication). The Bluetooth architecture specifies authentication using symmetric mechanisms exclusively.

However the evaluation of the requirements from WP1 and WP2 has shown that many communications between the different parties have to be authenticated by other means as symmetric mechanisms cannot provide appropriate functionality. As already indicated the 4th generation mobile networks will often require more spontaneous mechanisms than bilaterally agreeing on a shared secret before communicating. Former mobile services like GSM or the now introduced UMTS still run on a subscription basis, that allows the use of symmetric keys stored on SIMs/USIMs whereas many services in future generations of mobile networks will not use this model.

Authentication for alternative access scenarios

To build the bridge to the topic 'Access by alternative means' discussed in WP1, some further considerations are given:

It seems feasible that access to future heterogeneous mobile networks is granted on the basis of online payment. Classical GSM networks were designed for post-pay customers who have already subscribed to the service. With the Intelligent Networks (IN) technology, the operators could also offer services to prepay-customers. In future networks online payment can provide additional means for network access. In this scenario, the mobile node is neither authenticated in the access network nor in the home network (i.e. no client authentication). Instead, access depends on the outcome of a payment procedure which is part of (real time) accounting. Access may also be initially granted and later withdrawn when payment units are exhausted. Depending on the means of payment, the mobile user may remain anonymous. Details may be found in WP1 deliverable M1.2, section 3.2.

On the other hand, the access network must identify itself towards the mobile user (server authentication). This requires PK related authentication as discussed below. Here the access network presents a X.509 certificate which needs to be checked by the mobile node. The public key from the certificate will be used to encrypt a session key. The mobile user and the access router use this key to protect their communication.

Additionally, the payment protocols usually have their own authentication mechanisms. These may again use PKI infrastructure, e.g. by securing the credit card number via SSL. The user may also authenticate the payment with a shared secret (e.g. a PIN). Authentication and payment can even coincide when the user submits electronic coins which the access network (in the role of a merchant) checks for validity.

Payment scenarios then require additional communication between the access network and background payment systems. This link also needs to be secured which may involve PK related authentication.

5.2.2 PK-related authentication

PK-related authentication using certificates to transport information about certain parties, i.e. persons or instances, and their cryptographic keys nowadays is a widespread mechanism used to provide flexible mechanisms to verify the binding of key and owner. The most important applications nowadays are certainly SSL/TLS and S/MIME which offer the possibility to authenticate the sender of an e-mail properly.

These applications have shown that the commonly used X.509-certificate-format is still a specification that allows many possible interpretations, so that interoperability is an important issue and may not be neglected.

The adaptation of these formats to fit in restricted mobile environments was e.g. approached in the specification of WAP-specific server-certificates. This adaptation was one step towards the feasibility of PKI-mechanisms in current mobile infrastructures but showed some deficiencies, too.

It is still open how the use of certificates will develop in the near future. As the technical restriction of mobile devices will mostly disappear the protocols and formats will most likely converge to the established internet-standards. One example for this development is the development of WAP 2.0.

5.2.2.1 Client-side-authentication requirements vs. server-side- authentication requirements

Obviously it makes a big difference whether a client wants to authenticate a party in a fixed network or the other way round. This is the case as one can assume different technical infrastructures on the two sides. First of all today's mobile devices are still technically restricted concerning their computing power and their storage space and in addition their connection to the fixed internet is in most cases only temporary and only of low or medium bandwidth (see also D04, chapter 7.3).

To underline this, here is a table with some comparisons referring to the technical restrictions:

Client to Server Authentication	Server to Client Authentication
No restrictions on server-side: Possible use of: • Easy access to databases in fixed networks • Server-located additional information about the client • Sophisticated software handling certificate-issues • Sophisticated revocation handling possible • Root-keys to be loaded authentically in an easy way	Restrictions concerning: • Limited access to databases in fixed networks by the client • Additional information about the authenticated server will generally not be available • Certificate-handling-capabilities are limited • Revocation handling restricted due to limited storage for CRLs or the lack of constant availability of online-connections for online-checking.

We now look at the mentioned issued in more detail:

Access to databases:

In order to obtain or verify certificates different mechanisms can be used. Generally one can distinguish between two different concepts: push- and pull-services.

The use of these concepts depend directly on the application making use of PK-methods and the technical environment. SSL is a typical application that uses a push concept, as the certificates are exchanged during the SSL-handshake as specified in the SSL-protocol. In contrast to SSL the mobile equivalent, WTLS, makes use of both concepts. The server pushes his certificate to the mobile client,

whereas the client has the possibility either to push his certificate too or alternatively only send a reference where to find the certificate, so that the server consequently pulls it.

A pull-service is only difficult to implement for the client-side as the capabilities to find and retrieve certificate-information may possibly be troublesome for the client if not completely automated.

This example shows a typical situation, namely a scenario where we can assume more flexibility and power on the server-side. This situation will appear in the next discussions in a similar way.

Certificate-handling:

The certificate-handling on the client-side has to be done automatically by the application as security-features are not features the user wants to be bothered with. Thus many applications using PK-cryptography and therefore handle certificates, are designed to meet this requirement. Unfortunately this leads to applications, which have only rudimentary interfaces for the user to deal with certificate-issues, so that certificate-formats that are not accepted will lead to major problems. These problems will be hard to solve as firstly the majority of users will not be familiar with certificate-issues in detail and secondly, as already mentioned, the possibilities to solve the problem will be very limited because of insufficiently implemented interfaces.

Certificate handling on the server-side can be done in a more sophisticated way, as software can be designed to be powerful in this context and functionality may be changed or added. Additionally one can assume that specialised stuff is available to deal with certificate-related problems.

Verification and revocation-handling:

The situation concerning this issue is gets us to the same conclusions as the topics before. The implementation of a revocation mechanism is significantly more difficult on the client side than on the server side. This is the case with current CRL based approaches (due to bandwidth-limitations and due to the necessity of a sophisticated logic) as well as with future online approaches as discussed in chapter 5.1.

Root-Key-Issues:

Root-keys as anchors to verify subordinately issues certificates are a crucial and at the same time a delicate issue.

A client-side device may have problems to retrieve certain root-certificates necessary to check other certificates in a reasonable time if they are capable of retrieving them at all. This can be directly related to the certificate handling capabilities of client-software as described above.

Again on the side of the ASP this issue will be less critical as most of the relevant roots will be stored at the ASP anyway and the retrieval is probably less complicated and less time-consuming in fixed networks.

5.2.2.2 Interoperability-Issues

PKI interoperability issues can be split into different categories.

- Interoperability of formats
- Interoperability of applications
- Interoperability between client and the PKI-authorities (management processes like certificate requesting etc.)
- Interoperability of directories

In regard to authentication we will focus on certificate formats and authentication within applications:

Certificate formats:

Will certificates for a mobile environment be of a special format and therefore differ from the ones used already today in a fixed network environment? Today dedicated formats are used due to the

technical restrictions for mobile devices (for instance WAP) as discussed before in this document. Though the future will probably bring the merging of formats and protocols for authentication for the different technical scenarios. This may be different for authorisation as this issue is often considered for restricted solutions. A broader range of possible requirements may lead to different solutions using different certificate based approaches. Thus authorisation may be a field to think about new certificate-formats.

Even if this merging will happen, a most delicate thing to consider when establishing a PKI is to specify the certificate-format in such a way, that it can be verified by most software-modules. The X.509-standard as specified by the ITU and adapted by the IETF is still a pretty open standard and time has shown, that there are many degrees of freedom for defining an X.509-certificate.

Not only the much talked about extensions introduced in the third version of X.509 are a potential risk to lose interoperability, but also for example the use of the DN. Certain protocols require certain attributes included in the certificate. A good example from the fixed network is the mandatory matching-feature of S/MIME, that means that the sender e-mail-address of an S/MIME secured e-mail has to be compared to the address included in the certificate by the receiving agent. The certificate can provide the address in two different manners: by inclusion in the DN-field of the certificate-owner or by inclusion in the 'subject alternative name' field.

Certainly it is not possible to specify a strict certificate profile that meets all the requirements for different PKI scenarios and makes different interpretations impossible, but some general conclusions can be drawn:

- Certificates should be specified with as little parameters as possible to improve interoperability with other solutions.
- Only important major extensions should be used and the criticality bit should only be true if it is really necessary to do so
- The introduction of private extensions which are set critical is not recommended at all.

Applications:

Applications dealing with certificates as a mechanism to provide authentication will be located on client and on server side. Different kinds of authentication have to be considered:

- Entity authentication: Explicit authentication with dedicated protocols as for instance connecting to a network
- Origin-related authentication: Implicit authentication, for instance by signing a document or a file.

Even when the certificates used to authenticate conform to the standard they are based on, this does not mean, that the authentication by the corresponding software-module will be successful. The implementation of the protocol in the application might lead to further problems.

A good example is the incompatibility of certain Microsoft applications concerning large public exponents leading to wrong interpretations of the parameter.

As authentication will probably be performed with standard software components in many scenarios, thorough testing has to take place before launching a service. A further problem could be, that, at least on the client side, you won't have a possibility to patch flaws in applications or even change them.

5.3 Authorisation

Authorisation or access control prevents unauthorised use of network resources. Access to network resources must be restricted and conform to the security policies in place. If attackers were to get unauthorised access to any of the network resources, various attacks, like denial of service, eavesdropping or masquerade, could follow. Therefore, when a mobile node performs global roaming among various heterogeneous networks authorisation is a mandatory security feature [21].

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 certificates in different formats and/or conforming to different policies and profiles. Therefore it may be expected that Users and Application Service Provider (ASP) may need to verify certificates in a form different to that created by their 'own' CA(s).

Authorisation is always coupled with authentication: prior to allowing access to special network resources the entity that requests the access is authenticated by the owner of the resource. The authorisation method may depend on how the authentication is done. In the case authentication is done based on certificates, then it is likely that the authorisation is also based on certificates. Therefore, authentication issues are considered as well.

Issues considered are the interoperability of public key certificates and PKIs when public key mechanisms are used to accommodate the authentication and authorisation requirements that are identified in D04.

5.3.1 Non-certificate based access control methods

Although these access control methods do not require the use of public cryptographic mechanisms, they are briefly described to be able to compare them with certificate based access control methods. A more detailed investigation regarding the use of non-certificate based access control methods can be found in WP1 and WP2 deliverables.

5.3.1.1 Use of Access Control Lists (ACL)

Authorisation or access control to resources (files, memory, application software, printer, node in a network…) can be done by the use of ACLs storing information concerning the rights or the role of a user. The role can then be used to look up the actual rights

This mechanism, of course, is related to the authentication of the requestor, so that the service-provider is sure about the party asking for access and can look up possible additional attributes and the corresponding rights in his database.

An advantage of the use of ACLs is definitely, that the requestor does not have to deliver any additional credentials to his authentication-information, hence no additional software is necessary on the client side.

The ACL on the providers side can either be stored locally or on a central server. The solution to chose depends on the service offered to the customer. If the authorisation-information is very specific to the service offered, it makes sense to store it locally (e.g. file ACLs may be stored on the same file system as the objects to be protected). The central approach is more favourable if the access depends on information that is of use for many access-points. Such information could for instance be the role of a user in a company. Another reason to store such information centrally is, that changes of the access rights associated with the role do not have to be communicated to all affected systems as they are pulling this information anyway from the central server. Certainly with a central storage one has to consider this database as a bottleneck and a single point of failure, too.

The use of ACLs in mobile scenarios could be considered by service-providers, as the server-side-technique is already well-established in fixed scenarios and can be transferred to mobile clients easily. Nevertheless this model fits only in scenarios, where the requesting party is already known by the provider or by a network, the provider participates in.

5.3.1.2 Use of non-cryptographic credentials

Besides credentials based on cryptographic mechanisms, there are 'out-of-band-alternatives' that can be used for authorisation. In many scenarios today credit-card-numbers are used to make deals over the internet.

One can think of using authorisation information in connection with such alternatives to do a kind of authorisation. One example could be, that a user transmits his credit-card number in a signed message

that includes his commitment to pay for a certain service via his credit-card account. Certainly a lot of legal questions have to be considered, which are out-of-scope in this chapter.

5.3.2 Certificate based access control

This subchapter tries to point out possible mechanisms in public key cryptography to provide access control, i.e. authorisation. The focus here is not on the mechanisms themselves (which are introduced only in a brief way) but the evaluation of their suitability and the ease of implementation.

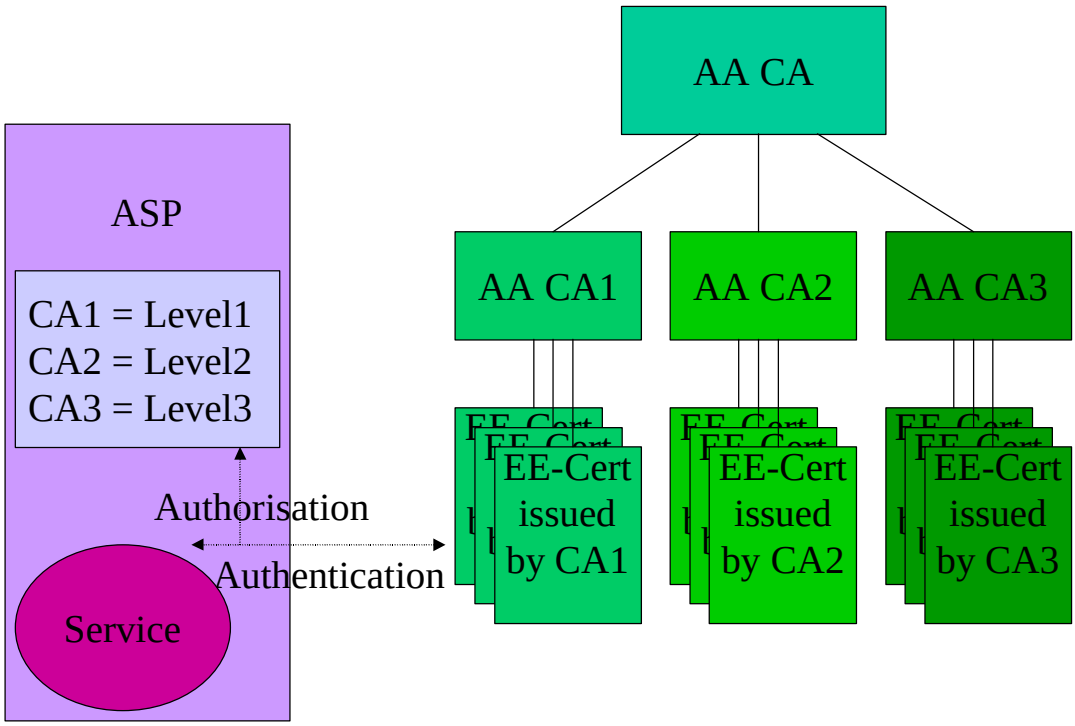
5.3.2.1 Use of virtual CAs for authorisation

A pragmatic approach to implement a solution for authorisation with standard methods is the following:

Suppose you have a authorisation-concept that grants certain role-based access-levels. A user gets a certificate due to his/her role. A certificate-issuer in this scenario has a number of virtual CAs located below another CA owned by the same issuer or by another authority.

Due to the access-level the user is granted, his certificate is issued by one of the virtual CAs that is related to this access-level. The certificate, first-hand only used for authentication-purposes now includes some implicit authorisation-info. Of course, the party providing access has to be able to interpret this information in the right way to grant the right access level to the user. The advantage of this approach is, that there is no explicit, additional information needed in the certificate, so that interoperability issues in respect to authorisation-info has not to be considered.

As the authorisation is done via ID-based certificates the change of the authorisation-level of a user induces the revocation of the ID-certificate. Therefore this solution makes only sense in scenarios where the user does not change his role frequently.



5.3.2.2 Use of authorisation-related extensions in Public Key Certificates

Certificate extensions and authorisation may be closely related to each other, as the extensions in ID-based X.509 certificates may be used for authorisation purposes.

Systems authenticating users via the verification of an X.509-certificate may use additional information given by certain extensions to grant some level of access to the user. The extension(s) used for this purpose will be presented to an application as private extensions. Of course, applications have to be able to evaluate these extensions and interpret them in the right way. Here again, like discussed in the chapter on authentication, the situation on the server-side and the situation on the client-side differ substantially. Whereas one can presume a server-side-system to be able to evaluate such extensions or at least to offer the possibility for such a feature to be implemented, a client-side engine will probably not be able to perform such actions natively, and as experienced in the past, client-side modifications are usually not accepted for an 'open' environment. Thus the management of access to the mobile device by applications will probably not be implemented with this solution.

Therefore a proprietary solution using private extensions is only favourable to do client-authorisation.

As already described in D04 [2], an extension is a triple consisting of the extension-identifier, the extension-value and the criticality-bit.

In this context the criticality-bit has not to be set true, so that an application not capable of evaluating this extension is not forced to reject the whole certificate.

Another important point to consider is how to make sure, that only accepted CAs will issue certificates with the authority-relates extensions. To assure the proper granting of rights the evaluation, the server-side system has to relate the authentication and the authorisation directly. Otherwise the following scenario could arise:

The server-side party accepts several CAs for issuing certificates used for authentication. To get special rights, some of the issuers have also the right to include private extensions as described above. Other CAs don't have these privileges. Now, a malicious CA not entitled to issue authorisation-certificates could include the relevant extensions nevertheless. In this case, if authentication and authorisation are not related directly, the client using the faked certificate is granted a wrong level of access. Certainly, the issue of trust is a delicate issue and one could argue, that either you trust a CA or not, but in this scenario it's the question of prevention of misuse or reaction to misuse. If you don't relate authentication and authorisation directly you can only be reactive, if you do a preventive security is available.

To close this section on authorisation using ID-certificates the combination of authentication-features and authorisation-features in one certificate is briefly discussed:

A problem that arises in this combination is, that attributes concerning certain rights often change before the ID-based certificate's lifetime expires. As the authorisation-related extensions are an inherent part of the certificate and may not be excluded the consequence in such a situation is the revocation of the whole certificate.

5.3.2.3 Use of Attribute Certificates (AC)

The use of dedicated attribute certificates to enforce access control is a very sophisticated way of managing the authorisation-issue. Here, instead of including authorisation-info in ID-related certificates, attributes are certified in a separate data-structure. This mechanisms has certainly advantages and disadvantages:

Attribute certificates can be issued without any initial identification process of the requester as the ID-related certificate of the requester is used as an 'anchor'. Therefore attribute certificates can be issued on the fly. Another advantage arising out of this fact is, that attribute certificates can be issued for short validity times. That means they are very practical in cases where roles or attributes change frequently or for one-off access like for instance pay-per-view events.

The major disadvantage is, that the handling of attribute certificates is more complex than the other approaches discussed above on server-side and on client-side. Additional logic has to be implemented.

Attribute certificates is a much talked about concept already implemented by big PKI-players. Nevertheless there are no stable standards yet, so that the usage of attribute certificates at present is not advisable in open user-groups. A profile for attribute certificates is standardised at the IETF. Presently the document is at a draft status version 9.

5.3.2.4 Use of SPKI Authorisation Certificates

SPKI was roughly specified a few years ago at the IETF. Unfortunately only little of the efforts were implemented. Only few documents are available, stating the ideas of SPKI. Recently the transformation of SPKI certificates into XML documents and vice versa has come up again at the IETF. The approach is different from the one proposed early 2000.

In contrast to X.509 and the PKIX standards, SPKI was developed primarily for the deployment of authorisation and not for authentication purposes. Instead of binding a key to a certain identity, SPKI's intention was to bind a public key to certain attributes. The concept is to identify a keyholder by the public key itself. Furthermore SPKI does not promote public directories as they are commonly used for example in PKIX-scenarios. The owner of the certificate may decide whether to publish it or not. Nevertheless SPKI offers the binding of a key to a 'local' name in addition.

Requirements to the structure of SPKI-certificates are:

- Minimal number of fields in the certificate
- Described in an easy way
- Written in a way that makes difficult parsing redundant

As SPKI certificates usually only mean something to the issuer and the owner of a certificate and the certificates are pushed by the user to the access granting party, no extensive infrastructures, especially no directory-services have to be implemented and maintained.

In interesting aspect covered by the SPKI-WG is the one of delegation. Different models of delegation were discussed and the group agreed on 'Boolean control'-model, which means that either a subject may grant a delegation of the attributes of a certificate to a delegate or not. No maximum delegation depth is specified here as it was agreed upon that such a number would not be foreseeable at the time the original certificate was generated.

As with the concept of delegation, many people are in the position to grant certain rights (if they possess these rights themselves) the danger arises, that a person in the chain of delegation gives certain permissions to people that a person being more up in the chain might not be happy with. The distribution of certain rights conforming to a certain policy might be undermined by people by accident (for instance not really knowing the policy) or on purpose.

The concept of SPKI fits in 'closed' environments, i.e. in scenarios where only 'selected' parties take part in.

[1] Another interesting aspect is the anonymity of users that can be achieved by using SPKI. Whereas the issuing party may still hold corresponding list of certificates and users, the certificate itself sent over the internet does not state any user-related information.

5.4 PKI for limited devices

It is expected that some of the requirements posed by WP1 and WP2 will involve public key operations such as signature generation and verification. It is well known that public key operations are computationally expensive, and therefore require a considerable computing power on the side of the device performing them. Furthermore, a public key solution typically requires a PKI, which presents requirements on the processing power and storage capacity of the device, as well as requirements on the bandwidth of the wireless link. In this document, we attempt to identify the types

of public key operations that the limited device has to perform, and list the main issues related to their limited resources.

5.4.1 Public key operations

The requirements posed by WP1 and WP2 are considered in D07, and are clustered according to the type of feature that is required for their satisfaction (e.g., authorisation, authentication), and their suitability for a PK solution.

5.4.1.1 PK operations for securing heterogeneous access to core networks

We start by listing the main types of public key operation, and for each we list the main steps involved.

1. **Digital signature generation**
    a. Execute a signature generation algorithm (computationally expensive)
    b. Send a set of public key certificates (memory, bandwidth expensive)
2. **Digital signature verification**
    a. Check the revocation status of the certificate (computationally and memory expensive; there is also the latency issue with certificate revocation checking mechanisms, both in terms of client timers and in making the user wait for application installation until the check is done)
    b. Identify a certificate chain for validating the certificate (computationally, memory, bandwidth expensive)
    c. Execute a signature verification algorithm (computationally expensive)
3. **Presentation of credentials for authorisation purposes**
    a. Execute a signature verification algorithm (computationally expensive)
    b. Send an authorisation certificate (attribute certificate, SPKI certificate) (memory, bandwidth expensive) (computationally expensive?)
4. **Public key encryption**
    a. Obtain a public key certificate for the public key needed to perform the encryption (memory, bandwidth expensive)
    b. Check the revocation status of the certificate (computationally and memory expensive; there is also the latency issue with certificate revocation checking mechanisms, both in terms of client timers and in making the user wait for application installation until the check is done)
    c. Identify a certificate chain for validating the certificate (computationally, memory, bandwidth expensive)
    d. Execute a public key encryption algorithm (computationally expensive)
5. **Public key decryption**
    a. Execute a public key decryption algorithm (computationally expensive)
6. **Asymmetric key establishment**

    Note that the precise set of steps for this operation varies widely depending on the nature of the key establishment protocol used. We list below some steps common to a significant number of such protocols.

    a. Exchange public key certificates for the respective public key agreement keys (memory, bandwidth expensive)

b. Check the revocation status of the respective certificates (computationally and memory expensive; there is also the latency issue with certificate revocation checking mechanisms, both in terms of client timers and in making the user wait for application installation until the check is done)

c. Identify certificate chains for validating the respective certificates (computationally, memory, bandwidth expensive)

d. Prepare key establishment messages (computationally expensive)

e. Exchange key establishment messages (bandwidth expensive)

f. Compute the shared key (computationally expensive)

7. Generation of asymmetric key pairs

a. Generation of random numbers (computationally expensive)

b. Use of random numbers, e.g. to generate random primes (computationally expensive)

5.4.1.2 PK operations for secure execution environment

The requirements appear similar to those for 5.4.1.1. However, in this case it is possible that there is only a need for signature verification.

5.4.1.3 PK operations for inter-PAN authorisation and communication security

The requirements appear similar to those for 5.4.1.1.

5.4.2 Issues for further investigation

In the previous sections, we identified the PK operations that may have to be performed by, or on behalf of, the limited device. In this section, we consider each one of them, and list the related issues that have to be investigated.

5.4.2.1 Signature generation

Signature generation requires the signer's private key and the data to be signed to be available for processing. Because of the sensitivity of the private key this will often be held in a physically secure subsystem, e.g. a chip card. In such a case there are two main options for computing the signature:

a) Transfer the data to be signed into this subsystem, or

b) pre-processing the data to be signed externally to the subsystem, and then only transferring a data-dependent value (e.g. a hash-code) to the subsystem for signature computation.

Option (a) is 'more secure', although (b) is often preferred to minimise the computation requirements for the subsystem, and most importantly, to reduce the amount of data transferred to and from the subsystem.

The following issues arise when considering the generation of signatures in very limited devices.

1. What are the preferred signature generation algorithms in terms of efficiency? There are a number of choices here, e.g., RSA, ECDSA, DSA, etc.

2. Who performs the signature generation? It is conceivable that some trusted server does the actual computation on behalf of the limited device in the case where the limited device both

   a) lacks the ability to perform signature computations, and

   b) needs to compute a signature in order to meet the security requirements of some external entity.

   If such an approach is followed, then what are (a) the trust implications, and (b) the computational and bandwidth ramifications? Also, what are the ramifications of sharing the computation over more than one device, as discussed above?

Further, the trust implications will vary depending on the nature, ownership and location of the trusted server. In the 'most trusted' case, the server could be a PDA computing a signature for another mobile device within the same PAN (and worn by the same individual). More generally, one device within a PAN could act as the PK server, responsible for performing all PK and PKI computations for all other devices within the PAN. Should the security architecture for future mobile terminals consider security issues associated with transferring cryptographic computations between devices within a PAN? Or is this just another service?

Finally, if third party signature computation seems to be a viable possibility, are there existing protocols for providing such a service? If not, then do we need to define such a protocol?

3. What certificate formats are used? That is, what are the computational, storage and bandwidth implications of choosing one format over another? Note that, in practice, X.509 is essentially forced upon us – nothing else is likely to be accepted by the standards bodies, unless there is a very small forum (e.g. the WAP forum when it was just four companies).
4. Where is the certificate of the user of the limited device stored? It is possible that the certificate is stored at a server, and is retrieved by the entity that receives the signature, thus saving storage at the limited device, and wireless bandwidth. Again, what are the trust ramifications of separate storage.

5.4.2.2 Signature verification

The following issues arise when considering the verification of signatures in very limited devices.

1. How is the revocation status of the received certificate checked? Besides a CRL, there is also the option of using an OCSP service. What are the bandwidth and trust implications of such a move? Would it be sensible for a device within a PAN to provide an OCSP service to other devices within the same PAN? Or would a more lightweight protocol be more appropriate?
2. Who constructs the certificate chain that leads to the root key? It is conceivable that a trusted server may retrieve the required certificates. However, if this approach is followed then we need to consider at what stage this might be done, i.e. after the download of an EE cert to the device, or prior to a whole chain being delivered to the device.
3. Who performs the actual verification of the signature? Again, a trusted server may do the verification on behalf of the limited device. Again, what are the bandwidth and trust implications of such a move? Again, would it be sensible for a device within a PAN to provide a signature verification service to other devices within the same PAN? As for signature generation, do there exist protocols to provide such a service, or does a new protocol need to be defined?
4. The issue of the choice of signature generation/verification algorithms is relevant here too.

5.4.2.3 Authorisation credentials

The following issues arise when considering the use of authorisation credentials in very limited devices.

1. What types of certificates are used for authorisation purposes? There is the choice of attribute certificates, SPKI certificates (more?)
2. In the case of SPKI certificates, who constructs the required list of certificates?
3. Who does the actual generation of the digital signature?

5.4.2.4 Public key encryption

The following issues arise when considering the use of public key encryption in very limited devices.

1. Where is the required certificate obtained from?
2. As for signature verification, how is the revocation status of the received certificate checked? Besides a CRL, there is also the option of using an OCSP service. What are the bandwidth and trust implications of such a move? Would it be sensible for a device within a PAN to provide an

OCSP service to other devices within the same PAN? Or would a more lightweight protocol be more appropriate?

3. As for signature verification, who constructs the certificate chain that leads to the root key? It is conceivable that a trusted server may retrieve the required certificates. However, if this approach is followed then we need to consider at what stage this might be done, i.e. after the download of an EE cert to the device, or prior to a whole chain being delivered to the device.
4. As for signature generation and verification, which public key encryption algorithm should be used?

5.4.2.5 Public key decryption

The following issues arise when considering the use of public key decryption in very limited devices.

1. (As for signature generation and verification), which public key encryption algorithm should be used?

5.4.2.6 Asymmetric key establishment

The following issues arise when considering asymmetric key establishment in very limited devices.

1. As for signature verification, how is the revocation status of the received certificate checked? Besides a CRL, there is also the option of using an OCSP service. What are the bandwidth and trust implications of such a move? Would it be sensible for a device within a PAN to provide an OCSP service to other devices within the same PAN? Or would a more lightweight protocol be more appropriate?
2. As for signature verification, who constructs the certificate chain that leads to the root key? It is conceivable that a trusted server may retrieve the required certificates. However, if this approach is followed then we need to consider at what stage this might be done, i.e. after the download of an EE cert to the device, or prior to a whole chain being delivered to the device.
3. As with all previous cases, which key establishment protocol is optimal with respect to limited devices?

5.4.2.7 Generation of asymmetric key pairs

The following issues arise when considering the generation of key pairs in very limited devices.

1. Can the key generation task safely be delegated to other entities in the PAN?
2. As with all previous cases, which algorithm is optimal with respect to the ease of key generation?

5.4.3 The way forward

Further more detailed analysis is required of the questions raised above. Perhaps the most fundamental question is whether existing techniques and protocols meet all the likely needs, or whether new techniques and/or protocols are needed. If existing solutions are appropriate, then what options are there, and what are their relative advantages and disadvantages?

Finally, are there new insights to be gained by looking at the trust implications of the various solutions (existing and novel)? For example (and this may be trivial to answer) what are the **trust** differences **in a mobile context** between relying on an entity providing

a) a key pair generation service;

b) a CA service (i.e. trusting all certificates generated by this CA);

c) an OCSP or other public key/certificate validation service;

d) a signature generation or a signature verification service;

e) a public key encryption or a public key decryption service?

Do these trust differences vary depending on the nature of the entity providing the service? For example, if the entity is a trusted device within a PAN, do any differences between (a), (c), (d) and (e) disappear?

References

[1] Shaman D13, *Final Technical report comprising the complete technical results, specification and conclusion*. Not yet available.

[2] Shaman D4, *Initial report on PKI requirements for heterogeneous roaming and distributed terminals.* 10th September, 2001.

[3] Shaman, *Intermediate deliverable for Milestone MS1.2*. 17th January 2002.

[4] Shaman MS2.2, *Intermediate requirements and functional specification for security architecture for distributed mobile applications and service access*. December 2001.

[5] C. Gehrmann and K. Nyberg, 'Enhancements to Bluetooth baseband security', in: *Proceedings of Nordsec 2001, Copenhagen, November 2001*.

[6] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[7] D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing", in *Advances in Cryptology – CRYPTO 2001*, (Springer LNCS **2139**), 2001, pp.213-229.

[8] C. Cocks, "An Identity Based Encryption Scheme Based on Quadratic Residues''. In: B. Honary (ed.): *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings* (Springer LNCS **2260**), 2001, pp. 360-363.

[9] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity", *J. of Cryptology* **1**, 77-94, 1988.

[10] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems", in A.M. Odlyzko (ed.), *Advances in Cryptology – CRYPTO '86*, (Springer LNCS **263**), 1987, pp.186-194.

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

[12] ShamanD3, *Security Architecture for Future Terminals and Applications.* 29th June, 2001.

[13] 3GPP TS23.057, "Mobile Station Application Execution Environment (MExE)".

[14] *WAP Signed Content*, WAP-233-SCONT-20010531-t, Prototype Version 31st May, 2001. At http://www1.wapforum.org/tech/terms.asp?doc=WAP-233-SCONT-20010531-t.pdf

[15] *Mobile Information Device Profile Next Generation (MIDP NG).* At http://www.jcp.org/jsr/detail/118.jsp

[16] *Wireless Identity Module* (WIM), WAP-260-WIM-20010712-a, Approved Version 7th June, 2001. At http://www1.wapforum.org/tech/terms.asp?doc=WAP-260-WIM-20010712-a.pdf

[17] *Online Certificate Status Protocol*, RFC 2560, Malpani, Galperin, Adams. At www.ietf.org/rfc.html

[18] *Evaluation of certificate validation mechanisms*, Hormann, T.P., Wrona, K., Holmanns, K., Ericsson Research. Submitted as input to the WAP Forum, September 2001.

[19] *XML Key Management Specification (XKMS 2.0)*, W3C Working Draft 31 January 2002. At http://www.w3c.org/2001/XKMS/Drafts/XKMS-20020131.

[20] *XML-Signature Syntax and Processing*, W3C Recommendation 12 February 2002. At http://www.w3.org/TR/2002/REC-xmldsig-core-20020212.

[21] Shaman D2, *Intermediate Report: Results of Review, Requirements and Reference Architecture*, 29th June, 2001.