IST-2000-25350 - SHAMAN

Deliverable Number	D13 – WP3 contribution
Deliverable Title	**Final technical report**
Date of delivery	23-Nov-2002
Document Reference	SHA/DOC/RHUL/WP3/D13A3/1.0

Contractual Delivery Date	
Actual Delivery Date	30-Nov-2002
Editor	Chris Mitchell
Participant(s):	Vodafone, Royal Holloway, Siemens ATEA, T-Nova, Nokia
Workpackage	WP3
Est. person months	
Security	Public
Nature	Report
Version	1.0
Total number of pages	108

Abstract:

Annex 3 of D13 provides an analysis of PKI issues arising from the other Shaman workpackages (WPs). PKI issues likely to be of importance to future mobile systems are also analysed, and results given.

Keyword list:

public key cryptography, mobile telecommunications, PKI, public key certificate, mobile terminal

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

Executive summary

In this part of D13 we provide an analysis of PKI issues for which an investigation was explicitly requested by other Shaman workpackages (WPs). PKI issues likely to be of importance to future mobile systems were also analysed, and the results are given here.

With regard to the PKI issues for WP1 (secure access from mobile terminals to heterogeneous access networks) the major issue of mobile node (MN) to access network (AN) authentication and key agreement arises. If this authentication process uses asymmetric cryptographic methods, then the resulting requirements for PKI are simplest if the MN's certificate is issued by its home network. This is the case in the traditional subscription scenario. Moreover, issuance by the home network potentially provides the MN with greater identity privacy towards the AN provider. Apart from the traditional subscription case, the alternative access case provides network access to users that pay by ad hoc methods (using e-payment mechanisms) for services received by the access network. In this case, there is no home network involved in the authentication and key agreement process and PK mechanisms are used to secure the (partly) wireless link between MN and access router (AR) during initial access. Public key based authentication and key agreement protocols are considered for both cases.

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 considered.

- '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 also 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
Dent	Alex	Royal Holloway, University of London (RHUL)		+44 1784 443094
Garefalakis	Theo	Royal Holloway, University of London (RHUL)		+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

The purpose of this annex to Deliverable D13 is to report on the main research findings from workpackage 3 (WP3) in the second year of the SHAMAN project. Whilst this document includes all the main results of one previous deliverable from WP3 (namely D07 [40]), it does not reproduce the findings of the initial survey of PKI requirements and technologies (namely D04 [39]). This annex should be read as a sequel to D04.

1.1 Scope and purpose

The purpose of this final 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 final results of the WP3 analysis on the issues covered.

1.2 Contents of this report

In the initial WP3 deliverable, D04 [39], the security requirements listed by WPs 1 and 2 in deliverables D02 and D03 respectively [37], [38] were analysed for potential satisfaction of the requirements using asymmetric cryptographic techniques. Further, D04 [39] 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 D04 [39] 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. Chapters 2 and 4 are examples of this, having been developed in response to specific requests from WP1 and WP2 respectively.
- 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 D04 [39] would benefit from some further analysis. An example of this would be Chapter 5, where PK issues relating to Secure Execution Environments are considered.
- Further analysis of the general PKI issues raised in D04 [39], for instance, Chapters 3 and 7 where revocation and authentication/authorisation issues are considered.

In summary, the main parts of this document are as follows:

- Chapter 2 analyses the PKI requirements and PK techniques necessary for performing network access to foreign access networks in the traditional subscription case (assuming that PK techniques are to be used for this purpose) as well as the alternative access case.
- Chapter 3 considers issues relating to public key revocation in a mobile environment.
- Chapter 4 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.
- Chapter 5 analyses the PKI requirements of WP2 in relation to secure execution environments.
- Chapter 6 contains an analysis of PK and PKI issues relating to the implementation of PK algorithms on limited devices.
- Chapter 7 considers the issues relating to PKI support for the security services of authentication and authorisation.
- Chapter 8 provides general conclusions and possible future research areas.

1.3 List of abbreviations

Table 1 – List of abbreviations

Term	Definition
AAA	Authentication, Authorisation and Accounting
ACL	Access Control List
AH	Authentication Header
AN	Access Network
AR	Access Router
ASN.1	Abstract Syntax Notation 1
ASP	Application Service Provider
BER	Basic Encoding Rules
CA	Certification Authority
CCM	Certificate Configuration Message
CPS	Certification Practice Statement
CRL	Certificate Revocation List
DER	Distinguished Encoding Rules
DNS	Domain Name Service
DoS	Denial of Service
DPD	Delegated Path Discovery
DPV	Delegated Path Validation
DSS	Digital Signature Standard
ESP	Encapsulated Security Payload
GPRS	General Packet Radio Service
GSM	Global System for Mobile telecommunications
HMAC	A MAC scheme based on the use of a cryptographic hash-function
HTTP	HyperText Transfer Protocol
ID	Identifier
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange (an authentication and key establishment protocol)
IP	Internet Protocol
IPsec	IP security
JFK	Just Fast Keying (an authentication and key establishment protocol)
MAC	Message Authentication Code
ME	Mobile Equipment
MExE	Mobile Execution Environment

MIDP	Mobile Information Device Profile
MN	Mobile Node
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)
PKG	Private Key Generator
PKI	Public Key Infrastructure
PKIX	The family name for the IETF PKI standards
PRF	Pseudorandom Function
PSD	PAN Security Domain
QoS	Quality of Service
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
SEE	Secure Execution Environment
SHA-1	Secure Hash Algorithm revision 1 – a standardised cryptographic hash-function
SIM	Subscriber Identity Module
SM	Security Module
SMS	Short Message Service
SOI	Son-of-IKE
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
WP	WorkPackage
WTLS	Wireless Transport Layer Security
XKMS	XML Key Management System
XML	eXtensible Markup Language

2 Public key based network access

Access to foreign access networks is the main focus of WP1. Access can be based on either secret key or public key techniques. In this chapter, we consider the cryptographic requirements for public key based network access and evaluate the two public key authentication protocols chosen for further consideration by WP1, namely JFK and IKEv2. Based on this evaluation, we give a recommendation for the network access case.

2.1 Requirements for PK based network access

In this section, we consider the security and privacy requirements that a public key authentication protocol must provide for access to a foreign access network. There are two cases: the "traditional" subscription case, where a user has a subscription with a home operator and, as a consequence thereof, a long-term trust relationship with this home operator, and the alternative access case, where the user pays ad hoc (using e-payment mechanisms) for services received by the access network. The requirements for the subscription case are considered in section 2.1.1; the differences for the alternative access case are discussed in section 2.1.2. The discussion of requirements will help us to analyse the suitability of various candidate protocols that we describe in the next sections (section 2.2 and 2.3). The assessment of the protocols regarding the requirements formulated for the subscription and alternative access case is given in section 2.4.

2.1.1 Subscription case

We can divide the security requirements for a public key authentication protocol into five main categories. Each category is considered separately below.

- Entity authentication requirements;
- Key establishment requirements;
- Payment/QoS requirements;
- Privacy requirements;
- Non-repudiation requirements.

In the discussion below, we look at all the possible requirements in each of these classes, and consider whether they are likely to be genuine requirements in the network access scenario.

2.1.1.1 Entity authentication requirements

There are two main potential entity authentication requirements:

- authentication of the MN to the access network, and
- authentication of the access network to the MN.

The first possible requirement, namely *authentication of the MN to the access network*, is almost certainly a genuine requirement for network access. The access network will want to be sure that it is communicating with a genuine MN, otherwise there is a danger that a spurious MN will be able to fraudulently gain a level of service without ever intending to pay for this service.

The need for the second possible requirement, namely *authentication of the access network to the MN*, is not so obvious. The need for this direction of authentication depends very much on what else occurs during the authentication exchange. If, for example, the prime role of the exchange is to set up a shared secret key then, as long as there are appropriate guarantees for the key, this direction of entity authentication may not be required.

To see why this may be true we need to consider the service provided by entity authentication. Entity authentication is usually defined as the verification, at a point in time, of the claimed identity of a communicating party. Hence, *authentication of the access network to the MN*, will provide verification, at an instant in time (i.e. the moment when the protocol is executed) of the identity of the access network to the MN. Whether or not this is necessary depends very much in what execution of the protocol implies to the MN. If, for example, immediately after the protocol executes the MN takes actions which it would not wish to do if the access network was not as claimed, e.g. modifying internally stored parameters, then clearly authentication is necessary. However, if, as is often the case in such a scenario, the protocol is merely a way of establishing a session key with the access network, then the MN will be happy to use the session key with no explicit entity authentication as long as he/she knows that the session key is only available to the correct access network (this amounts to implicit key authentication). The MN can, for example, send sensitive data to an unauthenticated access network as long as it is encrypted under an appropriately established session key, in the knowledge that, if the access network is false, the access network will not be able to decrypt the sensitive data.

2.1.1.2 Key establishment requirements

There are several potential key establishment requirements, as follows. Note that the terminology used here is based on that defined in [19] (see also [23]).

- agreement between the MN and the access network on one or more shared secret keys,
- implicit key authentication from the MN to the access network for the shared secret key(s),
- implicit key authentication from the access network to the MN for the shared secret key(s),
- key confirmation from the MN to the access network for the shared secret key(s),
- key confirmation from the access network to the MN for the shared secret key(s),
- freshness guarantees to the MN for the shared secret key(s),
- freshness guarantees to the access network for the shared secret key(s),
- forward secrecy with respect to the private key of the MN for the shared secret key(s),
- forward secrecy with respect to the private key of the access network for the shared secret key(s), and
- key control requirements for the shared secret key(s).

Note that implicit key authentication and key confirmation implies explicit key authentication (see [19]).

The first possible requirement, namely *agreement between the MN and the access network on one or more shared secret keys*, is a definite requirement. It is an implicit assumption that secret session keys are established, which can then be used to protect data and signalling traffic exchanged between MN and access network. We now examine the other possible requirements.

- The two possible *implicit key authentication* requirements are definitely needed in the this scenario. Indeed, it is difficult to imagine situations where a session key is established and implicit key authentication is not required.
- The two possible *key confirmation* requirements do not appear necessary. As and when the keys are used by the parties, key confirmation will be provided.
- The two possible *freshness guarantee* requirements are again definitely needed. Without these guarantees the possibility of re-establishing old (and potentially compromised) keys exists, which will potentially compromise subsequent exchanges between MN and access network.
- The two possible *forward secrecy* requirements appear relatively unimportant. Whilst forward secrecy would certainly be an advantage, it is also probably of relatively low importance in the network access scenario. The potential for future fraud arising from private key compromise is

almost certainly much more serious than the possibility of decrypting ‘old’ encrypted exchanges between MN and access Network.

- It is also not clear whether it matters which of the MN and access network has *key control* (or whether they have it jointly, as in the case where a key agreement technique is used).

2.1.1.3 Payment/QoS requirements

The possible security requirements in this category are as follows.

- Secure agreement of the means of payment for service,
- Secure agreement on a QoS level.
- Secure agreement on a rate of payment for service provision.

The necessity of these potential requirements depends very much on details of the network access scenario.

2.1.1.4 Privacy requirements

We can identify the following possible privacy requirements:

- Confidentiality of the identity of the MN on the radio interface,
- Prevention of linking between pairs of the authentication exchanges involving the same MN, and
- Confidentiality against radio interface eavesdropping for data exchanged during the authentication protocol.

The necessity of the first privacy requirement, i.e. *Confidentiality of the identity of the MN on the radio interface* is well-established for scenarios as these. The same is true of the second requirement, i.e. *Prevention of linking between pairs of the authentication exchanges involving the same MN*. The need for the third requirement (*Confidentiality against radio interface eavesdropping for data exchanged during the authentication protocol*) very much depends on the type of data transmitted across the radio interface. For example, if the protocol involves exchanging requested QoS parameters, then, if these are sent in cleartext, a breach of privacy might result for those users who are known to request unusual QoS parameters.

In general, the confidentiality of the MN identity is vulnerable against active and passive attacks. According to Aiello et al. [2], an active attack involves an adversary who modifies or injects messages and a passive attack involves an adversary who attempts to defeat a cryptographic technique by simply recording data and thereafter analyse it. In our scenario, the active attacker could initiate the public key authentication protocol with the MN in order to reveal the MN’s identity and the passive attacker could eavesdrop messages sent between the MN and the access network. In the analysis section, we will assess the public key authentication protocols with respect to both active and passive attacks.

2.1.1.5 Non-repudiation requirements

There are two main potential non-repudiation requirements:

- Non-repudiation by the MN of data sent to the access network, and
- Non-repudiation by the access network of data sent to the MN.

The necessity of these potential requirements depends very much on details of the network access scenario. For example, if the protocol involves secure agreement of a QoS level or a level of payment for service provision, then non-repudiation by one or both parties may well be required. On the other hand, if the protocol is primarily to establish a shared secret key, then non-repudiation of the protocol itself achieves little, since all data subsequently protected using the shared secret will be repudiable.

2.1.1.6 Summary of identified requirements

From the analysis above we can say that the following are definite requirements for the network access scenario.

- authentication of the MN to the access network,
- agreement between the MN and the access network on one or more shared secret keys,
- implicit key authentication from the MN to the access network for the shared secret key(s),
- implicit key authentication from the access network to the MN for the shared secret key(s),
- freshness guarantees to the MN for the shared secret key(s),
- freshness guarantees to the access network for the shared secret key(s),
- Confidentiality of the identity of the MN on the radio interface, and
- Prevention of linking between pairs of the authentication exchanges involving the same MN.

2.1.2 Alternative access case

The alternative access case is described in section 4.6 of Annex 1. There, the user cannot exploit a relationship with a home network in order to gain access to an access network. The user may not have a home network, or the home network may not be accessible. Instead, the user pays immediately for the services received by the access network. This ad hoc payment is based on electronic and mobile payment systems respectively.

Because there is no long-term relationship with a home network that can be exploited, network access procedures have to be based on public key techniques. This section reviews the requirements for public key based network access for the alternative access case. The requirements differ from the subscription case due to architectural differences: the mobile user may remain anonymous in this scenario and the validation of payment related credentials is postponed to the accounting phase. Thus, client authentication is not mandatory.

2.1.2.1 Entity authentication requirements

Unlike in the subscription scenario, authentication of the access network towards the MN is required in the alternative access scenario. On the other hand, the MN may not need to authenticate towards the access network. Instead, the MN may perform

- preliminary authentication – instead of authenticating the identity of the user, the affiliation of the user to a certain payment scheme is verified (e.g. the customer is a GeldKarte customer). In this way, it can be guaranteed to some extent at the time of initial access that the customer is able to carry out specific electronic payment transactions of a payment scheme that is supported by the access network. This assures the access network that the access procedures for the MN may not be in vain.
- no authentication – in this case, the user of the MN remains anonymous and authentication is delayed to the accounting phase (as part of the payment transaction). The justification for this scenario is that the identity of the user is irrelevant as long as he/she pays for the transaction cost. However, in this case the foreign access network may perform some access procedures for MNs in vain, which can lead to DoS attacks in the access network.

Whether or not the client is properly authenticated, is part of the payment protocols during the accounting phase and will not be discussed here. Hence, these authentication requirements may not be fully compliant with general AAA principles where client authentication is mandatory. Here, the client (or more precisely, his payment credentials) is authenticated during the accounting phase.

Authentication of the access network towards the MN is, however, usually required since it

- provides protection against false networks (active attacks) or not preferred networks which also offer their service.
- identifies the merchant for subsequent payment communications.

It should be noted that the AR (that is, the AAA client) needs to assign a session ID to identify a particular session. The session ID is linked to an IP address but not necessarily to a user identity.

Since MN and access network may not have a prior relationship, secret key mechanisms cannot be used to authenticate the network. Hence, public key methods shall be applied. The access network sends a signed message along with its certificate and the MN checks the validity with the certificate chain it has installed. The access network should send a certificate which is signed by a CA contained in the MN's certificate chain. The MN also needs to check (possibly online) a Certificate Revocation List (CRL) to identify revoked certificates.

2.1.2.2 Key establishment requirements

The key establishment requirements do not differ from the subscription case (see section 2.1.1.2).

2.1.2.3 Payment/QoS requirements

Although payment requirements include the "secure agreement of the means of payment for service" and "secure agreement on a rate of payment for service provision" (see section 2.1.1.3), we assume that these requirements are tackled as part of the alternative access protocol (see section 4.6 of Annex 1).

2.1.2.4 Privacy requirements

In the alternative access scenario, the MN may perform preliminary authentication or remain anonymous, at least before the start of the payment communication. In the first case only general information regarding a specific payment scheme is revealed (e.g. this customer can pay by GeldKarte). In the latter case, no information about the MN's identity is sent. Hence, "confidentiality of the identity of the MN on the radio interface" is not a requirement (see section 2.1.1.4). However, "prevention of linking between pairs of the authentication exchanges involving the same MN" is still a valid requirement because although the MN's identity is not sent to the access network, MN related data is sent.

2.1.2.5 Non-repudiation requirements

Similar to section 2.1.1.5, we assume that non-repudiation is not a requirement here. However, non-repudiation will certainly be a requirement for the payment protocol (see section 4.6 of Annex 1).

2.1.2.6 Summary of identified requirements

The following list summarises the requirements for the alternative access case.

- [preliminary authentication of the MN to the access network],
- authentication of the access network to the MN,
- agreement between the MN and the access network on one or more shared secret keys,
- implicit key authentication from the MN to the access network for the shared secret key(s),
- implicit key authentication from the access network to the MN for the shared secret key(s),
- freshness guarantees to the MN for the shared secret key(s),
- freshness guarantees to the access network for the shared secret key(s), and
- Prevention of linking between pairs of the authentication exchanges involving the same MN.

2.2 JFK

User identity confidentiality is one of the design goals of the JFK protocol. There are several protocol variants: First, the protection can cover the initiator, or the responder or both. Second, the protection can be valid either against active attackers or alternatively only against passive eavesdroppers. According to [2], there are two variants of the protocol. The variants, denoted JFKi and JFKr, are very similar in many respects, with two main differences: JFKi provides active identity protection for the initiator and no identity protection for the responder, whereas JFKr provides active identity protection for the responder and passive identity protection for the initiator. In addition, JFKi contains an additional (amortisable) signature.

The notation used in the description of JFK is presented in Table 2. Please note that the following protocol description corresponds to the description in [2].

$E\{K\}(M)$	Encryption of M with secret key K.
$HMAC\{K\}(M)$	Keyed hash of M using key K in an HMAC scheme.
$SIG\{x\}(M)$	Digital signature of M using the private key belonging to principal x (Initiator or Responder). It is not assumed to be a signature with message recovery (but it can be).
G^x	Diffie-Hellman exponentials, also identifying the group-ID.
g^i, g^r	Initiator and Responder exponentials.
N_i	Initiator nonce, a random bit-string. The Initiator must pick a fresh nonce at each invocation of the JFK protocol.
IP_i	The Initiator's network identifier (IPv4 or IPv6 address). It is used by the Responder to counter a “cookie-jar” attack[1], when verifying the authenticator upon receipt of message 3.
N_r	Responder nonce, a random bit-string. The Responder must pick a fresh nonce at each invocation of the JFK protocol. The nonces are used in the session key computation, to provide key independence when one or both parties reuse the same Diffie-Hellman exponential; the session key will be different between independent runs of the protocol, as long as one of the nonces or exponentials changes.
sa	Defines the cryptographic and other properties of the Security Association (SA) the Initiator wants to establish. It contains a Domain-of-Interpretation, which JFK understands, and an application-specific bit-string.
sa'	Any information that the Responder needs to provide to the Initiator with respect to the application SA (e.g., the Responder's SPI, in IPsec).
HK_r	A transient hash key private to the Responder; this is a global parameter for the Responder (i.e., it is not different for every different protocol run), which changes periodically: the Responder must pick a new g^r every time HK_r changes.
K_{ir}	A shared key derived from g^{ir}, N_i, and N_r, used as part of the application SA (e.g., IPsec SA).
K_e	A shared key derived from g^{ir}, N_i, and N_r, used to protect messages 3 and 4 of the protocol. Although the input parameters are the same with K_{ir}, a different key derivation mechanism is used to ensure key independence.
ID_i, ID_r	Initiator and Responder certificates or public-key identifying information. Multiple such payloads may appear in a message, to indicate multiple certificates, CRLs etc. For simplicity and clarity, the notation in this draft shows only one such payload per message.
$ID_{r'}$	An indication by the Initiator to the Responder as to what identity (and corresponding key material) the latter should use to authenticate to the former. The Responder may ignore this hint. This field may contain the certificate of a CA

[1] An attacker accumulates lots of cookies from lots of IP addresses over time and then replays them all at once to overwhelm the Responder.

	trusted by the Initiator (which means that the Initiator is requesting that the Responder authenticate with a certificate chain "rooted" at that CA), or the certificate of the Responder (effectively identifying the public, and corresponding private, key of the Responder).
GRPINFO$_r$	A list of all Diffie-Hellman groups supported by the Responder, the symmetric algorithm used to protect messages 3 and 4, and the hash function used for key generation.

Table 2: JFK Notation

2.2.1 JFKi

The following message flow between Initiator and Responder describes the JFKi protocol.

Message 1, I->R:	$N_i, g^i, ID_{r'}$
Message 2, R->I:	$N_i, N_r, g^r, GRPINFO_r, ID_r,$
	$SIG\{r\}(g^r, GRPINFO_r),$
	$HMAC\{HKr\}(g^r, N_r, g^i, N_i, IP_i)$
Message 3, I->R:	$N_i, N_r, g^i, g^r, HMAC\{HKr\}(g^r, N_r, g^i, N_i, IP_i),$
	$E\{K_e\}(ID_i, sa, SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa))$
Message 4, R->I:	$E\{K_e\}(SIG\{r\}(N_i, N_r, g^i, g^r, ID_i, sa, sa'), sa')$

The key used to protect Messages (3) and (4), K_e, is computed as $HMAC\{g^{ir}\}(N_i, N_r, 1)$. The session key used by IPsec (or any other application), K_{ir}, is $HMAC\{g^{ir}\}(N_i, N_r, 0)$.

In Message (1), it is assumed that the initiator already knows a group and generator that is acceptable to the Responder. This message also contains an indication as to which ID the Initiator would like the Responder to use to authenticate. In contrast to JFKr, $ID_{r'}$ is sent in the clear; however, notice that the responder's ID in Message (2) is also sent in the clear, so there is no loss of privacy in this respect.

Message (2) is more complex. Assuming that the Responder accepts the Diffie-Hellman group in the Initiator's message, he replies with a signed copy of his own exponential (in the same group), information on what secret key algorithms are acceptable for the next message, a random nonce, his identity (certificates or a bit-string identifying his public key), and an authenticator calculated from a secret, HK_r, known to the Responder; the authenticator is computed over the two exponentials and nonces, and the Initiator's network address. The authenticator key is changed at least as often as g^r, thus preventing replays of stale data. Finally, note that the Responder does not need to generate any state at this point, and the only “expensive” operation is a MAC calculation.

Message (3) echoes back the data sent by the Responder, including the authenticator. The authenticator is used by the Responder to verify the authenticity of the returned data. The authenticator also confirms that the sender of Message (3) uses the same address as in Message (1) – this can be used to detect and counter a “cookie jar” DDoS attack. The message also includes the Initiator's identity and service request, and a signature computed over the nonces, the Responder's identity, and the two exponentials. This latter information is all encrypted under a key derived from the Diffie-Hellman computation and the nonces N_i and N_r. The encryption and authentication use algorithms specified in $GRPINFO_r$. The Responder keeps a copy of recently-received Message (3)'s, and their corresponding Message (4). Receiving a duplicate (or replayed) Message (3) causes the Responder to simply retransmit the corresponding Message (4), without creating new state or invoking IPsec. This cache of messages can be reset as soon as g^r or HK_r are changed. The Responder's exponential (g^r) is re-sent by the Initiator because the Responder may be generating a new g^r for every new JFK protocol run (e.g., if the arrival rate of requests is below some threshold).

Message (4) contains application-specific information (such as the Responder's IPsec SPI), and a signature on both nonces, both exponentials, and the Initiator's identity. Everything is encrypted by K_e, which is derived from N_i, N_r, and g^{ir} (the result of the Diffie-Hellman computation).

2.2.2 JFKr

The second variant of the JFK protocol, JFKr, provides the same DoS protection and identity protection against passive attackers for both the Initiator and the Responder, but no protection against active identity discovery attacks for the Initiator (the Responder is protected against active identity discovery).

Message 1, I->R:	N_i, g^i
Message 2, R->I:	$N_i, N_r, g^r, GRPINFO_r,$
	$HMAC\{HK_r\}(g^r, N_r, N_i, IP_i)$
Message 3, I->R:	$N_i, N_r, g^i, g^r, HMAC\{HK_r\}(g^r, N_r, N_i, IP_i),$
	$E\{K_e\}(ID_i, ID_{r'}, sa, SIG\{i\}(N_i, N_r, g^i, g^r, GRPINFO_r)),$
	$HMAC\{K_a\}('I', E\{K_e\}(ID_i, ID_{r'}, sa, SIG\{i\}(N_i, N_r, g^i, g^r, GRPINFO_r)))$
Message 4, R->I:	$E\{K_e\}(ID_r, sa', SIG\{r\}(g^r, N_r, g^i, N_i)),$
	$HMAC\{K_a\}('R', E\{K_e\}(ID_r, sa', SIG\{r\}(g^r, N_r, g^i, N_i)))$

The keys used to protect Messages (3) and (4), K_e and K_a, are computed as $HMAC\{g^{ir}\}(N_i, N_r, 1)$ and $HMAC\{g^{ir}\}(N_i, N_r, 2)$ respectively. The session key used by IPsec (or any other application), K_{ir}, is $HMAC\{g^{ir}\}(N_i, N_r, 0)$.

In Message (1), it is assumed that the initiator already knows a group and generator that is acceptable to the Responder.

In Message (2), the Responder replies with his own exponential (in the same group), information on what secret key algorithms are acceptable for the next message, a random nonce, and an authenticator calculated from a secret, HK_r, known to the Responder; the authenticator is computed over the Responder exponential, the two nonces, and the Initiator's network address. Please note that the Responder does not have to generate any state at this point.

Message (3) echoes back the data sent by the Responder, including the authenticator. The authenticator is used by the Responder to verify the authenticity of the returned data. The authenticator also confirms that the sender of Message (3) uses the same address as in Message (1). The message also includes the Initiator's identity and service request, and a signature computed over the nonces, the Responder's identity, and the two exponentials. This latter information is all encrypted under a key derived from the Diffie-Hellman computation and the nonces N_i and N_r. The encryption and authentication use algorithms specified in $GRPINFO_r$. The Responder keeps a copy of recently-received Message (3)'s, and their corresponding Message (4). Receiving a duplicate (or replayed) Message (3) causes the Responder to simply retransmit the corresponding Message (4), without creating new state or invoking IPsec (or other application using JFKr as the key establishment protocol). This cache of messages can be reset as soon as HK_r is changed. The Responder's exponential (g^r) is re-sent by the Initiator because the Responder may be generating a new g^r for every new JFK protocol run (e.g., if the arrival rate of requests is below some threshold). The message is also protected by an MAC (such as HMAC), using a key derived from the Diffie-Hellman computation and the nonces N_i and N_r. Notice that this key is different form the one used for encrypting the message.

Message (4) contains application-specific information (such as the Responder's IPsec SPI), and a signature on both nonces and both exponentials. Everything is encrypted by K_e, which is derived from N_i, N_r, and g^{ir} (the result of the Diffie-Hellman computation). As in Message (3), this message is also protected by a MAC using key K_a.

In JFKr, the parties obtain a shared encryption key, K_e, before any of the parties send their identities. Therefore, both parties send their identities encrypted with K_e, thus providing both parties with identity protection against passive eavesdroppers. In addition, the party that first reveals its identity is the initiator. This way, the responder is required to reveal its identity only after it verifies the identity of the initiator. This guarantees active identity protection to the responder.

2.3 IKEv2

IKE performs mutual authentication and establishes an IKE security association that can be used to efficiently establish SAs for ESP, AH and/or IPcomp. This version, namely IKEv2 [12], greatly simplifies IKE by replacing the 8 possible phase 1 exchanges with a single exchange based on either public signature keys or shared secret keys. The single exchange provides identity hiding, yet works in 2 round trips (all the identity hiding exchanges in IKE v1 required 3 round trips). Latency of setup of an IPsec SA is further reduced from IKEv1 by allowing setup of an SA for ESP, AH, and/or IPcomp to be piggybacked on the initial IKE exchange. It also improves security by allowing the Responder to be stateless until it can be assured that the Initiator can receive at the claimed IP source address.

Please note that the following protocol description corresponds to the description in [12].

The setup of the IKE-SA is called “phase 1” and subsequent IKE exchanges “phase 2” even though setup of a child-SA can be piggybacked on the initial phase 1 exchange. The phase 1 exchange is two request/response pairs. A phase 2 exchange is one request/response pair, and can be used to create or delete a child-SA, rekey or delete the IKE-SA, or give information such as error conditions.

The IKE message flow always consists of a request followed by a response. It is the responsibility of the requester to ensure reliability. If the response is not received within a timeout interval, the requester retransmits the request.

The first request/response of a phase 1 exchange, which is called IKE_SA_init, negotiates security parameters for the IKE-SA, and sends Diffie-Hellman values. The response is called IKE_SA_init_response.

The second request/response, which is called IKE_auth, transmits identities, proves knowledge of the private signature key, and sets up an SA for the first (and often only) AH and/or ESP and/or IPcomp. This is called the response IKE_auth_response.

If the Responder feels it is under attack, and wishes to use a stateless cookie, it can respond to an IKE_SA_init with an IKE_SA_init_reject with a cookie value that must be sent with a subsequent IKE_SA_init_request. The Initiator then sends another IKE_SA_init, but this time including the Responder's cookie value[2].

Phase 2 exchanges each consist of a single request/response pair. The types of exchanges are CREATE_CHILD_SA (creates a child-SA), or an informational exchange which deletes a child-SA or the IKE-SA or informs the other side of some error condition. All these messages require a response, so an informational message with no payloads can serve as a check for liveness.

The phase 1 exchange

The base phase 1 exchange is a four message exchange (two request/response pairs). The first pair of messages, the IKE_SA_init exchange, negotiate cryptographic algorithms, (optionally) indicate trusted CA names, exchange nonces, and do a Diffie-Hellman exchange. This pair might be repeated if the response indicates that none of the cryptographic proposals are acceptable, or the Diffie-Hellman group chosen by the Initiator for sending her Diffie-Hellman value is not the group that the Responder would have chosen, or if the Responder is under attack and will only answer IKE_SA_init requests containing a valid returned cookie value.

[2] In IKEv2, the term cookie has two possible meanings. First, cookies are used as IKE-SA identifiers in the headers of IKE messages. Second, cookies are used for a limited protection from denial of service attacks.

The second pair of messages, the IKE_auth and the IKE_auth_response, authenticate the previous messages, exchange identities and certificates, and establish the first child_SA. This pair of messages is encrypted with a key established through the IKE_SA_init exchange, so the identities are hidden from eavesdroppers.

In the following description, the payloads contained in the message are indicated by names such as *SA*. The details of the contents of each payload are described later. Payloads which may optionally appear will be shown in brackets, such as [*CERTREQ*], would indicate that optionally a certificate request payload can be included.

	Initiator		Responder
(1) IKE_SA_init_request	*HDR, SA_{i1}, KE_i, N_i*	→	
(2) IKE_SA_init_response		←	*HDR, SA_{r1}, KE_r, N_r,* [*CERTREQ*]
(3) SA_auth_request	*HDR*, ID_i,* [*CERT,*] [*CERTREQ,*] [*ID_r,*] *AUTH, SA_{i2}, TS_i, TS_r*	→	
(4) SA_auth_response		←	*HDR*, ID_r,* [*CERT,*] *AUTH, SA_{r2}, TS_i, TS_r*

Table 3: IKEv2 phase 1 exchange

In message (1) of Table 3, the SA_{i1} payload states the cryptographic algorithms the Initiator supports for the IKE SA. The *KE* payload sends the Initiator's Diffie-Hellman value. N_i is the Initiator's nonce.

The Responder chooses among the Initiator's cryptographic algorithms and expresses that choice in the SA_{r1} payload of message (2), completes the Diffie-Hellman exchange with the KE_r payload, and sends its nonce in the N_r payload.

At this point in time each party generates *SKEYSEED* and its derivatives. The following two messages, the SA_auth and SA_auth_response, are encrypted and integrity protected (as indicated by the '*' following the IKE header) and the encryption bit in the IKE header is set. The keys used for the encryption and integrity protection are derived from SK_a and SK_e as described below.

In message (3), the Initiator identifies herself with the ID_i payload and authenticates herself to the Responder with the *AUTH* payload, and begins negotiation of a child-SA using the SA_{i2} payload. The fields starting with SA_{i2} are described in the description of Phase 2. There are optional fields where the Initiator can provide certificates [*CERT*] the Responder might find useful in validating *AUTH*, her list of preferred root certifiers [*CERTREQ*], and the name of the entity with which she is trying to open a connection [ID_r] (for the case where multiple named entities exist at a single IP address).

Finally in message (4), the Responder identifies himself with an *ID* payload optionally sends one or more certificates, authenticates himself with the *AUTH* payload, and completes negotiation of a child-*SA* with the additional fields described below in the phase 2 exchange.

Generating keying material for the IKE-SA

The shared secret information is computed as follows. A quantity called *SKEYSEED* is calculated from the nonces exchanged during the IKE_SA_init exchange, and the Diffie-Hellman shared secret established during that exchange. *SKEYSEED* is used to calculate three other secrets: SK_d used for deriving new keys for the child-SAs established with this IKE-SA; SK_a used for authenticating the component messages of subsequent exchanges; and SK_e used for encrypting (and of course decrypting) all subsequent exchanges. *SKEYSEED* and its derivatives are computed as follows:

$$SKEYSEED = prf\,(N_i \mid N_r,\ g^{ir})$$

$$SK_d = prf\,(SKEYSEED,\ g^{ir} \mid N_i \mid N_r \mid CKY\text{-}I \mid CKY\text{-}R \mid 0)$$

$SK_a = prf\,(SKEYSEED,\ SK_d \mid g^{ir} \mid N_i \mid N_r \mid CKY\text{-}I \mid CKY\text{-}R \mid 1)$

$SK_e = prf\,(SKEYSEED,\ SK_a \mid g^{ir} \mid N_i \mid N_r \mid CKY\text{-}I \mid CKY\text{-}R \mid 2)$

CKY-I and *CKY-R* are the Initiator's and Responder's cookies, respectively, from the IKE header. g^{ir} is the shared secret from the ephemeral Diffie-Hellman exchange. N_i and N_r are the nonces, stripped of any headers. 0, 1, and 2 are represented by a single octet containing the value 0, 1, or 2 (the values, not the ASCII representation of the digits). *prf* is the "pseudo-random" cryptographic function negotiated in the IKE-SA-init exchange.

The two directions of flow use different keys. Keys used to protect messages from the original initiator are taken from the first bits of SK_a and SK_e. Keys used to protect messages in the other direction are taken from subsequent bits. Each algorithm takes a fixed number of bits of keying material, which is specified as part of the algorithm. If the total number of key bits needed is greater than the size of the output of the *prf* function, the keying material must be expanded.

For situations where the amount of keying material desired is greater than that supplied by the *prf*, *KEYMAT* is expanded by feeding the results of the *prf* back into itself and concatenating results until the required keying material has been reached. In other words,

$KEYMAT = K_1 \mid K_2 \mid K_3 \mid \ldots$

where:

$K_1 = prf\,(SK_x,\ 0)$

$K_2 = prf\,(SK_x,\ K_1)$

$K_3 = prf\,(SK_x,\ K_2)$ etc.

where 0 is represented by a single octet containing the value 0 (the value, not the ASCII representation of the digit), and SK_x is either SK_e or SK_a depending on which keying material needs expansion.

Authentication of the IKE-SA

The peers are authenticated by having each sign (or MAC using a shared secret as the key) the concatenation of their own first message and the other peer's nonce. The octets to be signed start with the first octet of the header and end with the last octet of the last payload. The octets of the nonce are only the content and not the header.

Note that all payloads of the peer's own first message are included under the signature, including payload types not defined in this document. It is possible that some other payloads defined in the future might appropriately be zeroed before signing, but such a possibility is not supported by this version of IKE. Optionally, messages 3 and 4 may include a certificate, or certificate chain providing evidence that the key used to compute a digital signature belongs to the name in the *ID* payload. The signature or MAC will be computed using algorithms dictated by the type of key used by the signer, an RSA-signed PKCS1-padded-SHA1-hash for an RSA digital signature, a DSS-signed SHA1-hash for a DSA digital signature, or the negotiated PRF function for a pre-shared key. There is no requirement that the Initiator and Responder sign with the same cryptographic algorithms. The choice of cryptographic algorithms depends on the type of key each has. This type is either indicated in the certificate supplied or, if the keys were exchanged out of band, the key types must have been similarly learned. It will commonly be the case, but it is not required that if a shared secret is used for authentication that the same key is used in both directions. In particular, the initiator may be using a shared key derived from a password while the responder may have a public signature key and certificate.

The CREATE-CHILD-SA (phase 2) exchange

A phase 2 exchange is one request/response pair, and can be used to create or delete a child-SA, delete or rekey the IKE-SA, check the liveness of the IKE-SA, or deliver information such as error conditions. It is encrypted and integrity protected using the keys negotiated during the creation of the IKE-SA.

Messages are cryptographically protected using the cryptographic algorithms and keys negotiated in the first two messages of the IKE exchange. Encryption uses keys derived from SK_e, one in each direction; Integrity uses keys derived from SK_a, one in each direction.

Either endpoint may initiate a phase 2 exchange, so in this section the term Initiator refers to the endpoint initiating this exchange. When relevant, the Initiator of the IKE SA will be referred to as such.

A child-SA is created by sending a CREATE_CHILD_SA request. If PFS for the child-SA is desired, the CREATE_CHILD_SA request contains *KE* payloads for an additional Diffie-Hellman exchange. The keying material for the child-SA is a function of SK_d established during the establishment of the IKE-SA, the nonces exchanged during the CREATE_CHILD_SA exchange, and the Diffie-Hellman value (if *KE* payloads are included in the CREATE_CHILD_SA exchange).

In the child-SA created as part of the phase 1 exchange, a second *KE* payload must not be used, and the nonces are not transmitted but are assumed to be the same as the phase 1 nonces.

	Initiator		Responder
(1)	$HDR^*, SA, N_i, [KE_i], TS_i, TS_r$	→	
(2)		←	$HDR^*, SA, N_r, [KE_r], TS_i, TS_r$

Table 4: CREATE_CHILD_SA

In the CREATE_CHILD_SA request (message (1) in Table 4), the Initiator sends SA offer(s) in the *SA* payload(s), a nonce in the N_i payload, optionally a Diffie-Hellman value in the *KE* payload, and the proposed traffic selectors in the TS_i and TS_r payloads.

The message past the header is encrypted and the message including the header is integrity protected using the cryptographic algorithms negotiated in phase 1.

In the CREATE_CHILD_SA response (message (2) in Table 4), the Responder replies (using the same Message *ID* to respond) with the accepted offer in an *SA* payload, a Diffie-Hellman value in the *KE* payload if and only if the Initiator included one, and the traffic selectors for traffic to be sent on that *SA* in the *TS* payloads, which may be a subset of what the Initiator of the child-SA proposed.

Generating keying material for IPsec SAs

Child-SAs are created either by being piggybacked on the phase 1 exchange, or in a phase 2 CREATE_CHILD_SA exchange. Keying material for them is generated as follows:

$$KEYMAT = prf\,(SK_d,\ protocol \mid SPI \mid N_{in} \mid N_{out})$$

For phase 2 exchanges with PFS the keying material is defined as:

$$KEYMAT = prf\,(SK_d,\ g(p2)^{ir} \mid protocol \mid SPI \mid N_{in} \mid N_{out})$$

where $g(p2)^{ir}$ is the shared secret from the ephemeral Diffie-Hellman exchange of this phase 2 exchange. In either case, *"protocol",* and *"SPI"*, are from the SA payload that contained the negotiated (and accepted) proposal, N_{in} is the body of the sender's (inbound using this SPI) nonce payload minus the generic header, and N_{out} is the body of the destination's (outbound using this SPI) nonce payload minus the generic header.

A single child-SA negotiation results in two security associations – one inbound and one outbound. Different nonces and SPIs for each SA (one chosen by the Initiator, the other by the Responder) guarantee a different key for each direction. The SPI chosen by the destination of the SA and the nonces (ordered source followed by destination) are used to derive *KEYMAT* for that SA.

This keying material (whether with PFS or without) must be used with the negotiated SA. In the case of an ESP SA needing two keys for encryption and authentication, the encryption key is taken from the first octets of *KEYMAT* and the authentication key is taken from the next octets. Each cryptographic algorithm takes a fixed number of bits of keying material specified as part of the algorithm.

For situations where the amount of keying material desired is greater than that supplied by the *prf*, *KEYMAT* is expanded by feeding the results of the *prf* back into itself and concatenating results until the required keying material has been reached. In other words,

$$KEYMAT = K_1 \mid K_2 \mid K_3 \mid \ldots$$

where:

$$K_1 = prf\,(SK_d,\ [\ g(p2)^{ir} \mid\ ]\ protocol \mid SPI \mid N_{in} \mid N_{out})$$

$$K_2 = prf\,(SK_d,\ K_1 \mid [\ g(p2)^{ir} \mid\ ]\ protocol \mid SPI \mid N_{in} \mid N_{out})$$

$$K_3 = prf\,(SK_d,\ K_2 \mid [\ g(p2)^{ir} \mid\ ]\ protocol \mid SPI \mid N_{in} \mid N_{out}) \text{ etc.}$$

Generating keying material for IKE-SAs from a create-child exchange

The create-child exchange can be used to re-key an existing IKE-SA. New Initiator and Responder cookies are supplied in the SPI fields. The *ID* and *TS* payloads are omitted when rekeying an IKE-SA. *SKEYSEED* for the new IKE-SA is computed using SK_d from the existing IKE-SA as follows:

$$SKEYSEED = prf\,(SK_d\ (old),\ [g(p2)^{ir}] \mid 0 \mid CKY\text{-}I \mid CKY\text{-}R \mid N_i \mid N_r)$$

where $g(p2)^{ir}$ is the shared secret from the ephemeral Diffie-Hellman exchange of this phase 2 exchange, *CKY-I* is the 8-octet “SPI” from the SA payload in the CREATE_CHILD_SA request, CKY-R is the 8-octet "SPI" from the SA payload in the CREATE_CHILD_SA response, and N_i and N_r are the two nonces stripped of any headers. *“0”* is a single octet containing the value zero (the protocol ID of IKE).

The new IKE SA must reset its message counters to 1. SK_d, SK_a, and SK_e are computed from *SKEYSEED* as specified in “Generating key material for the IKE-SA”.

Informational (phase 2) exchange

At various points during an IKE-SA, peers may desire to convey control messages to each other regarding errors or notifications of certain events. To accomplish this IKE defines a (reliable) Informational exchange. Usually Informational exchanges happen during phase 2 and are cryptographically protected with the IKE exchange.

Control messages that pertain to an IKE-SA must be sent under that IKE-SA. Control messages that pertain to Child-SAs must be sent under the protection of the IKE-SA which generated them (or its successor if the IKE-SA keys are rolled over).

There are two cases in which there is no IKE-SA to protect the information. One is in the response to an IKE_SA_init_request to request a cookie or to refuse the SA proposal. This would be conveyed in a *Notify* payload of the IKE_SA_init_response.

The other case in which there is no IKE-SA to protect the information is when a packet is received with an unknown SPI. In that case the notification of this condition will be sent in an informational exchange that is cryptographically unprotected.

Messages in an Informational Exchange contain zero or more *Notification* or *Delete* payloads. The Recipient of an Informational Exchange request must send some response (else the Sender will assume the message was lost in the network and will retransmit it). That response can be a message with no payloads. Actually, the request message in an Informational Exchange can also contain no payloads. This is the expected way an endpoint can ask the other endpoint to verify that it is alive.

ESP, AH, and IPcomp SAs always exist in pairs, with one SA in each direction. When an SA is closed, both members of the pair must be closed. When SAs are nested, as when data is encapsulated first with IPcomp, then with ESP, and finally with AH between the same pair of endpoints, all of the SAs (up to six) must be deleted together. To delete an SA, an Informational Exchange with one or more delete payloads is sent listing the SPIs (as known to the recipient) of the SAs to be deleted. The recipient must close the designated SAs. Normally, the reply in the Informational Exchange will contain delete payloads for the paired SAs going in the other direction. There is one exception. If by chance both ends of a set of SAs independently decide to close them, each may send a delete payload and the two requests may cross in the network. If a node receives a delete request for SAs that it has already issued a delete request for, it must delete the incoming SAs while processing the request and the outgoing SAs while processing the response. In that case, the responses must not include delete payloads for the deleted SAs, since that would result in duplicate deletion and could in theory delete the wrong SA.

A node should regard half open connections as anomalous and audit their existence should they persist. Note that this specification nowhere specifies time periods, so it is up to individual endpoints to decide how long to wait. A node may refuse to accept incoming data on half open connections but must not unilaterally close them and reuse the SPIs. If connection state becomes sufficiently messed up, a node may close the IKE-SA which will implicitly close all SAs negotiated under it. It can then rebuild the SA's it needs on a clean base under a new IKE-SA.

	Initiator		Responder
(1)	*HDR*, N, ..., D, ...*	→	
(2)		←	*HDR*, N, ..., D, ...*

Table 5: Informational Exchange

The processing of an Informational Exchange (see Table 5) is determined by its component payloads.

2.4 Analysis

2.4.1 Subscription case

This section assesses the protocols introduced in the previous section regarding the requirements formulated for the subscription case (see section 2.1.1).

2.4.1.1 Assessment of JFK

In the following, we assume that the Initiator *I* represents the MN and the Responder *R* represents the access network.

JFKi

In the following, we list the requirements identified in section 2.1.1.6 and indicate whether they are satisfied by the JFKi protocol. We also give the cryptographic structure that fulfils the specific requirement.

Authentication MN to access network – Yes, signature in message (3) ($SIG\{i\}(N_i, N_r, g^i, g^r, ID_r, sa)$ – parameters for Diffie Hellman key agreement signed by *I*).

Key agreement – Yes, Diffie-Hellman key agreement based on the values of N_i and N_r (nonces) and g^i and g^r (Diffie Hellman keys) respectively.

Implicit key authentication MN to access network – Yes, the certificate of the Initiator relating to g^i (namely ID_i) is sent in message (3). With this, the Responder can verify that the Diffie Hellman key g^i does indeed belong to the Initiator.

Implicit key authentication access network to MN – Yes, the certificate of the Responder relating to g^r (namely ID_r) is sent in message (2). With this, the Initiator can verify that the Diffie Hellman key g^r does indeed belong to the Responder.

Freshness guarantees to MN – Yes, the nonce N_i is used for the Diffie Hellman key agreement.

Freshness guarantees to access network – Yes, the nonce N_r is used for the Diffie Hellman key agreement.

Confidentiality of MN identity – Yes, the MN's identity is protected against active and passive attacks. Active identity protection is guaranteed since the Responder, that is the access network, authenticates before the MN. Hence, the MN can be assured about the trustworthiness of the access network before it authenticates itself. Moreover, passive identity protection is given because the Initiator's identity is encrypted in message (3).

Prevention of linking between pairs of authentication exchanges involving the same MN – The fulfilment of this requirement depends on the nature of the Diffie Hellman (public) key g^i which is linked to the MN's identity. In JFK, Diffie Hellman exponentials can be recomputed arbitrarily, depending on implementation policies. Reusing Diffie Hellman exponentials can be very useful in order reduce the workload. However, in our scenario we require the MN's Diffie Hellman exponential to be recomputed after each JFK protocol run. Otherwise, pairs of authentication exchanges involving the same MN can be linked.

In addition, the JFK_i protocol fulfils a number of requirements that are not relevant in our network access scenario, namely

Authentication access network to MN – Yes, signature in message (2) ($SIG\{r\}(g^r, GRPINFO_r)$).

Explicit key authentication MN to access network – Yes, signature in message (3).

Explicit key authentication access network to MN – Yes, signature in message (4).

Confidentiality of responder identity – No, ID_r sent in the clear.

Further remarks:

- The Responder is stateless until the receipt of message (3). This prevents a DoS attack.
- The protocol requires 2 round trips.

JFKi fulfils all requirements formulated in section 2.1.1.6. Hence, we can recommend JFKi for public key based network access in the subscription case.

<u>JFKr</u>

In the following, we assess the JFKr protocol. The assessment yields a similar result to the assessment of JFKi.

Authentication MN to access network – Yes, signature in message (3) ($SIG\{i\}(N_i, N_r, g^i, g^r, GRPINFO_r)$).

Key agreement – Yes, Diffie-Hellman key agreement based on the values of N_i and N_r (nonces) and g^i and g^r (Diffie Hellman keys) respectively.

Implicit key authentication MN to access network – Yes, the certificate of the Initiator relating to g^i (namely ID_i) is sent in message (3). With this, the Responder can verify that the Diffie Hellman key g^i does indeed belong to the Initiator.

Implicit key authentication access network to MN – Yes, the certificate of the Responder relating to g^r (namely ID_r) is sent in message (4). With this the Initiator can verify that the Diffie Hellman key g^r does indeed belong to the Responder.

Freshness guarantees to MN – Yes, the nonce N_i is used for the Diffie Hellman key agreement.

Freshness guarantees to access network – Yes, the nonce N_r is used for the Diffie Hellman key agreement.

Confidentiality of MN identity – No, the MN's identity is protected against passive, but not against active attacks. Passive identity protection is guaranteed because the Initiator's identity is encrypted in message (3). However, because the access network only authenticates in message (4), the MN's authentication in message (3) is performed towards an at that point unknown access network. The access network could be a bogus access network.

Prevention of linking between pairs of authentication exchanges involving the same MN – Since Diffie Hellman exponentials can be recomputed arbitrarily, the requirement is fulfilled under the assumption that the MN's Diffie Hellman exponential is recomputed after each protocol run.

The MN's identity is not protected against active attacks. Since active identity protection is an important requirement in the network access scenario, we cannot recommend JFKr as a public key authentication protocol for network access of a MN to an access network.

2.4.1.2 Assessment of IKEv2

In the following, we assume that the Initiator *I* represents the MN and the Responder *R* represents the access network.

Authentication MN to access network – Yes, ID_i (Initiator identity) and *AUTH* (Initiator authentication) sent in message (3).

Key agreement – Yes, Diffie-Hellman key agreement using KE_i (Diffie Hellman key of Initiator) and N_i (nonce of Initiator) in message (1), and KE_r (Diffie Hellman key of Responder) and N_r (nonce of Responder) in message (2).

Implicit key authentication MN to access network – Yes, the certificate of the Initiator relating to KE_i (namely *CERT*) is sent in message (3).

Implicit key authentication access network to MN – Yes, the certificate of the Responder relating to KE_r (namely *CERT*) is sent in message (4).

Freshness guarantees to MN – Yes, the nonce N_i is used for the Diffie Hellman key agreement.

Freshness guarantees to access network – Yes, the nonce *Nr* is used for the Diffie Hellman key agreement.

Confidentiality of MN identity – No, the MN's identity is protected against passive, but not against active attacks. Passive identity protection is guaranteed because the Initiator's identity (ID_i) is encrypted in message (3). However, because the access network only authenticates in message (4), the MN's authentication in message (3) is performed towards an at that point unknown access network. The access network could be a bogus access network.

Prevention of linking between pairs of authentication exchanges involving the same MN – The fulfilment of this requirement depends on the nature of the Diffie Hellman (public) key KE_i which is linked to the MN's identity. KE_i is sent in the clear in message (1). Since Diffie Hellman exponentials can be recomputed arbitrarily, the requirement is fulfilled under the assumption that the MN's Diffie Hellman exponential is recomputed after each protocol run.

The following list states a number of other cryptographic requirements that are fulfilled by IKEv2, that are, however, not essential in the network access scenario:

Explicit key authentication MN to access network – Yes, *AUTH* (Initiator authentication) sent in message (3).

Explicit key authentication access network to MN – Yes, *AUTH* (Responder authentication) sent in message (4).

Confidentiality of access network identity –Yes, the identity of the access network (ID_r) is encrypted in message (4).

Authentication access network to MN – Yes, ID_r (Responder identity) and *AUTH* (Responder authentication) sent in message (4).

Further remarks:

- The protocol uses a variable-round-trip handshake, with 4 messages under normal circumstances and 6 under attack. The extra two messages are a simple cookie exchange designed to force the attacker to prove that he has a round-trip to the Responder.
- AUTH is the authentication message and can be based on secret (MAC) or public (signature) keys. However, here we are looking at public keys, because the MN and access network have not met beforehand and hence do not share a secret.

In IKEv2, the MN's identity is not protected against active attacks. Since active identity protection is an important requirement in the network access scenario, we cannot recommend IKEv2 as a public key authentication protocol for network access of a MN to an access network.

2.4.2 Alternative access case

2.4.2.1 Assessment of JFK

The assessment of JFK is similar to the assessment for the subscription case. Here, we will briefly summarise the main differences, considering joint results for JFKi and JFKr. The main differences in terms of the requirements are that in the alternative access case, we do not need "authentication MN to access network" and "confidentiality of MN identity", but "authentication access network to MN" instead.

[Preliminary authentication MN to access network – Yes, signature in message (3)]

Authentication access network to MN – Yes, signature in message (2) ($SIG\{r\}(g^r, GRPINFO_r)$).

Key agreement – Yes, see section 2.4.1.1.

Implicit key authentication MN to access network – Yes, see section 2.4.1.1.

Implicit key authentication access network to MN – Yes, see section 2.4.1.1.

Freshness guarantees to MN – Yes, see section 2.4.1.1.

Freshness guarantees to access network – Yes, see section 2.4.1.1.

Prevention of linking between pairs of authentication exchanges involving the same MN – Yes, if the Diffie Hellman key g^i of the MN is recomputed for each protocol run.

Moreover, there are two important issues to consider:

- JFK provides mutual authentication between Initiator and Responder. In the alternative access case, we do not require Initiator authentication or we require preliminary authentication only. Therefore, application of JFK for alternative access depends also on the question whether the protocol can be employed with only unilateral server-side authentication. This could be achieved by using self-signed certificates on the client side. Whether this is possible depends on policy issues.
- In the alternative access case, payment credentials may need to be transmitted as part of the network access protocol. Whether this is possible in JFK, needs to be investigated further.

JFKi and JFKr are both suitable for network access in the alternative access case, assuming that they can be employed with unilateral server-side authentication only. Moreover, the Diffie Hellman key g^i of the MN has to be recomputed for each protocol run.

2.4.2.2 Assessment of IKEv2

The assessment of IKEv2 is similar to the subscription case (see section 2.4.1.2). The following list summarises the results:

[Preliminary authentication MN to access network – Yes, ID_i (Initiator identity) and $AUTH$ (Initiator authentication) sent in message (3).]

Authentication access network to MN – Yes, ID_r (Responder identity) and $AUTH$ (Responder authentication) sent ion message (4).

Key agreement – Yes, see section 2.4.1.2.

Implicit key authentication MN to access network – Yes, see section 2.4.1.2.

Implicit key authentication access network to MN – Yes, see section 2.4.1.2.

Freshness guarantees to MN – Yes, see section 2.4.1.2.

Freshness guarantees to access network – Yes, see section 2.4.1.2.

Prevention of linking between pairs of authentication exchanges involving the same MN – Yes, if the Diffie Hellman key KE_i of the MN is recomputed for each protocol run.

The same issues that were discussed in the previous section, namely unilateral server-side authentication or preliminary authentication and payment related payload need to be considered.

2.5 Conclusions

In this chapter, network access based on public key techniques has been discussed. First, the subscription case where the user has a long-term trust relationship with a home network was described and analysed. Second, we described the alternative access case where there is no home network available. Instead, the user pays ad hoc for the services he/she requires. In the latter case, public key techniques are used to secure the (partly) wireless link between MN and AR.

The main goal of this chapter was to identify the requirements for public key mechanisms for network access, first for the subscription and second for the alternative access case (see sections 2.1.1 and 2.1.2 respectively). Two public key authentication and key agreement protocols were then reviewed, namely

- JFK, and
- IKEv2

The protocols were evaluated with respect to the requirements identified previously. The assessment for the subscription case yielded that JFKi fulfils all requirements for network access. However, JFKr and IKEv2 do not protect the user from active attacks from bogus access networks and therefore cannot be recommended for the network access scenario.

In the alternative access case, the set of requirements differs because the MN may use preliminary authentication or even remain anonymous initially. This implies that "authentication of the MN towards the access network" and "MN confidentiality" must be considered differently to the subscription case. However, the MN requires the access network to authenticate itself towards the MN. The assessment of the public key protocols for the alternative access case yielded that JFKi, JFKr, and IKEv2 are all suitable for the network access scenario.

Future work in public key based network access involves following the work of the IETF regarding a son-of-IKE protocol and possibly highlighting the requirements for our application, namely public key based network access, to the IETF. Moreover, other public key authentication protocols could be investigated.

Also, the role of the MN in the alternative access scenario needs to be further refined. Two cases, namely preliminary authentication and no authentication, have been identified and need to be investigated further. For example, what are the consequences of no authentication on the public key authentication protocol? Is this approach acceptable to network operators considering that it may open the door for DoS attacks on the access network?

3 PK revocation in a mobile environment

In D04 [39] 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 were considered. 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 [39] 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 updates considerably reduces the effectiveness of CRL use.

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

To compare the two methods a set of criteria for the comparison are required. The criteria used here are as follows.

- 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.

3.1 OCSP

3.1.1 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.

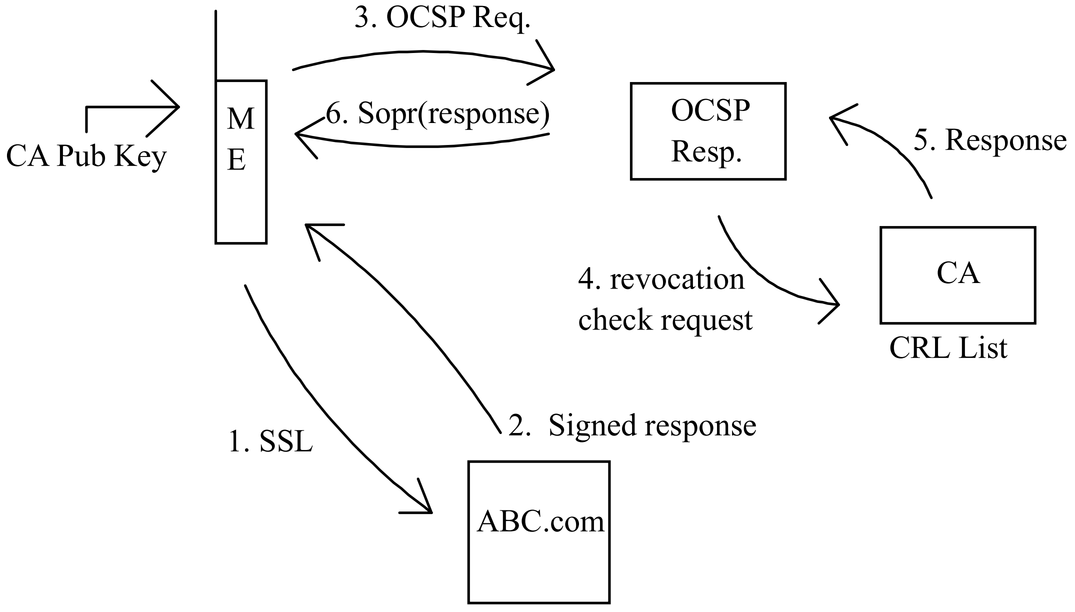

Figure 1 – OCSP using issuing CA as OCSP responder

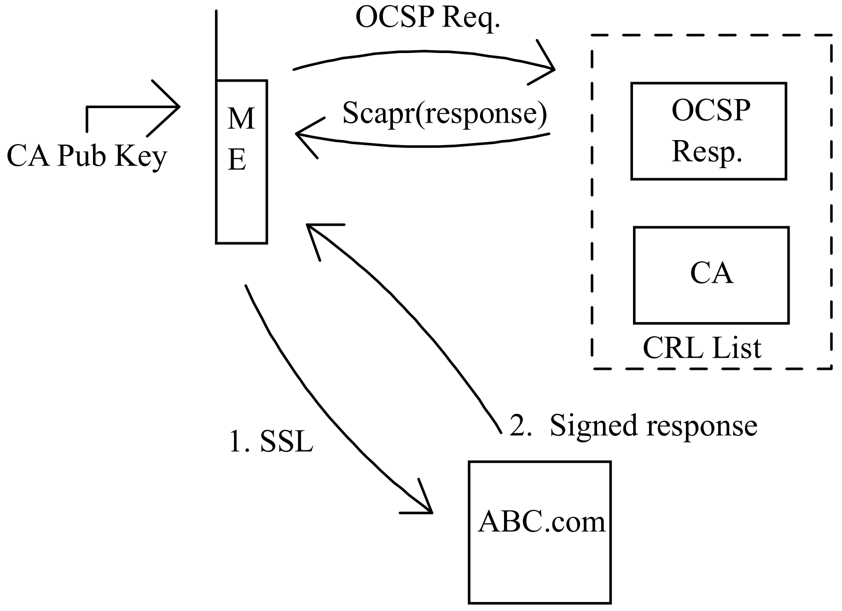

Figure 2 – OCSP using delegated OCSP responder

The first scenario, shown in Figure 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 the second scenario, shown in Figure 2, 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 [30] is that the CA issues the responder with a certificate containing an extension indicating that the responder is authorised to issue an 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). It is also necessary in this case for the client to know where to go in order to get status information for the end entity certificate. In most cases a certificate extension field known as the AuthorityInformationAccess extension is present in the end entity certificate. The AuthorityInformationAccess field indicates where the client must go to get the status information for this certificate. In some implementations clients may be configured to go to a centralised trusted responder, then passing the AuthorityInformationAccess to that trusted responder to fetch the status information. This approach can be used when a client needs to do certificate status checks for a certificate chain, the trusted responder gathers the status information for all certificates on the chain and then sends back one signed OCSP response to 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.

3.1.2 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.

3.1.3 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 exist as plug-ins to fixed Internet browsers – there are no mobile-specific implementations as yet. OCSP requests and responses are encoded using ASN.1 Distinguished Encoding Rules (DER)[3] encoding. Currently there is limited support for ASN.1 Basic Encoding Rules (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

[3] When the first recommendation on ASN.1 was released in 1988, it was accompanied by the *Basic Encoding Rules* (BER) as the only option for encoding. BER is a somewhat verbose protocol. It adopts a so-called TLV (type, length, value) approach to encoding, in which every element of the encoding carries some type information, some length information and then the value of that element. Where the element is itself structured, then the Value part of the element is itself a series of embedded TLV components, to whatever depth is necessary. In summary BER is not a compact encoding but is fairly fast and easy to produce. The Basic Encoding Rules come in three variants: BER – which allows options for the encoder, DER (*Distinguished Encoding Rules*) – which resolves all options in a particular direction, and CER (*Canonical Encoding Rules*) – which resolves all options in the other direction. That is DER and CER are unambiguous, since there are no encoding options.

support for ASN.1 encoding it will be easier to implement further ASN.1 support than to support another protocol altogether.

3.1.4 Bandwidth considerations

From [13], 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.

3.2 XKMS

3.2.1 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 Section 4.9.1 pf SHAMAN D04 [39]) and process a certificate chain path. It also allows for key registration. XKMS is specified in [47].

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 on the terminal.

3.2.2 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 response 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.

3.2.3 Current Support/Hardware support

Compared to OCSP, XKMS is a fairly new specification. It has been published within 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 with all other XML protocols, such as XML digital signatures [48]; 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 is perhaps more suitable, as limited support for ASN.1 encoding is already available on mobile handsets. WAP supporting handsets have support for WBXML, a 'compressed' version of XML where the XML is converted into binary digits. This, however, will not be useable for XKMS.

3.2.4 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 be 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.

3.3 Comparison and Conclusion

Signed OCSP responses and requests are seen to be nearly four times 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 preferable 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 be 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 [13] 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.

4 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 [43]), 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.

Some work on Personal PKI was already included in D07 [40], the 'Intermediate specification of PKI for heterogeneous roaming and distributed terminals'. For completeness, the main parts of this material have also been included in this document.

This chapter provides requirements and issues in the first two subchapters, discusses some of them in the following sub-chapters and finally introduces an alternative approach, namely ID-based cryptography.

At the end of the chapter conclusions and issues for further research are provided.

4.1 Issues in the personal PKI

We start by listing the various aspects of public key management for which solutions need to be found. More detailed discussions of the issues will be provided in the sub-chapters that follow.

- *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 same 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.

4.1.1 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 certificate for an 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 and the parameters of the keys are still considered sufficient 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.

4.1.2 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.

4.1.2.1 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.

4.1.2.2 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.

4.1.3 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 4.4.1.1 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 4.4.1.1 above will need to be used.

4.2 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 when 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 [43] 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 is 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 [43], 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.

4.3 Personal CAs

4.3.1 Operation of a personal CA

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

4.3.1.1 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 4.2 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.

4.3.1.2 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).

4.3.1.3 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 [43].

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 4.4.1 below.

4.3.1.4 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 in general is discussed in more detail in chapter 3, but solutions discussed there, e.g. OCSP, may not be appropriate within the PAN environment. Therefore a chapter on revocation in PANs was added (see 4.6).

4.3.2 Multiple personal CAs

Networks that consist entirely of mobile devices are necessarily of a more ad-hoc nature than fixed networks. Mobile devices that perform certain tasks in the network may simply not be present at all times. For example, an extreme case is presented by the fact that mobile phones are prone to theft. In the context of the personal PKI, the device whose absence most dramatically affects the operation of the system is the one acting as a personal CA. For this reason, we consider the possibility of having several devices within one PAN that can act as CAs. This redundancy makes the system more robust, since there is no single device whose absence would make secure communication within the PAN impossible.

In every PKI an implicit (but fundamental) assumption is that the CA is secure. In traditional PKIs, this is reasonable assumption, since a lot of effort is usually expended on keeping the CA physically secure. The same however cannot be guaranteed for the personal CA in a PAN. Mobile devices are prone to theft, and thus their security cannot be guaranteed. Therefore, we need to make the reason why a personal CA is not present in the PAN absolutely explicit. This case is typically characterised by one of the following two conditions:

1. The device is compromised or suspected of compromise, e.g. as would be the case if it has been stolen (it may or may not be absent). In this case, the user positively knows that the device cannot be trusted any more, and needs to transfer the CA functionality to another device.
2. The device is not present, but its security is not compromised. This is a more common situation, where, e.g., the device is simply switched off.

In this section, we describe (at a very high level) a solution for both situations. Throughout the section we make the following two assumptions:

a. There are two or more devices capable of acting as CAs. One is nominated as the *Primary CA*, and the others are *Secondary CAs*.

b. All CAs are known to every device in the PAN. More specifically, every device that enters the PAN is initialised with the primary CA, but is also given a trusted copy of every secondary CA's public verification key.

   Hence, in a PAN with multiple CAs, we suppose that the secondary CA(s) is (are) also known to every device in the PAN. Hence, if the secondary CA has to take over primary CA responsibilities, then every other device in the PAN will recognise it as a valid CA. At any time, all the CAs are kept synchronised. This can be achieved as follows: whenever the primary CA performs an operation, e.g., issues a new certificate, it informs the secondary CA, which keeps a state practically indistinguishable from that of the primary CA (e.g., the secondary CA has a list with all certificates issued by the primary CA). We can also suppose that the primary CA equips every newly imprinted device with a copy of the public key of all secondary CAs, at the same time as it transfers its own public key.

Of course, some additional organisational aspects have to be taken into account. A policy for the PAN-PKI-structure has to be specified. This shall cover issues such as what happens when a CA is compromised, and who defines which component is the primary and which components are secondary CAs. Furthermore an imprinting method for the CAs themselves has to be negotiated.

We now describe the actions that have to be taken when one of the above situations (1 or 2) arise.

4.3.2.1 Primary CA compromised

In situation 1, i.e., when the primary CA is compromised, rather extreme measures have to be taken. This is because the corrupted CA may corrupt the secondary CAs or even take over the PAN, if the secondary CAs are not notified immediately. Clearly, little can be done to secure the inter-PAN communications after the primary CA is corrupted and before the PAN is notified. However, one must make sure that once the PAN is notified, secure communications can resume. To achieve that, the whole system must come to the state prior to any initialisation. Thus, a list of valid CAs has to be entered into each device manually (or in any other secure way, e.g. as was used to originally initialise the devices). This list will obviously not contain the corrupted CA. A new CA is nominated as primary, and the devices are initialised one by one with the new primary CA.

4.3.2.2 Primary CA switched off

In situation 2, i.e., when the primary CA is secure but not present, the transition is smoother. Of course, one could treat this situation in the same way as the previous one, but this would incur unnecessary re-initialisations. After all, the absence of a CA may not be noticed at all by the devices in the PAN (i.e. if no device in the PAN needs the CA during its absence). This should be taken into account in the proposed solution; actions (and thus computational and communications overheads) should be kept to a minimum, and taken only when necessary. Following this principle, no action will be taken unless the secondary CA is contacted.

4.3.2.3 Synchronisation issues

In this section, we consider issues of synchronisation between the CAs of the system. As was mentioned before, all secondary CAs are kept updated by the primary CA while they are in the PAN. An issue that was left open arises in the situation where a (secondary) CA re-enters the PAN after a period of absence. This CA has not been updated for the duration of its absence, and therefore keeping it updated from this point on is not enough. Thus, at the time it re-enters the PAN, the secondary CA contacts the primary CA, and receives a signed list of the public keys of all devices, possibly including those that may not currently be in the PAN, but whose public keys are valid, and a CRL. From this point on, the secondary CA is kept updated as discussed before.

4.4 Device initialisation

4.4.1 A protocol for device initialisation

The security requirements for the device initialisation process have been listed in Section 4.2 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 [43]. We sketch this protocol below. The WP2 part of this deliverable (D13) describes in full protocols that can be used to securely transfer security parameters (e.g. a root certificate) from one device to another and/or ensure that both devices possess the same particular security parameter. These protocols can be used for two devices to exchange P_{CA} and P_M as described below.

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 discussed further below in chapter 4.4.2.

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

4.4.1.1 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 match 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.

4.4.1.2 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 4.4.1.1, 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 4.4.1.4 below).

4.4.1.3 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 4.4.1.1 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.

A detailed discussion of proof of possession in PAN-scenarios is in available in chapter 4.5.

4.4.1.4 Analysis of protocol

The purpose of the protocol described in Section 4.4.1.1 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}.

4.4.2 Initialisation methods for limited devices

In chapter 4.4.1 above, it was demonstrated how device initialisation can be achieved provided that the two communicating devices have sufficient input/output capabilities. In particular, it was assumed that they have numerical keypads and displays. The purpose of this section is to study the same problem in the case where when one of the two devices has very limited input/output capabilities. For the rest of the section we assume that one of the devices (the one acting as the personal CA) has both a numerical keypad and a display.

4.4.2.1 Case 1: no numerical keypad

Here we assume that the very limited device does not have a numerical keypad, but does possess a display. In this case, essentially the same protocol that was given in 4.4.1. In the description of the protocol, we adopt the following convention: device A is the very limited device, and device B is the device with both display and keypad, i.e. the CA.

1. Device A sends to device B its value X_A.
2. Device B sends to device A its value X_B.
3. Device A generates a temporary PIN K, and displays it.
4. Device A sends $MAC_K(X_A, Y_B)$ to device B, where Y_B is the value received by A.
5. The user enters K into device B.
6. Device B uses K to compute $MAC_K(Y_A, X_B)$, where Y_A is the value received by B.
7. If the received MAC matches the computed MAC then device B accepts, and notifies the user. If not, then device B rejects and notifies the user.

The correctness of the protocol rests on the observation that if the values X_A and X_B are not tampered with then the protocol terminates successfully. The protocol is also secure against online attacks because the attacker would have to intercept the MAC and substitute it with a value M, such that M= $MAC_K(Y_A, X_B)$, if device B is to accept (and the attack to be successful). However, finding such a value M can be done with a very small probability, since the key K is not exposed to the attacker. Of course, the key K can be recovered by an offline search (after the key exchange has been completed successfully), but this is not a problem, since a new value K is used for each execution of the protocol.

4.4.2.2 Case 2: no numerical keypad/display

Now we assume that the limited device has neither a numerical keypad nor a display. The problem now becomes considerably harder, as device A can only communicate over an unauthenticated channel. Note that in the protocol of the previous section, the assumption that A has a display provided an authenticated channel (namely the user), which could be used for very limited data (namely a short PIN). We see no way to achieve our goal unless we assume that this “user channel” is available. Our assumption for this section is that the very limited device A comes with a pre-installed PIN, which is known to the user. Then the protocol of section 4.4.2.1 can be used safely, but only once! This is because an offline attack will reveal the password to the attacker, who can then use it in subsequent executions of the protocol. One solution to this problem is the following: execute the protocol for the first time, to exchange the authenticated data. This data essentially give public key capabilities to the device A. The first thing to do then for device A, is to send a *new* encrypted PIN to the device B (the personal CA), which is securely stored. This is the new PIN to be used in case the private key of device A is compromised, and there is no other way to exchange authenticated data with device B.

Note that the new PIN will also need to be displayed to the user by device B, who will need to write it down and store it securely. The new PIN should not be stored by the CA device, since this would potentially make the PIN available to anyone who steals of compromises the CA, preventing the secure reinitialisation of device A.

4.4.3 The Maher Patent

David Maher filed in 1993 a patent regarding authenticated key exchange over an insecure channel (U.S. patent number 5,450,493). The scheme is based on a Diffie-Hellman key exchange. As the plain Diffie-Hellman protocol is subject to a man-in-the-middle attack, Maher's proposal includes a second stage for authenticating the established key. Let A and B be the two communicating parties who wish to establish a shared secret key. After the Diffie-Hellman exchange, party A possesses a value DH_A and party B possesses a value DH_B. If the communication has not been tampered with, the two values are the same. If, however, a man-in-the-middle attack has been mounted, then the two values are different. Maher proposes four different methods for discovering such an attack (and aborting the exchange).

In the first method, both parties use a suitable function f_2, and compute the so-called anti-spoof variables as $f_2(DH_A)$ (for party A) and $f_2(DH_B)$ (for party B). They then compare the computed values over an authenticated (but not private) channel, e.g., by speaking over the phone. If the values do not differ, then with high probability the Diffie-Hellman values also should not differ.

The second method is quite similar to the first one, but assumes that the two parties already share a secret, called the *Netkey*. This could, for instance, be a key shared by everyone in a given domain. Then the anti-spoof variables are computed using a function f_3 and the *Netkey*. Party A computes $f_3(DH_A, Netkey)$, while party B computes $f_3(DH_B, Netkey)$. Again the two values are compared using an authenticated (but not private) channel. The patent discusses briefly how this *Netkey* can be established; this key establishment has to be performed in a manner not vulnerable to a man-in-the-middle attack, e.g., by a Diffie-Hellman exchange followed by authentication using method one.

In the third method, the two parties make use of certificates to establish an initial common secret, which is subsequently used as a *Netkey* in method two.

In the fourth method, a somewhat different situation is considered. Suppose that party A wants to communicate the same message to two parties B and C in a secure manner. Suppose further, that A shares with B and C Netkeys N_{AB} and N_{AC}. The problem is that A does not want to encrypt the message differently for each recipient. A generates a secret key K, a random number R, and then encrypts the message using K, and sends it to the B and C along with a header. The header is sent in clear, and contains the following information:

1. R,
2. $h(N_{AB}, R) + K$ (for the message to B)
3. $h(N_{AC}, R) + K$ (for the message to C)

where + denotes bit-wise exclusive-or. The function h is the SHA hash algorithm. When B receives the message, he computes $h(N_{AB}, R)$ and recovers K. The same procedure is followed by C.

Comments and comparisons

We now provide some comments on the Maher patent proposals.

- **Method 1**. This scheme appears, at first sight, very similar to the Gehrmann-Nyberg (GN) imprinting scheme discussed in [11] and also in D07 [40]. However, there are some important differences. Suppose Method 1 is used in the PAN scenario, i.e. suppose two mobile devices first conduct a Diffie-Hellman (DH) key exchange over an (insecure) wireless link. Suppose also that both devices output a function f_2 of the keys established as a result of the DH exchange. Given that both devices are within the control of a single user, this user can verify that the two devices have the same key by comparing the outputs.

  The differences to the GN scheme become apparent when we consider possible choices for the function f_2. Note that we assume here that f_2 is a public and non-keyed function. To understand the requirements on f_2, we need to consider possible attacks against the scheme. Thus, suppose an attacker wishes to employ a naïve man-in-the-middle attack. Then the attacker can agree two separate DH keys with A and B, K_1 and K_2 say. If the attacker can arrange for the equation

  $$f_2(K_1) = f_2(K_2)$$

  to hold, then the attacker can launch a successful attack against the scheme. Hence f_2 must be collision-resistant (note that the attacker is not completely free to choose K_1 and K_2, since both are established as a result of a DH process where one part of the input is chosen by the genuine party).

  Thus, in particular, the output of f_2 must contain at least 100 bits or so. If it was significantly shorter than this, the attacker could simply perform a brute force search by generating a large number of candidate values for K_1 and K_2, until a match is found. Note that the only reason that the output of f_2 does not need to be even longer (say 160 bits) is that the attacker will have a very limited period of time in which to conduct the necessary search.

This contrasts with the GN scheme, which only requires very short values to be compared. This is much more attractive for the mobile scenario, where carefully comparing long strings of hexadecimal digits could become very onerous for the typical user. Also, the GN scheme works well when one of the devices is very limited, e.g. it does not possess a display, and the comparison is performed internally to one of the two devices. In such a case, keeping the comparison strings short is vital, to minimise the amount of data entry required.

We note that Maher proposes a way to use functions f_2 with a short output that resist man-in-the-middle attacks. To achieve this, one has to make it virtually impossible for the attacker to find a collision. In a traditional Diffie-Hellman exchange, the attacker can observe the keys of the two parties, and choose the values that he/she transmits so that the established keys (K_1 and K_2) satisfy $f_2(K_1) = f_2(K_2)$. To avoid this attack, Maher proposes that the two parties, A and B, should split their Diffie-Hellman values in two halves, and transmit them in two rounds. This way, the attacker can no longer choose his/her values so that a collision occurs.

This method allows for functions f_2 with short output, just as in the GN scheme. The disadvantage, however, is that, unlike in the GN scheme, the key exchange itself has to be modified and made more complex.

- **Method 2**. This would appear, at first sight, to be more similar to the GN scheme, since a key is involved. However, it would appear that the *Netkey* used in the Maher scheme is a full length cryptographic key. The great advantage of the GN scheme is that the key generated and transferred as part of the DH validation process is a very short transient key. This notion, which is at the heart of the GN proposal, would appear to be missing from the Maher proposals.
- **Method 4**. This would appear to be an old idea. It was certainly used in *Privacy Enhanced Mail* (*PEM*), the (now obsolete[4]) IETF security scheme for electronic mail – for further discussion of integrity issues in this context see [28] and [29].

4.5 Proof of possession

Proof of possession is required to demonstrate the knowledge of the private key as the counterpart of the public key sent in a request to a certifying party. This concept has been used in conventional PKI for a long time. Proof of possession in the personal PKI may be performed in a similar way. Nevertheless we have to analyse whether assumptions and requirements for the personal PKI will lead to a different view of proof of possession. Furthermore the scenarios where proof of possession is relevant have to be identified.

The idea of 'proof of possession' of a private key as part of the public key certification process now appears to be well-established. That is, to avoid certain 'source substitution' attacks on cryptographic protocols, it is generally accepted that it is good practice for a CA to ensure that the submitter of a public key knows the corresponding private key. This idea is now incorporated into PKI standards – see, for example, ISO/IEC 15945 [21].

Generally one can think of two different scenarios of source substitution attacks:

- o An attacker may request a certificate for a public key of another person also spoofing the other person's identity.
- o An attacker may request a certificate for a public key of another person without spoofing the other person's identity, i.e. using another identity.

[4] The reason that PEM is obsolete is that it is not compatible with MIME – this led to the development of the S/MIME scheme.

4.5.1 Motivation for establishing proof of possession

One example of why such a proof of possession might be useful is provided by the following description of a source substitution attack on the MTI/A0 key establishment protocol[5], taken from Note 12.54 (pages 518/519) of the Handbook of Applied Cryptography [23]. To put this attack into context we also provide the description of the MTI/A0 protocol provided in 12.53 of the Handbook.

We first give the protocol description.

Protocol MTI/A0 key agreement

SUMMARY: two-pass Diffie-Hellman key agreement secure against passive attacks.

RESULT: shared secret K known to both parties A and B.

1. *One-time setup*. Select and publish (in a manner guaranteeing authenticity) an appropriate system prime p and generator α of Z_p^*, $2 \le \alpha \le p - 2$. A selects as a long-term private key a random integer a, $1 \le a \le p$ -2, and computes a long-term public key $z_A = \alpha^a \bmod p$. (B has analogous keys b, z_B). A and B have access to authenticated copies of each other's long-term public key.
2. *Protocol messages*.

   $A \rightarrow B : \alpha^x \bmod p$ (1)

   $A \leftarrow B : \alpha^y \bmod p$ (2)
3. *Protocol actions*. Perform the following steps each time a shared key is required.

   (a) A chooses a random secret x, $1 \le x \le p$ - 2, and sends B message (1).

   (b) B chooses a random secret y, $1 \le y \le p$ - 2, and sends A message (2).

   (c) A computes the key $k = (\alpha^y)^a (z_B)^x \bmod p$.

   (d) B computes the key $k = (\alpha^x)^b (z_A)^y \bmod p$. (Both parties now share the key $k = \alpha^{bx+ay} \bmod p$).

The attack is then as follows.

Source-substitution attack on MTI/A0.

As a general rule in all public-key protocols, prior to accepting the authenticated* public key of a party A, a party B should have assurance (either direct or through a trusted third party) that A actually knows the corresponding private key. Otherwise, an adversary C may claim A's public key as its own, allowing possible attacks, such as that on MTI/A0 as follows.

Assume that in a particular implementation, A sends to B its certified public key in a certificate appended to message (1). C registers A's public key as its own (legitimately proving its own identity to the certificate-creating party). When A sends B message (1), C replaces A's certificate with its own, effectively changing the source indication (but leaving the exponential α^x sent by A to B unchanged). C forwards B's response α^y to A. B concludes that subsequently received messages encrypted[6] by the key $k = \alpha^{bx+ay}$ originated from C, whereas, in fact, it is only A who knows k and can originate such messages.

A more complicated attack achieves the same, with C's public key differing from A's public key z_A. C selects an integer e, computes $(z_A)^e = \alpha^{ae}$, and registers the public key α^{ae} . C then modifies α^y sent by B in message (2) to $(\alpha^y)^e$. A and B each compute the key $k = \alpha^{aey}\alpha^{xb}$, which A believes is shared with B (and is), while B believes it is shared with C.

[5] Thus this particular attack applies to key establishment key pairs.

[6] The key might also be used for MAC generation, thereby guaranteeing the origin and integrity of the message.

> In both variations, *C* is not actually able to compute *k* itself, but rather causes *B* to have false beliefs. Such attacks may be prevented by modifying the protocol such that the exponentials are authenticated, and binding key confirmation evidence to an authenticated source indication, e.g., through a digital signature.

The Handbook ([23], page 537) also points out that active attacks related to the above attack are considered by Diffie, van Oorschot, and Wiener [7], and Menezes, Qu, and Vanstone [24].

Although the above attack only applies to key pairs used for key establishment, other attacks can be constructed for protocols based on signature and/or encryption/decryption key pairs. One very naïve attack applying to signature key pairs is as follows.

> Suppose *A* wishes to send a secret message to *B*, and wishes *B* to make an appropriate reply. In order to ensure that the message is not available to anyone other than *B*, *A* encrypts the message using *B*'s public encryption key. In addition, in order that *B* can verify the origin of the message, *A* signs it using the private signature key of *A*.
>
> Meanwhile, malicious eavesdropper *C* has, by some means, arranged for *A*'s public signature verification key to be certified as belonging to *C*. *C* now intercepts the signed encrypted message and prevents it reaching *B*. *C* now re-sends the message to *B*, claiming that it originates from *C*. On receipt of the message, *B* verifies the signature using *C*'s public key, and verifies that it does indeed come from *C*. *B* now replies to *C* (instead of to *A*), and in doing so may reveal the contents of the secret message.

Finally note that it is considered good practice to design cryptographic protocols which are resistant to source substitution attacks – see, for example, [15]. Nevertheless, this does not mean that, for the moment at least, it is safe to omit the Proof of possession step, since protocols not protecting against such attacks may still be in use.

4.5.2 Assumptions and requirement for Personal PKIs

We start by considering the issue of key generation. There are various different ways in which a key could be generated. When proof of possession is being considered, it is important to take into account the place and the time that a key is generated. There are three main cases to consider.

- The key-pair is generated by the user's device. This situation is essentially the same as in fixed network scenarios. The certifying party has to use a proof of possession algorithm to obtain assurance that the requesting party is using a legitimate public key in the certificate request.
- The asymmetric key-pair is generated by the manufacturing party before the device is delivered to the customer. In this situation the need for proof of possession depends on the possibilities of the user to read out the public and/or private key from the device. Reading out the private key should not be possible for the user or an attacker. If the key to be certified is sent in a secured way, no proof of possession might be necessary.
- The asymmetric key-pair is generated by the certifying party when the certificate is requested. No proof of possession is necessary in this context as both parts of the key-pair are generated by the certifying party. In this case the establishment of an authentic and confidential channel for the transport of the private key has to be discussed.

The second major issue concerns the type of the public key to be certified, typically one of Encryption, Signature verification and/or Key establishment (e.g. as used in an authenticated Diffie-Hellman key agreement protocol).

If a public key to be certified is to be used for a particular purpose, then there may be restrictions on the way proof-of-possession is performed. For example, if a private key is used for signing, then the request for a certificate for the corresponding public key may be signed with the private key, whereas a private key only permitted to be used to perform decryption operations may not be used to sign such a request. To use the wrong kind of PoP-algorithm may result in a breach of security.

In some cases the use of the private key to sign the message may merely break key separation rules; in other cases it may simply not be possible, e.g. if there is no known digital signature algorithm which employs key pairs of the appropriate form. The typical case will probably be somewhere between these two extremes, in that, although a public key may be usable with a signature scheme, it may be usable with many such schemes, and it may not be simple to choose one. For example, in most public key cryptosystems based on discrete logarithms, the public key is equal to a base value raised to the power of the private key – in such a case, there will be a very large number of signature schemes for which the key pair would be a valid key pair. The problem would then be of coming to an agreement between signer and verifier about precisely which signature scheme should be used.

For any discrete logarithm based public key cryptosystems (including elliptic curve cryptosystems) the private key can be used to create an ElGamal signature on the certificate request, even if it is to be used subsequently as an encryption or key agreement key. However, this might be a problem, since it breaks the usual key separation requirements. One way of avoiding any problems might be as follows. Prior to computing the signature, generate a random value x, and if a is the client's private key, then generate the signature using $a+x$ *as the input*, and send x to the CA along with the signature and the public key g^a. The public key to verify the signature will simply be $g^a g^x$.

The different algorithms as discussed in 4.5.4 therefore have to be mapped to the kind of key they can be used for.

The third major issue is the nature of the algorithm to be used with the public keys. It needs to be investigated whether different algorithms specified to be used with a certain key make any difference in handling proof-of-possession. One possible criterion might be, whether a signing algorithm always produces the same signature for a certain input or whether the signature is different every time, as with RSA and El-Gamal based signatures respectively.

The fourth and final issue concerns what information is passed to the verifier. The party proving the possession of a private key sends some kind of information to the requesting party, i.e. the certifying party. The sensitivity of this information can be different using the several proposed mechanisms. Whereas in one case the information may be uncritical there may be mechanisms where the proving party is used to perform an action that enables an attacker to decrypt a certain message or in the worst case to compromise secret information. The proposed PoP mechanisms could therefore be further investigated in respect to zero-knowledge properties.

4.5.3 Analysis of the necessity of proof of possession in different scenarios

Certain scenarios may require PoP mechanisms whereas in other scenarios PoP is not necessary. Different factors may influence the necessity of PoP-mechanisms.

As already mentioned in the preceding chapter it is essential to consider where a key is generated.

When key-generation is performed by the certifying party and the key is sent to the client together with the certificate, a PoP mechanism is certainly not necessary as the issuing party has not to check the existence of the counterparts of private and public key.

If a key is generated at the manufacturer and pre-installed on a device before shipping, proof of possession may be necessary in the case an attacker has the possibility to get hold of the public key before the device reaches its owner. If this is not possible, proof of possession is not necessary if the authenticity of the key sent to be certified is secured with another mechanism. One example for such a mechanisms could be the following: a manufacturer of smartcards pre-installs cryptographic keys on his cards. In addition, a non-personal certificate is generated just to prove, that the public key sent for certification was generated by the manufacturing party. During the request, this non-personalised certificate is sent with the request. The certifying party then is at least sure that the owner of the card has sent some request and will get suspicious if he won't get a proper certificate from the certifying party. That means that the only risk left is a man-in-the-middle attack that is discovered very soon by the owner of the card. Furthermore the issuing party can be sure, that the public key to be certified has cryptographically good properties (as it was generated by the issuing party).

A variant of the mechanism described above could be as follows: the manufacturer stores a keypair on the card and certifies the public key before the device is delivered to the customer as described above. But in contrast to the last approach, this keypair is not intended to be used by the customer for other purposes than proving the possession of a key or the possession of the device.

Scenarios, where the key is generated by the requesting party, either on a hardware-device like a smartcard, or with software mechanisms, will be the most interesting scenarios for PoP considerations. Nevertheless there may be scenarios, where even here PoP are unnecessary. An example for the latter group could be a PAN-scenario, where people are in a local environment and can trust the transmission of data or the authenticity of the sending party with out-of-band mechanisms.

4.5.4 Proof of possession mechanisms

We now consider a number of different mechanisms for establishing PoP. All these mechanisms are somehow specialised in that they only apply to certain types of key pairs. It would appear rather difficult if not impossible to devise general purpose mechanisms for PoP, applicable for all key pairs.

1. Signature of the request or a part of the request

   In this approach a user generates a key pair or uses a key-pair that is already on his device or token. Before sending the request for certification of the public key, the request itself or a certain part of the request is signed by the corresponding private key and the signature is added to the request. If only a part of the request is signed, a dedicated solution might have to be developed, whereas signing the whole request may be done with standard signature-mechanisms and standard-applications.

2. Signature of a certain value derived from the request

   This solution works like the one described in 1. The difference is that the value to be signed is now not a direct part of the data of the request.

3. Signature of a value that is independent of the request

   This solution works like the one described in 1. The difference is, that the signed value is independent of the request-data.

4. Prompt the user to decrypt a specified challenge

   This approach has to be investigated properly as the user is giving away information by decrypting a value selected by the CA. This means that the user is used as an oracle. As the proof-of-possession is only performed once, during the certification-process, it may nevertheless be a useful method. This is especially likely to be the case if the decrypted message has to be in a pre-agreed format. In the case an attacker wants to use the user to decrypt some challenge, the decrypted value would not fit the format and the answer could be discarded by the party proving the possession of the private key.

5. Prompt the user to decrypt a certain value and send back a value derived from the result.

   To overcome the dangers as described under 4, the decrypted value could be put into a one-way-function before returning it to the CA. So if an attacker wanted to use the requesting party as an oracle, he would not get the original message but only the result from the one-way-function, that is not very useful for the attacker.

6. Prompt the user to decrypt a certain value and to prove the knowledge of this value with a zero-knowledge protocol

   To avoid the sending of any information related to the secret key but yet prove the possession of the secret key to the requesting party, a zero-knowledge protocol may be used. As with this kind of protocol no information about the secret (i.e. the decrypted value) is given away to the requesting party a potential attacker would get no information at all. Only the legitimate requester can use the information to verify that the proving party is in possession of the decrypted value.

The use of a zero-knowledge proof may, unfortunately, result in a more complex protocol especially concerning the number of steps to be performed.

7. Issue the certificate in an encrypted format, so that the requester is only able to get hold of the certificate if he is the owner of the private key.

This method may be used in certain scenarios, where the certificate is not published automatically. If automatic publication of the certificate takes place, the mechanism is of no use, as an attacker knows that the certificate was issued by receiving the encrypted certificate from the CA and would then be able to retrieve the certificate from the directory.

If the certificate is only distributed by pushing it and not available to pull from some directory, this method is very efficient.

4.5.5 Proof of possession in standards

Proof of possession is also an issue addressed in standards. The IETF has produced two RFCs deal with PoP.

- IETF RFCs 2511: Internet X.509 CRMF (Certificate Request Message Format).

  RFC 2511 discusses the PoP issue briefly and proposes some of the methods mentioned above. As discussed here, RFC 2511 distinguishes between the uses of a key, then basically signing and decrypting are the proposed mechanisms. As a third alternative it is proposed to compute a MAC on the certificate-request with a key derived from a secret, shared between the CA/RA and the requesting party. This approach may only be of little use in our scenario, as this assumption will probably not hold in PAN-scenarios. A message format for PoP is given as follows:

  The general structure lists the type of possible PoP-mechanisms:

```
ProofOfPossession ::= CHOICE {
    raVerified          [0] NULL,
    signature           [1] POPOSigningKey,
    keyEncipherment     [2] POPOPrivKey,
    keyAgreement        [3] POPOPrivKey }
```

  The format of the different mechanisms is a follows.

  (Some parts have been omitted, for a full description see RFC 2511)

```
POPOSigningKey ::= SEQUENCE {
    poposkInput           [0] POPOSigningKeyInput OPTIONAL,
    algorithmIdentifier       AlgorithmIdentifier,
    signature                 BIT STRING }

POPOSigningKeyInput ::= SEQUENCE {
    authInfo              CHOICE {
        sender                [0] GeneralName,
        publicKeyMAC          PKMACValue },
    publicKey             SubjectPublicKeyInfo }


POPOPrivKey ::= CHOICE {
    thisMessage         [0] BIT STRING,
    subsequentMessage   [1] SubsequentMessage,
    dhMAC               [2] BIT STRING }


SubsequentMessage ::= INTEGER {
    encrCert (0),
    challengeResp (1) }
```

- IETF RFC 2875: Diffie-Hellman Proof-of-Possession Algorithms:

This RFC provides two methods for generating an integrity check value from a Diffie-Hellman key pair. The two different approaches differ in the way they are using information concerning the receiver or not. The first solution produces a PoP value that can only be verified by the intended recipient, whereas in the second solution a PoP value is generated everybody can verify.

- ISO/IEC 15945: Information technology – Security techniques - Specification of TTP Services to support the Application of Digital Signatures

  As this standard only applies to signature keys, PoP is viewed from this angle. The mechanisms and the syntax used for PoP of the signature keys are similar to the ones from RFC2511, as described above. Of course, the syntax in this standard is reduced to the relevant parts, i.e. the parts describing the use of PoP for signature keys (signature [1] POPOSigningKey).

4.5.6 Efficiency of POP-mechanisms

In the table below, the main properties of the mechanisms described in Section 4.5.4 are summarised.

Mechanisms	Steps to perform	Computational complexity
1. Signature of the request or a certain value	Client: sign CA: verify	Two PK-operations
2. Signature of value derived from request	Client: sign CA: verify	Two PK-operations
3. Signature of an independent value	Client: sign CA: verify	Two PK-operations
4. Prompt the user to decrypt a challenge	Client: Send public key CA: send challenge Client: decrypt and return CA: verify	Two PK-operations
5. Prompt user to hash a decrypted value X	Client: Send public key CA: send challenge Client: decrypt X & hash CA: verify	Two PK-operations, Hash-computation is negligible
6. Zero-Knowledge proof	Depending on the concrete implementation	Depending on the concrete implementation, but probably higher than the other mechanisms
7. Encrypted issueing	CA: encrypt Client: decrypt	Two PK-operations

4.5.7 Suitability of the methods for mobile environments

To prove the possession of a private signing key it is probably the most convenient mechanism, to sign the complete request for certification. It is possible that the signature is only on the request-format, e.g. on a PKCS#10 structure or over a whole message containing the request. No particular dangers have been identified for this mechanism. To sign only parts of the request or an independent value is only slightly more efficient as the hash-function before signing might be performed faster, but the signature process is certainly less standardised than signing the whole message.

In case of PoP for an encryption key, the mechanisms have to be investigated more thoroughly. The most critical factor in this context is the leakage of information from the proving party. This potential danger can be overcome by using zero-knowledge protocols, but these protocols are generally more complex and time-consuming.

The efficiency of the proposed methods can be summarised as follows: Just signing the request or parts of it requires two steps and two PK-operations to be performed, the signature by the requester and the verification of the signature by the certifying party. The number of steps grows to four, where the decryption of a challenge is part of the PoP process, though the number of PK-operations to be performed does not change and is therefore two.

4.6 Revocation in PAN-PKIs

4.6.1 Assumptions and requirement for Personal PKIs

- Structure of the PAN

  A PAN is usually a relatively small structure so that there may be the possibility to implement the management of revocation information in a different way to conventional methods, where a large number of entities are involved.

- Availability of all PKI-users to the CA

  In a PAN the availability of the users might be better than in conventional PKI-structures, as the number of users is essentially smaller. That means all users can be reached at a certain time or the CA can keep track of which users haven't received revocation information, so that the CA can pass on the information at the next login.

- Structure of the used certificates

  The structure of the used certificates might be adapted to certain requirements and possibilities in a PAN. This aspect is discussed in more detail in chapter 4.7.

4.6.2 Analysis of the necessity of revocation in different scenarios

The issue of revocation might be more important in certain PAN scenarios than in others. This directly depends on the certification policy and the environment the certificates are used in.

Some aspects regarding the scenario are given in the following sub-chapter.

4.6.3 PAN-specific vs general revocation mechanisms

In conventional PKIs mechanisms to get hold of revocation-information one usually provides pull-mechanisms like the provision of revocation lists or online status services. This is the case due to the following reasons:

- Often a large number of clients takes part in the PKI. That means push services would lead to a bandwidth-problem and cannot be managed by the CA. A broadcast service could be implemented, but then clients not logged in during the broadcast have no chance to get recent revocation information.
- Large PKIs are usually long-time-PKIs where revocation-information has to be provided on a long-time-basis
- Not all clients are always-online so to receive all revocation-info that is distributed
- Not all clients are interested in all the revocation information at a certain time. So a broadcast might be inefficient.
- The revocation information gets that large, that not all clients are able to cache the information. So there has to be the possibility to reload the desired information.

In contrast to conventional PKIs push-services may probably be implemented in Personal-PKIs as the following assumptions hold:

- Personal PKIs may be short-time-PKIs. Therefore revocation-information may only be relevant for a short time. The issuing of 'renewed' certificates on a more frequent basis might not be a problem due to the small number of certificates in the structure. In full consequence one could propose to introduce short term certificates that are only valid for a short period of time and have to be renewed e.g. every day, like it is already done in the SSL-server-certificate context. This could result in the negligence of revocation mechanisms at all.
- Only a restricted number of parties take part in the Personal PKI. Therefore more time- or resource consuming mechanisms (like the issuing of short term certificates as described above), that don't scale in large PKIs, might be feasible to be implemented in the PAN scenario.
- When components make use of Personal PKI they are online. Therefore revocation information can be pushed to them when logging in to the Personal PKI. The limited number of PKI users in a PAN makes it possible for the CA to keep track of the users and the revocation information they have already received.

Mechanisms adapted from conventional PKIs

As already discussed in the chapter 3, revocation mechanisms used in traditional PKIs can be used to do revocation management in Personal PKIs as well. The two general approaches using CRLs or requesting status information online can be implemented. Here is a short summary of the discussion of relevant issues from this chapter:

The use of certificate revocation lists has only been discussed briefly in D07 [40] as there arise problems with handling large amounts of data in mobile environments. Therefore, the Online Certificate Status Protocol (OCSP) as well as The XML Key Management System (XKMS) have been looked at in more detail. The conclusions were as follows:

- The format of OCSP-messages is far more efficient that the format of XKMS-messages
- The encoding format of OCSP-messages (ASN.1) is better supported in mobile environments than the XML-format.
- XKMS provides more services than OCSP and makes it possible to delegate tasks to a trusted third party.

PAN-specific mechanisms

Due to the characteristics of A PKI within a PAN, i.e. a personal PKI, new mechanisms to manage the revocation issue which may not be implemented in traditional PKIs can be proposed:

Generally the new situation is, that the PKI is a relatively small structure. Therefore you can think about implementing some kind of push-mechanism. That means, that a member of the PKI gets informed automatically about recent revocation incidents. Several models about how this could be done have to be considered:

- CA-based distribution models:
    1. Automatic distribution of newly generated CRLs
    This requires the CRLs not to be too big. It is not the most elegant way, as the CRL-concept was intended to be a pull-concept. The major advantage could be that the CRL-concept is more up-to-date as the lists can be pushed when a new entry has been added. Of course this mechanisms is only efficient when there are not too many revocations.
    2. Automatic distribution of new revocation incidents
    This solution requires to introduce a new protocol/application which is able to store single revocation incidents or put them together and store them authentically on client-side. In comparison to the distribution of complete CRLs it's a more efficient way to distribute

only recent incidents, but as we have learned in the past, the introduction of new functionality on the client is always problematic.

3. Automatic distribution of CILs (Current Identity Lists)
   Instead of 'black-lists' we introduce the distribution of 'white-lists'. This has the advantage, that a user can be sure that a certificate is not revoked **and** that this certificate was issued by the concerned CA. The use of white-lists makes only sense when the number of participants is not too big. It has to be specified when the white-lists are published. Of course the current list has to be sent to new members and members re-entering the PAN. A question in this context is, whether clients already logged in get the whole list again when a new member enters the PAN or whether there may be another mechanisms just announcing the new member similar as the proposed mechanisms to distribute revocation incidents only (see above).

- Ad-hoc-distribution of CRLs (as already introduced in D07 [40])
  1. This approach was already discussed in D07 [40]. To summarise: the idea was, that the distribution of CRLs is done between the clients, so that the necessity to contact a directory is not given. The actuality of the CRLs is handled by introducing serial-numbers, such that client update always to most recent version.

Local caching in clients possible

An important question for revocation in PAN-scenarios is whether very limited devices can cope with certain revocation information at all. If we think of very limited devices storage space may be an issue. Therefore the caching of CRLs may not be possible on every device. Usually revocation lists will not get very big in PAN-scenarios as there are only a small number of components taking part in the local PKI, nevertheless there may be scenarios where a list can get longer. For instance, if you consider a PAN, having a lot of guest-members. The question here is, whether the personal CA has the capability to issue a certificate to those parties having only a short validity period; therefore avoiding the necessity to revoke a lot of certificates.

Support of different mechanisms by one CA necessary or favourable?

As discussed in the subchapter before, mechanisms may be chosen, that are favourable for a certain class of devices whereas they may be unfavourable or even infeasible for another class. Thus the necessity to integrate at least two revocation mechanisms in one PAN have to be discussed. With the introduction of different mechanisms there may arise new issues:

- Additional logic could be necessary to be implemented on the personal CA device.
- If push-mechanisms are used, devices in the PAN must be able to cope with them, i.e. evaluate the pushed information or discard it if an evaluation is not possible.

4.6.4 Suitability of the methods for mobile environments

PANs offer a lot of possibilities to implement revocation mechanisms differing from the ones known in fixed network scenarios. Presently the different mechanisms are only proposed mechanisms that have still to be implemented. It will be difficult to implement such mechanisms that copcern the client-software as this functionality had to be standardised. Nevertheless, as PAN-scenarios get more and more attention in the mobile world and will be an essential part of tomorrows mobile infrastructure, it's necessary to approach these concepts and to develop solutions that may be more efficient that the ones available today.

4.7 Certificate formats in the personal PKI

The format of certificates is always an issue to be discussed when a new variation of PKI or a new kind of service is introduced. A lot of ideas have been introduced in this context, but only few of them have become widely accepted. This is for interoperability reasons. Often, common certificate formats have been profiled for certain requirements/scenarios. In the personal PKI we have a structure that is locally restricted. One might say that this is an argument in favour of introducing a new certificate-format. The counter-argument is, that PANs are local structures but that there is inter-PAN communication which makes the scenario again more global. So to introduce a new certificate format we will have to illustrate a real additional value. This could be, for example, a simplification of revocation management, the reduction of the certificate size or the possibility to integrate additional functionality.

4.8 An alternative approach: 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.

4.8.1 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 [23], 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.

4.8.1.1 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 [3] and Cocks [6].

4.8.1.2 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 he 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, [8], [9] and [20].

4.8.1.3 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.

4.8.2 Practical issues

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

4.8.2.1 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.

4.8.2.2 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.

4.8.3 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.

4.8.4 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.

4.8.4.1 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.

4.8.4.2 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 [43]. 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 is described in more detail in the WP2 part of this deliverable (D13] under the term, the MANA protocols, where "MANA" is short for "MANual Authentication". 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,δ) are the same. If they are the same then the received values are indeed those transmitted. If not, the process is repeated.

4.8.4.3 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.

4.8.4.3.1 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 [43].

4.8.4.3.2 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 4.8.4.2. 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.

4.8.4.4 Key revocation

As explained in Section 4.8.2, 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 4.8.4.3.1. 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 4.8.4.3.1. 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.

4.9 Comparison of ID-based cryptology and traditional PKI

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.

4.9.1 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.

4.9.2 Initial exchanges

As discussed in Sections in the previous chapters, 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.

4.9.3 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 [3], 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).

4.9.4 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.

4.9.5 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 an 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.

4.9.6 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.

4.10 Conclusions and issues for further research

The investigation of Personal PKI issues has shown, that there are differences in the requirements and potentials for new mechanisms to be implemented compared to a more global-scaled PKI. Especially the issue of revocation in the PAN offers the possibility to look at new approaches. Furthermore it became clear, that these new approaches need to be discussed and respective standards have to established.

In the following paragraph the results, concerning the different sub-chapters is presented:

Personal CAs: The most interesting aspect of CAs regarded in the context of Personal PKIs is the concept of multiple CAs. That means, due to the restricted availability and the danger of lost or stolen devices, mechanisms have to be specified that enable CAs to hand over possibility and to synchronise after a period of absence from the PAN. On the client side this concept has also to be introduced as a hand-over of CA-duties is not common in traditional PKIs.

Device Initialisation: Protocols for device initialisation on very limited devices can be specified. One protocol for devices without keypad but with a display was derived from the protocol shown in 4.4.1. Another approach for devices without display and keypad was introduced, where new credentials have to be sent to the CA after using the credentials available after shipment for a first initialisation.

Proof of possession: The discussion on proof of possession has shown, that the requirements and the methods known from traditional PKI in fixed networks can mostly be transferred to the PAN scenario.

Revocation in PANs: Due to the nature of PANs, the considerably small number of participants and the fact, that no ‘offline-mode’ is necessary, new mechanisms for revocation can be introduced. These mechanisms are not yet standardised and may therefore be of interest for standardisation groups.

ID-based cryptographic systems: In ID-based cryptographic systems the public key is not distributed in certificates but derived from a unique User-ID. Therefore certificates don’t have to be distributed. This results in less communication overhead. The computational complexity of both approaches is about the same, as mostly the same operations are performed. In contrast to traditional PKI the ID-based solution needs a much higher trust into a third party, as this party knows all the private keys.

5 PKI for secure execution environments

This section examines issues concerned with the use of PKI for securing execution environments (for example, Java Virtual Machines) on mobile devices. Security issues with execution environments in general, not necessarily those relating to PKI, are also, as a matter of relevance, dealt with.

We begin in section 5.1 by examining the security requirements for execution environments that were first recorded in SHAMAN deliverable D03 [38], and determining which are met by Shaman recommendations and/or by existing standards and products.

It turns out that all the requirements which are not met relate to authorisation. We therefore next examine the issues that authorisation raises for secure execution environments in section 5.2.

There is a two way relationship between secure execution environments and authorisation in that the post-factory flexibility in terminal functionality engendered by execution environments raises issues for authorisation (and authentication) in general. This is examined in section 5.3. We examine, for example, what "device authentication" can really mean if the behaviour of the terminal is highly flexible.

Finally, section 5.4 examines a collection of practical issues with the use of PKI for secure execution environments, and looks at solutions to these issues and alternatives to the use of PKI. This section, in large part, is a result of "brainstorm" session between some of the Shaman partners so should not be seen as a final, fully resolved set of proposals, but rather as a collection of open issues and partial solutions, recorded here for the benefit of those conducting future research in this area.

5.1 Requirements review

This section will examine those WP2 requirements which were both listed in SHAMAN deliverable D03 [38] and considered in SHAMAN deliverable D04 [39], as requirements which can be satisfied using asymmetric techniques. In particular, those requirements relating to secure execution environments are considered here. This work was conducted in response to a suggestion that the use of asymmetric techniques to satisfy these requirements would benefit from further analysis.

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 D03 [38].

5.1.1 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)[7];

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

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.

5.1.2 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).

5.1.3 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).

5.1.4 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 D04 [39]. 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.

5.1.5 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 [1] 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 on the terminal. However, the CCM does not use standard formats and is unpopular with terminal manufacturers. The mechanism for deciding who the Administrator is (the Administrator serves within MExE a role similar to the PAN Manager), 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 5.2.

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 [45], in that a Trusted CA (the term used in [45] 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 is an area for further study.

   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 is examined in section 3.

5.2 Authorisation issues for Secure Execution Environments

A major issue with MExE [1], and also now with MIDP NG [25] 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 3.

(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 become 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.

5.3 SEE issues for authorisation

The previous section dealt with some authorisation issues for SEE. This section goes in the opposite direction and considers some issues for authorisation, and authentication, that are raised by the presence of an execution environment on the device, whether secure or not. More correctly, it is not execution environments that raise the issue but the presence of flexibility in terminal behaviour in general.

We will explore this issue using two examples, Digital Rights Management and Bluetooth, and then explore the general issues that these examples raise. Finally, we will examine some ways that public key techniques can be used to resolve this issue.

5.3.1 Example 1 – Digital Rights Management (DRM)

Digital Rights Management (DRM) is about controlling the use on a terminal of valuable content. For instance, a record label may wish to sell singles as MP3 files over the internet, but does not want those who purchase the MP3 files then make arbitrary number of copies for their friends and so reduce the number of paid-for sales of the single. DRM is envisaged as way to control the behaviour of the terminal so that it would not allow such copying and distribution of the single. More positively, DRM might also be seen as a way to enable new business models, in that a piece of content could be sold, for a low price or for free as a "preview", that is, the user would obtain the content but would only be able to play it once. If the user subsequently wants to play the content more, she can buy a license that will allow her to play the content more (either for a set number of plays, a set time period, or for as often and for as long as she likes). Without DRM, this business model would not be possible as once the user had obtained the content for the preview, they would then be able to do whatever they liked with it.

However, the big problem for those wishing to sell content to devices claiming to follow a particular DRM specification is assurance that the device actually will follow the specification and will observe the conditions on content delivered to the device. The OMA "phase 1" DRM specifications, [32], give rise to this problem for content owners. The specifications provide only "identification" of the terminal type (whether a Nokia 7650, software v3.6.2 or a Siemens SL45i, software v2.4, for instance) in the headers of a http request for content – there is no cryptographic authentication of the terminal type. Hence, a content provider may think that they are sending content to a compliant terminal that will observe the conditions attached to some content, but in fact content is being sent to a PC with a terminal emulator which has been programmed to completely ignore the DRM information attached to the content.

Work on "phase 2" of the OMA DRM specifications is currently in progress. Device type and device authentication are widely seen as the most important part of the phase 2 work but if the behaviour of the device can be varied after manufacture to give it non-compliant DRM behaviour, then knowing that it is a particular device type, which had compliant DRM behaviour on exit from the factory is of limited use to the content provider.

Device behaviour can be modified in a number of ways. Examples are:

a) Devices with a proprietary "closed" OS can be reflashed, so that the OS is completely or partially replaced by another OS[8].

b) Devices with a closed OS but with an execution environment, for instance, a Java virtual machine, on board.

c) Devices with an open OS (e.g. Windows, Symbian EPOC) and an execution environment.

d) Devices with an open OS that allow download of applications.

Depending on the design of the terminal, any one of these ways could be used to "subvert" the DRM behaviour of the terminal so that it is no longer compliant. For instance, an application could be downloaded that takes an encrypted file, say, an MP3 of a single from one part of the terminal file store, the content encryption key (CEK) for that piece of content from another part of the file store and then calls a decryption function to decrypt the content.

One way to reduce the occurrence of this problem is just to make terminals less flexible. However, the trend is exactly the reverse. Java is no longer just on high end phones or PDAs but on mass market terminals. Similarly, open OS such as Windows and Symbian are now appearing on phones and not just on PCs or PDAs. There is every indication that these OS will in time appear on mass market phones and not just high end phones. Therefore, reduction in terminal behaviour flexibility is not a realistic option.

Instead, we must find ways for device behaviour flexibility to be controlled, and for device authentication protocols to support indication/authentication of this controlled and compliant device behaviour flexibility.

5.3.2 Example 2 – Bluetooth

A similar issue is raised with regard to Bluetooth. The Bluetooth specifications allow for authentication of **devices** (not subscriptions or SIMs) and are probably the reason for most device authentication that occurs at this time.

The current Bluetooth specifications give three categories which a Bluetooth device (call it A) can use to specify their "trust relationship" with another device (say B). These are (see D03 [38] for further details]):

- Unknown: A cannot authenticate the identity of B;
- Untrusted: A can authenticate the identity of B but the user of A has not awarded "trusted" status to B;
- Trusted: A can authenticate the identity of B and the user of A has awarded "trusted" status to B.

The question can be asked, on what basis does the user of A give device B a trusted status? Presumably it is because the user of A knows the user of B (perhaps they are the same person or friends or work colleagues) and trusts the user of B and therefore trusts that the behaviour of device can also be trusted. If device B is a mobile phone with no execution environment and where the Bluetooth behaviour of B is fixed and in accordance with the specifications, this attitude by the user of A is probably fine. The user of A is, though they may not realise it, trusting the manufacturer of

[8] Reflashing usually just involves modification of key bytes in the device OS, for instance, those bytes which control "SIM lock" behaviour of the terminal. SIM lock is used to tie a terminal to a particular operator's SIM.

device B to have built a compliant device, and the user of B not to misuse any of the data it is sent from A which device B will give the user of B some control over, and both of these are probably reasonable to trust, given knowledge of user B by user A. However, if device B is an open OS device, where there are APIs into the Bluetooth functionality of the phone from execution environments on the device (there are such APIs being defined within the Java Community Process, see [22]) then in addition to trusting the device manufacturer and the device user, the user of A is also trusting every piece of downloaded software on device B that can access Bluetooth functionality and indeed, any future piece of software that is downloaded to B during the duration of the relationship of devices A and B[9]. Indeed, since Bluetooth functionality might be used to bring sensitive data from device A into device B where it can be accessed by downloaded software on device B, then the user of A is trusting every piece of software on device B that has access to this data, and not just the software that has access to Bluetooth functionality.

Hence, as device behaviour flexibility continues to increase and become more prevalent, the trust categories of Bluetooth may become deficient in at least two ways:

- There will not be enough categories to reflect the real levels of trust which will exist between Bluetooth devices.
- The user will not be given enough information, or perhaps will be given too much information, on which to base their decision on whether another Bluetooth device can be trusted or not. (Indeed we can ask whether it is appropriate that the user is asked at all).

Hence, we can see the device behaviour flexibility, created by the presence of execution environments on terminals, has rendered device authentication less meaningful than it otherwise would be.

5.3.3 The meaning of (device) authentication, and the impact of SEE on this

It is appropriate at this point to consider why authentication is conducted at all. Authentication by a particular entity is not just conducted on another to give proof of the other's claimed identity but because the first entity is going to use the identity of the second for some purpose. If all the first device was going to do with the authenticated identity of the second was just to temporarily store the identity in memory and then delete it, it would not conduct authentication but would simply accept the claimed identity. However, where an identity is authenticated it is because we wish to do something significant with an identity, for instance, we wish, in the case of GSM or 3G cellular access, to bill events to the identity of a (U)SIM. Or, in the case of DRM, the content provider wants some assurances about the behaviour of a device before sending valuable content to that device, so they authenticate the claimed device type identity.

With regard to device authentication, as there are yet no applications for billing based on device identity, the primary reason for device authentication is to get assurance, or at least information, about device behaviour.

5.3.4 Some possible ways forward, and prerequisites for ways forward

In both cases above, we have seen that flexibility of device behaviour has rendered device authentication less meaningful, as the flexibility of device behaviour casts doubt on the assurances about device behaviour that the device authentication was designed to give.

As device behaviour flexibility cannot happen in arbitrary ways, but in particular ways enabled by the OS of the device, there are a number of ways forward here:

However, there seem to be at least two prerequisites for the effectiveness of these ways forward.

1. It must be possible to authenticate what is the basic determinant of behaviour on a device, the source of all device functionality, any possible flexibility in device functionality and the source of all assertions about the device functionality, that is the "base" device OS. For

[9] In fact, if device A leaves some sensitive data on device B even after the termination of their relationship, device B is being trusted for all time.

instance if a device authentication protocol informed an operator that a device possessed a MIDP 2.0 Java Virtual Machine (JVM), and that this JVM supported a certain number of JSRs, this would not help the operator that much as it would have no assurance about the sub-entity within the device that was making this claim, that is, what the base OS of the device was. This information might be given explicitly (e.g. a certificate for a device might include the string "`Symbian EPOC 7.1`") or implicitly in that the device might state its model, e.g. "`SonyEricsson P800`" and then the authenticating party looks up what the base OS of a SonyEricsson P800 is.

2. This first assertion about the identity of the base OS of the device will probably be made by the device manufacturer. However, this requires every authenticating party to understand the assertions of all possible device manufacturers and also to have a view on the trustworthiness of each device type. In the case of DRM, this would be a lot to ask of a small content provider (e.g. a band distributing its own songs).

   For the continuing usefulness of device authentication, it must therefore be possible for an entity other than the device manufacturer to also make a trusted assertion about the trustworthiness of a device. This would allow, for instance, an operator to declare as trusted all devices that had passed its tests. Partner service providers of the operator could then just authenticate the presence of this assertion by the operator instead of (or perhaps in addition to) authenticating the assertion made by the device manufacturer.

If these two prerequisites, or at least just the first, were in place, there seem to be two basic approaches to the problem.

Constrain the execution environment

This is the simpler of the two approaches. The basic approach would be to ensure that the range of flexibility of the device behaviour did not include behaviour that was considered sensitive. In practice, this would mean not having APIs from the execution environments to the functionality in question. So, with regard to take Java and DRM as examples yet again, this would mean not defining any Java APIs to any DRM functionality.

There are a number of problems with this approach:

a) This reduces the usefulness of execution environments. Execution environments have been put on mobile devices as this makes it easier to download new applications such as games, and perhaps in time, to write core device functionality that will be present on exit from the factory. Any approach that sets up blocks to the expansion of execution environment functionality is therefore intuitively inconvenient.

b) It is hard to say in advance which behaviour is sensitive and therefore to be denied to execution environments. Manufacturers and execution environment specification bodies may therefore work together, with a certain view of what is sensitive terminal behaviour, and implement devices where there is no execution environment access to what is considered sensitive terminal behaviour. However, during the lifetime of the device, it turns out some functionality open to the execution environment can be abused and we therefore wish to make it "sensitive" and denied to the execution environment. However, this cannot now be done[10]. Further, there will be problems in making this behaviour "sensitive" for future terminals since there will then be some devices where software can do certain things and other devices, more modern, where ironically, there is less functionality.

c) Sensitive functionality can sometimes be constructed from an amalgamation of non-sensitive functions. For instance, the auto-invocation of midlet possibilities within MIDP 2.0 [26] can be combined with the possibility of network access with blanket permission

[10] Unless the device behaviour flexibility includes flexibility in the flexibility! That is, if it is possible to change security policies within the execution environment after the device is in the field. This has been discussed within the JCP.

(where the midlet need only ask the user once for permission to access the network) and with control of the screen, which also individually would not be considered as functionality that could be denied to midlets, to give a midlet that can be activated at any time and make calls without any permission from the user or indication to the user on screen.

Restrict execution environment of sensitive functions to authorised parties only

In this approach, software running in the execution environment would have access to sensitive functionality, but only if the software came from an authorised source. This seems less of a constraint on the future of execution environments but also has a number of problems:

d) Who decides whether a source is authorised or not? There are clearly commercial implications here so the issues with authorisation that are discussed in section 5.2 with arise again.

e) How is the authenticating party informed of all the changes to the sensitive behaviour of the device that have been made by the authorised parties?

f) The combination problem may arise again, in that a combination of software from a number of authorised sources may, either deliberately or accidentally, produce non-compliant behaviour.

It seems likely, that for the short term, a combination of the approaches will be used, as there is not yet a structure that can be used for a full implementation of the second approach. As execution environments/device behaviour flexibility exists in more than one level within a device, this constraint will also exist at more than one level.

At the level of the base OS, it is likely that the second approach will be followed, and only the device manufacturer will be able to make changes to the base OS.

At the level of a “traditional” execution environment, it is likely that a combination of the two approaches will be used. Some functionality will be denied to the execution environment. Other functions will only be allowed to authorised parties, or will be given to unauthorised parties with restrictions. An example of the use of such an approach is MIDP 2.0, where only certified parties whose signatures can be verified by a certificate chaining to a root certificate on the device that is defined to be controlling a particular execution domain are considered as “Trusted”.

5.4 General issues with, and alternatives to, the use of digital signatures and certificates for SEE

Section 5.2 of this document raised a number of authorisation issues with the use of PKI for securing SEEs. D04 [39] also identified a number of other issues with digital signatures and certificates in general. In summary these were:

- The complexity of PKI for limited devices (this has been dealt with in section 6 of this document);
- The clash between trust and commercial models (this has been dealt with in section 5.2 above);
- The frequent absence of revocation solutions (this has been examined in section 3 of this document);
- The problems with interoperability of certificates and PKIs;
- The problems with the use of extensions in X.509;
- The problem of non-compliant providers and users of PKI;
- The effect of certificate lifetimes.

Since the use of PKI is clearly a difficult thing to get right, it is worth asking the question in the SEE context, *is it really worth using for making execution environments secure?* We will examine this question by first considering some alternatives to the use of PKI for SEE, and then examine a number of general issues with the use of PKI for SEE, in a discursive fashion, and not so as to arrive at any final conclusions.

5.4.1 Alternatives to the use of PKI for SEE

One answer to the question, is the use of PKI for SEE really necessary might be: 'Yes, to stop viruses and malicious programmes'. In fact, this answer is at the heart of the use of PKI for SEE.

We will consider two alternatives to the use of PKI for avoiding viruses and the like. They are:

- Don't trust the code – examine it;
- Use non-PKI trust methods.

5.4.1.1 Don't trust the code source – examine the code

One alternative to trusting the source of the code (this is what PKI gives you) is for the device to actually examine the code itself before executing it. An interpreted, non-compiled language like Java lends itself to this more than compiled languages, where the alternatives are for the device to examine compiled code or for the application developer to actually send source code to the device, something she might not be willing to do.

Though a challenging problem, it does not seem impossible that heuristic, AI-based code checkers could be developed to enable devices to examine code and see if it were a virus or malicious program before installing and executing it.

If this task were initially too intensive for a device to do on its own, a server could be used to provide the code checking service, either at the request of the device (the code or a URL for it being passed to the server) or as a measure enforced by the server, which would sit between the device and any source of code it might be used, and check all code that users attempted to download.

Since, in many cases, users will be downloading code that has been downloaded by at least one other user, the server assisted method of checking for code can be enhanced by the device just passed an identifier for the code (a URL or a hash of the code) to the server and this returning a yes or no to the device. Similarly, for enforced checking by the server, the server need not examine all instance of code that it sees but could simply generate a hash of the code and see if that code hash had been seen before. If it had, the previous examination result can be retrieved.

This use of code identifiers and hashes can also be made available in the form of white and black lists of respectively, safe and unsafe code instances. These lists could be downloaded periodically by devices. This is basically the method used, with regard to blacklists, by existing virus checking agents.

Note that the white and black list methods depend, if code identifiers and not code hashes are used, as do virus checking mechanisms, on the assumptions that the code has not been modified since it was assigned its identifier.

Further, the whole use of code examination instead of PKI means that code integrity is not protected by the code signer's digital signature but must, in the presence of threats to integrity (which will not necessarily exist, for instance in peer to peer sharing among friends over local links), be given integrity protection by other means. This could be provided using transport layer security for example.

There is promising assistance for the method of code examination by the use of formal proofs of a code's safety accompanying the code. This notion is explained in [31].

5.4.1.2 Use non-PKI based methods

PKI based methods for SEE security mean that in the end the user is trusting one CA or another. As examined in section 5.2 above, this is not something users probably realise they are doing. This lack of information on behalf of the user is probably not a problem for most users while things are going well, but if users end up download signed viruses and losing services, data or money, they will probably be annoyed to know that their device was programmed to trust some entity that the user did not trust, or even know about.

We might therefore wonder if we can replace a system based on user unknowingly trust entities they don't know to one based on users knowingly trusting entities they do know. Viewing peer to peer file sharing methods and PGP-like webs of trust as existing examples, we can imagine systems where users might trust some code because they have received it (with, in the presence of threats to integrity, some form of integrity protection) from another user that they trust.

This decision of whether the user trusts the sending party could be made at the instance of code reception by the user or could be made automatically by the device on the basis of some list of trusted sources in the device. Again, this list could just have been explicitly entered by the user, or could have been generated by the device itself based on, for instance, the contents of the device phone book and some record of which numbers the users calls most (frequency of contact indicating trust). Other additional sources of trust information on particular users or devices could be frequency of SMSs sent to or received from certain users/devices, whether code received from a particular user has worked well or not, frequency of contact via a local link.

On the other hand, immediately after the device is released from the factory, it will have no "experiences" on the basis of which to make intelligent decisions about whether to trust other entities or not, the device will be very like a small child. In such a case, a possible solution is that we give this child a mother! The mother could then help the "child" take decisions or completely take over the decision making process for the "child". The mother in this case could be some trustworthy entity such as the operator that the device's cellular subscription is with (if the device is a mobile phone), the device manufacturer or some other, "more experienced" device owned by the user. There are definite similarities here with the Resurrecting Duckling model proposed by Stajano and Anderson [44] and explored in WP2.

In addition, it should be noted that the processes of designating trust to those entities in the lists of trusted sources in the device need not just have binary trust settings, trusted or untrusted. Concepts of "fuzzy trust" could be developed so that entities could be signed a trust rating, say between 0 and 10. A minimum trust level could be set for a number of categories of code functionality, with the minimum trust rating required for the code to be seen as trusted rising as the privilege or functionality of the code rises.

It should also be noted that PKI can be integrated into this system. A piece of code signed directly by the user's operator or device manufacturer might be assigned a trust rating of 10 for instance, whereas some signed code from a developer the user has never used before, and where there is no certificate revocation information might only be assigned a rating of 5. In comparison, code from passed by local link from a device that the user's device communicates with every day, might be assigned a trust rating of 8.

5.4.2 Some issues with the use of PKI for SEE

5.4.2.1 Developer attitudes to signing

Many code developers apparently hate the idea of having to acquire a key pair, a certificate and to sign all their code. The reasons for this are not clear but the expense, inconvenience and potential liability issues are probably involved somewhere.

There are a number of responses to this.

One response might be "*Get over it!*" and to say that certificates are not so expensive nor difficult to use. You might point to the fact that many ordinary people are already using certificates and key pairs, for instance in some countries, tax returns can be entered electronically using digital signatures for security.

Alternatively one could take a slightly more positive approach, for instance, to ask why we are requiring developers to use digital signatures and to also explore what they mean. This is the approach taken in the next sub-section.

5.4.2.2 Why sign at all? What is the meaning of a certificate?

It is therefore worth asking why we ask developers to sign their code.

The first reason is so that certain functions can only be given to authorised parties, and that the possession of a certificate from an appropriate CA is considered sufficient reason to be considered authorised by the device. A question that follows from this is whether or not there are other ways of giving the developer a designation as authorised that the device can understand without requiring signing. One such way would be for the developer to submit their code to an entity that didn't mind signing so much, for instance a code signing service.

Another follow on from the statement that we require signing is so that certain functions can only be given to authorised parties, and that the possession of a certificate from an appropriate CA is considered sufficient reason to be considered authorised, is to ask why we are restricting these functions to authorised parties only? It might be said that authorised parties can be trusted more, but in the case of public CAs, there is no vetting of the "trustworthiness" of the signer at all, the CA will only check their identity (more about that later). With regard to developers who have been examined by the entity issuing them a certificate, as is expected for developers, say, working in the operator or manufacturer domains of MIDP 2.0, see [27], we could say that their possession of a certificate does indicate trustworthiness, but not in the case of public CAs. In the case of public CAs, the use of a certificate for authorisation does give the *opportunity* of widespread, effective withdrawal of authorisation, by using certificate revocation methods. However, very few fixed clients have certificate revocation capability and in the wireless area, virtually no clients have revocation checking capability. We can therefore say that in the absence of revocation checking (and easy but effective measure for developers to be identified as deserving of withdrawal of authorisation), the use of certificates issued by public CAs as a method of authorisation amounts to an unfounded and irreversible designation of trustworthiness.

It might be said in defence of such unfounded and irreversible designations of trustworthiness that, even without revocation, there is at least the potential for the source of a signed virus to be identified and therefore prosecuted, and that the threat of this will be a sufficient deterrent to would be fraudsters in this area. As there have been prosecutions for the development of viruses, (for example, Kevin Mitnick[11]), we can assume that prosecution of developers of signed viruses would also be possible. However, there are a number of issues with this:

- Prosecutions are expensive, and unless (using terms from UK law) the offence of developing and propagating a signed virus is considered a *criminal* offence that the police will pursue, the prosecution will have to be funded by a private individual, say an operator. Such private prosecutions give no guarantees of success and are only undertaken in severe circumstances. A developer might produce a signed virus that cause some damage but not enough to warrant a private prosecution. Hence, damage has been done but the developer has not suffered and hence there is no deterrent to further damage.
- The rogue developer may have posted their private key on the internet, and can therefore claim that it cannot be proven that they signed the executable.
- Even if a developer can be prosecuted, they may have insufficient resources to pay any damages to the effected party. The prosecuting party may therefore have the satisfaction of seeing the offending party in jail or with a criminal record, but they will not get any damages, nor, probably, and refunding of their legal costs. Again, we can wonder if there is a real

[11] For further details see, for example, http://www.gulker.com/ra/hack/.

deterrent to the generation of signed viruses and therefore if the purported value of certificates in enabling the tracing of executable source and therefore the deterrence, is really there.

- If certificates are seen as a way to trace the source of an executable, we can ask if there are not other ways of tracing the source. As many executables will be downloaded from web sites, there is the web site address to point to as the source to start with. Where the site is just acting as an executable aggregator (e.g. www.midlet.org) the site will be likely to have logs of who posted particular midlets to the site so the source could still be found. Where executables are pushed to users, as could be done with MMS, even where the sender has not allowed the receiver to see the sending identity, the identity of the sender is known by the sending MMSCs, and so can be obtained in the case of virus distribution using MMS.

5.4.2.3 Per application certificates and signing services

One way to overcome the problem of developer certificates which cannot be revoked is to make a certificate usable for the signing of one particular executable only. This can be achieved if a hash of the certificate is included in the certificate. This will not absolutely prevent signed viruses, but it does mean that the obtaining of a certificate by a fraudulent developer does not give him and anyone else he chooses to distribute his private key to, the capability to sign any executable they like but only the capability to sign one particular executable. It might be thought that the fraudulent developer would just then request as many certificates as they have fraudulent applications they want to propagate, but the use of per executable certificates would give a sensible CA the option not to issue many certificates all in one go but to first see if the executables issued by the developer are safe or not.

A logical extension of per executable certificates is for the CA to actually sign the code itself (rather a hash of it) and not issue a certificate for the developer to sign the code. The developer would then enclose the signed, timestamped hash of his executable along with the executable and any intermediate certificates required and distribute this bundle to devices.

5.4.2.4 Overcoming the absence of root ubiquity

5.4.2.5 One problem with the use of PKI for SEE is that if a verifying device does not have the required root certificate, then the developer certificate cannot be verified and does not indicate anything about the trustworthiness of the developer but merely his ability to generate certificates.

5.4.2.6 One method to resolve this is to use the resource sharing concepts described in the WP2 component of this deliverable (D13), in particular those described as part of the PSD concept to enable transfer of root certificates within a PAN. Therefore if a mobile device does not have a particular root certificate, it could request the certificate from a laptop in the PAN, which, with its large memory and greater number of default roots installed in the browser, will have more chance of having the desired root.

5.5 Conclusions and recommendations for further research

This chapter began with a review of the SEE requirements defined in D03 [38], seeing which were not met, or not met well, by existing standards or by work of the Shaman project. Examining the requirements which were not met (well) we see that all of them are to do with **authorisation** issues. This confirms what has been said about PKI techniques for sometime, that it can be used well to provide commercially and organisationally neutral security services such as authentication, integrity protection and confidentiality but that it does not provide any easy solutions to problems of authorisation. In fact, compared to symmetric techniques, in many cases public key techniques are poorer than symmetric with regard to authorisation.

It was found that there are some “out of band” solutions to the authorisation issues with the use of public key techniques for SEE and we believe that Shaman’s approach of taking a look at the problem that was wider than the pure cryptographic/security engineering view has been fruitful here. However, we believe that standardisation bodies should not give up on standards-based solutions to these problems. At the time of writing, the 3GPP MExE group looks like it will be disbanded. However, though this document finds the MExE work imperfect, the MExE specifications provide a good basis for future work.

Having seen that authorisation provided issues for SEE, it was then described how SEE, or rather, the post-manufacture flexibility in terminal functionality that SEE gives rise to, presents issues for authorisation and authentication. More precisely, if authentication is being carried out to establish whether a device is a particular type or not and therefore whether or not the behaviour of the device can be relied upon or not, post-manufacture flexibility of terminal behaviour can mean the behaviour of the device perhaps cannot be relied upon, because it has been adjusted to give unacceptable behaviour (for example, to ignore the rights information put upon DRM protected content). To cope with this situation, this document recommends that device authentication methods provide at least the identity of the base OS or kernel of the device, and that this base OS be in charge of which aspects of the device behaviour can be changed. Further, it is then required that this base OS itself cannot be changed. Further recommendations also follow, to do with the future development of SEE. It is recommended that SEE designers take note of the capabilities that they give to SEEs, and when giving capabilities that could be misused with regard to the compliant (in some context) behaviour of the device, should only give those capabilities to authorised parties.

We then considered some alternatives to the use of PKI and code signing for achieving secure execution environments. It seems that there is promise in alternative methods of designating code sources as trusted and untrusted, such as methods similar to the ways that humans differentiate between trusted and untrusted people. People trust others that they have known for some time and have good experiences of, and analogues of duration of knowledge and quality of experience (e.g. number of times called, previous code received from another device) can be found for inter-device communication. Further research into this area is encouraged.

6 PKI for limited devices

6.1 Introduction

It is well known that public key operations are computationally expensive and therefore require considerable computing power in the device performing them. Furthermore, a public key solution requires a PKI, which imposes requirements on the processing power and storage capacity of the device, as well as requirements on the bandwidth of the wireless link.

Limited devices are devices with very restricted capabilities with regard to user interfaces, such as small screens and limited keyboard, low memory resources, limited battery capacity and limited computational capabilities.

It is expected that these limitations will exist for some of the components of future distributed mobile devices. In Section 2 of SHAMAN Deliverable D10 [42], a component is defined as an independent computing unit, which means that it must have processing capabilities as well as digital memory. Additionally, a component must have at least one local interface to connect directly with at least one other component. However, components do not necessarily have a user interface, global interface or the ability to host a security module.

This means that a variety of components with very different capabilities and in particular security capabilities may be expected. For some components performing public-key cryptography and the associated PKI operations on the component itself is impossible, although it would be desirable to secure global and PAN-local communications.

In addition to the limited capabilities of the mobile device, wireless networks also offer only restricted bandwidth.

These constraints influence the design of a PKI. Several enhancements to a PKI are possible to adapt PKI concepts to the wireless world, such as compact certificates, the use of certificate URLs, short-lived certificates and piggybacking OCSP messages. All these issues are described in D04 [39].

Another solution considered in the past for public key protocols is asymmetric load distribution. Protocols can be designed such that the public key cryptographic operations performed by the user device are much less demanding than the corresponding operations required by the network server. It is assumed that this is no longer a requirement for mobile devices, as the limitations of such devices diminish.

Despite all those measures, some components of a distributed device still do not have sufficient resources to perform the required public key or PKI operations. This resource problem can be solved in some cases by using distributed security. In D08 [41] a distributed security module model and the functional split between the mobile device and the smart card is described.

This chapter provides information for the selection of a public key algorithm most appropriate for limited devices (6.2), analyses the public key and PKI operations to be performed by limited devices (6.3), and identifies methods enabling distributed security (6.4).

6.2 Selecting PK algorithms for limited devices

One aspect of implementing PK techniques in limited devices is selecting algorithms most appropriate for such devices. This is the subject of this subsection. There are three main parts of this discussion.

- Firstly, criteria for the selection of algorithms for such devices are provided. This is achieved by considering which aspects of such algorithms are of relevance in such a selection. These criteria are all efficiency related, given that this is likely to be a critical issue for such limited devices.

- Secondly, the properties of a range of public key algorithms with respect to certain of these criteria are given.
- Thirdly, guidance on the use of the selection criteria is given. The analyses in the two previous subsections do not enable specific algorithms to be recommended, since the relevance of different criteria will depend on the details of the implementation environment. However, it is possible to provide general guidance applying to a wide range of environments.

Following these two main sections some concluding remarks are given.

6.2.1 Criteria for algorithm selection

There are three main categories of (efficiency-related) criteria for the selection of public key algorithms. These are computational complexity issues, communications and storage complexity issues, and randomness requirements. We now consider each of these categories in turn.

6.2.1.1 Computational complexity issues

There are four main performance issues of this type, all of which measure the time taken for a computer system[12] to perform particular operations. In each case this assessment must also include the degree of use of any additional functions (e.g. the number of uses of a hash-function during signature computation)

C1. *Performing one private key based operation*. This criterion measures the complexity involved in performing one iteration of a private key operation. For example, this would measure the complexity of computing a single digital signature or decrypting one block of data. Note that for authenticated key agreement it is not appropriate to separate uses of private and public keys, since both are used in the establishment of an authenticated shared secret with another party; hence, in this case, C1 and C2 must be combined.

C2. *Performing one public key based operation*. This criterion measures the complexity involved in performing one iteration of a public key operation. For example, this would measure the complexity of verifying a single digital signature or encrypting one block of data. Note that for authenticated key agreement it is not appropriate to separate uses of private and public keys, since both are used in the establishment of an authenticated shared secret with another party; hence, in this case, C1 and C2 must be combined.

C3. *Generating a single key pair*. This criterion measures the complexity involved in generating a single key pair for use with this algorithm.

C4. *Generating system parameters*[13]. Some (but not all) asymmetric cryptosystems require certain system parameters to be selected. These parameters must be known and agreed by all users of the system. This criterion measures the complexity involved in selecting a set of system parameters.

6.2.1.2 Communications and storage complexity issues

There are six main performance issues of this type, all of which measure the size (in bits) of particular data objects related to the use of public key cryptosystems.

S1. *Size of system parameters*. Some (but not all) asymmetric cryptosystems require certain system parameters to be selected. These parameters must be known and agreed by all users of

[12] Of course, this assessment will vary depending on the type of system being used. It is vital that, when used in making decisions, assessments of different algorithms are made using the same (or very similar) systems. In addition, as far as the relevance to 'limited devices' is concerned, wherever possible emphasis should be given to comparisons based on simpler rather than more complex systems.

[13] This parameter selection operation is unlikely to be performed very often, and will typically not be performed by a 'limited device'. Hence the complexity of this operation is unlikely to be very significant for the selection of public key algorithms for use in limited devices. However it is included here for completeness.

the system. This criterion measures the size (in bits) of these parameters. These parameters will typically not need to be transferred between devices, except where they are stored in public key certificates[14].

S2. *Size of public key.* This criterion measures the size (in bits) of the public key. In some cases, use of the public key requires use of one or more auxiliary functions, as well as the asymmetric cryptographic primitive itself. For example, use of a signature verification public key will typically require the identification of a hash-function as well as the asymmetric primitive. Whilst we can ignore the bits required for asymmetric primitive identification (since this will be the same for all algorithms), the number of auxiliary functions to be identified is relevant.

S3. *Size of private key.* This criterion measures the size (in bits) of the private key.

S4. *Size of signature overhead.* This criterion measures either the size (in bits) of a digital signature (in the case of signatures with appendix[15]) or the difference in bit length between a signature and the maximum size of message information that can be received from a signature (in the case of a signature with message recovery).

S5. *Size of public key encryption overhead.* This criterion measures the difference in bit length between an encrypted block of data and the corresponding plaintext. In most cases, asymmetric encryption is used only for the encryption of secret session keys, and hence typically only one block of data is subjected to the encryption process.

S6. *Size of exchanged key establishment parameters.* This criterion measures the size (in bits) of the parameters exchanged between two parties using an asymmetric key establishment mechanism. Typically, the same amount of data will be transferred in both directions.

6.2.1.3 Randomness requirements

There are three main performance issues of this type. In each case must quantify the nature and quantity of random input required.

R1. *Randomness input to private key operation.* This criterion measures the randomness requirements for the performance of one private key operation. For example, this would measure the randomness needed (if any) to compute a single digital signature or decrypt one block of data.

R2. *Randomness input for key pair generation.* This criterion measures the randomness requirements for the generation of a single key pair.

R3. *Randomness input for system parameter generation*[13]. Some (but not all) asymmetric cryptosystems require certain system parameters to be selected. These parameters must be known and agreed by all users of the system. This criterion measures the randomness requirements for the selection of a set of system parameters.

6.2.2 Information on specific algorithms

Most, if not all, publicly discussed public key algorithms have associated parameters determining the size of keys (and the degree of security offered). For example, for RSA one must choose the length of the modulus, for discrete logarithm techniques one must choose the size of the multiplicative group,

[14] Reliable knowledge of system parameters is essential for the correct interpretation of public keys. Hence, in some cases, they may be explicitly included in public key certificates. However, they may also be implicitly included in such certificates (e.g. through a policy identifier or through the identity of the CA signing the certificate).

[15] Digital signatures can be divided into two main types: signatures with message recovery, where all or part of the message can be recovered from the signature, and signatures with appendix, where the signature acts purely as a check value on the message being signed. Signatures of the first type may be preferable in circumstances where storage space and/or communications bandwidth are at a premium, since that part of the message which is recoverable from the signature does not need to be stored or sent with the signature.

and so on. This, when comparing algorithms, it is necessary to choose the key parameters so that the compared algorithms provide roughly equivalent security (as determined by the current state of the art).

Note that relevant information on this topic is also provided in Wiener's 1998 article, [46].

6.2.2.1 Computational complexity assessments

The following table (Table 6) was taken from the web site

`http://www.eskimo.com/~weidai/benchmarks.html`

on 22nd September 2002. The final (Milliseconds/operation) column gives assessments of criteria **C1**, **C2** and **C3** for a wide range of public key algorithms.

Table 6 – Computational complexity of public and private key operations

Operation	Iterations	Total Time	Milliseconds/Operation
RSA 512 Encryption	73671	10.004	0.14
RSA 512 Decryption	5188	10.005	1.93
Rabin 512 Encryption	10539	10.004	0.95
Rabin 512 Decryption	3624	10.004	2.76
Blum-Goldwasser 512 Encryption	23662	10.005	0.42
Blum-Goldwasser 512 Decryption	4124	10.004	2.43
LUC 512 Encryption	61951	10.004	0.16
LUC 512 Decryption	2593	10.005	3.86
ElGamal 512 Encryption	3820	10.004	2.62
ElGamal 512 Encryption with precomputation	5739	10.005	1.74
ElGamal 512 Decryption	7327	10.004	1.37
RSA 1024 Encryption	31009	10.004	0.32
RSA 1024 Decryption	978	10.005	10.23
Rabin 1024 Encryption	4440	10.004	2.25
Rabin 1024 Decryption	829	10.005	12.07
Blum-Goldwasser 1024 Encryption	10405	10.004	0.96
Blum-Goldwasser 1024 Decryption	898	10.004	11.14
LUC 1024 Encryption	24814	10.005	0.40
LUC 1024 Decryption	525	10.014	19.07
ElGamal 1024 Encryption	907	10.005	11.03
ElGamal 1024 Encryption with precomputation	2232	10.004	4.48
ElGamal 1024 Decryption	1734	10.004	5.77
LUCELG 512 Encryption	1807	10.005	5.54
LUCELG 512 Decryption	3166	10.004	3.16
RSA 2048 Encryption	11260	10.005	0.89

RSA 2048 Decryption	156	10.004	64.13
Rabin 2048 Encryption	1690	10.004	5.92
Rabin 2048 Decryption	140	10.025	71.61
Blum-Goldwasser 2048 Encryption	3564	10.004	2.81
Blum-Goldwasser 2048 Decryption	150	10.035	66.90
LUC 2048 Encryption	8725	10.004	1.15
LUC 2048 Decryption	88	10.045	114.15
ElGamal 2048 Encryption	204	10.034	49.19
ElGamal 2048 Encryption with precomputation	663	10.004	15.09
ElGamal 2048 Decryption	395	10.015	25.35
LUCELG 1024 Encryption	416	10.014	24.07
LUCELG 1024 Decryption	760	10.005	13.16
RSA 512 Signature	5215	10.004	1.92
RSA 512 Verification	79436	10.004	0.13
Rabin 512 Signature	3448	10.005	2.90
Rabin 512 Verification	11897	10.004	0.84
RW 512 Signature	3762	10.005	2.66
RW 512 Verification	175022	10.004	0.06
LUC 512 Signature	2532	10.004	3.95
LUC 512 Verification	67551	10.005	0.15
NR 512 Signature	7452	10.004	1.34
NR 512 Signature with precomputation	10945	10.005	0.91
NR 512 Verification	6570	10.004	1.52
NR 512 Verification with precomputation	6654	10.004	1.50
DSA 512 Signature	5639	10.005	1.77
DSA 512 Signature with precomputation	8421	10.004	1.19
DSA 512 Verification	4955	10.004	2.02
DSA 512 Verification with precomputation	5179	10.005	1.93
RSA 1024 Signature	972	10.004	10.29
RSA 1024 Verification	33170	10.005	0.30
Rabin 1024 Signature	816	10.004	12.26
Rabin 1024 Verification	4127	10.004	2.42
RW 1024 Signature	835	10.005	11.98
RW 1024 Verification	89647	10.004	0.11
LUC 1024 Signature	527	10.015	19.00
LUC 1024 Verification	26325	10.004	0.38

NR 1024 Signature	1772	10.004	5.65
NR 1024 Signature with precomputation	4379	10.004	2.28
NR 1024 Verification	1528	10.004	6.55
NR 1024 Verification with precomputation	2692	10.005	3.72
DSA 1024 Signature	1819	10.004	5.50
DSA 1024 Signature with precomputation	4403	10.004	2.27
DSA 1024 Verification	1567	10.005	6.38
DSA 1024 Verification with precomputation	2724	10.004	3.67
LUCELG 512 Signature	3587	10.005	2.79
LUCELG 512 Verification	1783	10.004	5.61
RSA 2048 Signature	156	10.004	64.13
RSA 2048 Verification	11782	10.004	0.85
Rabin 2048 Signature	143	10.064	70.38
Rabin 2048 Verification	1649	10.004	6.07
RW 2048 Signature	146	10.054	68.86
RW 2048 Verification	36582	10.005	0.27
LUC 2048 Signature	88	10.044	114.14
LUC 2048 Verification	9072	10.005	1.10
NR 2048 Signature	406	10.004	24.64
NR 2048 Signature with precomputation	1314	10.005	7.61
NR 2048 Verification	355	10.004	28.18
NR 2048 Verification with precomputation	789	10.005	12.68
LUCELG 1024 Signature	825	10.014	12.14
LUCELG 1024 Verification	409	10.024	24.51
XTR-DH 171 Key-Pair Generation	2242	10.005	4.46
XTR-DH 171 Key Agreement	1122	10.004	8.92
XTR-DH 342 Key-Pair Generation	610	10.005	16.40
XTR-DH 342 Key Agreement	306	10.054	32.86
DH 512 Key-Pair Generation	7760	10.005	1.29
DH 512 Key-Pair Generation with precomputation	11658	10.004	0.86
DH 512 Key Agreement	7494	10.004	1.33
DH 1024 Key-Pair Generation	1827	10.005	5.48
DH 1024 Key-Pair Generation with precomputation	4538	10.004	2.20
DH 1024 Key Agreement	1774	10.004	5.64

DH 2048 Key-Pair Generation	410	10.005	24.40
DH 2048 Key-Pair Generation with precomputation	1336	10.004	7.49
DH 2048 Key Agreement	404	10.015	24.79
LUCDIF 512 Key-Pair Generation	3659	10.004	2.73
LUCDIF 512 Key Agreement	2712	10.005	3.69
LUCDIF 1024 Key-Pair Generation	835	10.004	11.98
LUCDIF 1024 Key Agreement	700	10.014	14.31
MQV 512 Key-Pair Generation	7791	10.005	1.28
MQV 512 Key-Pair Generation with precomputation	11420	10.004	0.88
MQV 512 Key Agreement	4098	10.005	2.44
MQV 1024 Key-Pair Generation	1824	10.004	5.48
MQV 1024 Key-Pair Generation with precomputation	4540	10.004	2.20
MQV 1024 Key Agreement	982	10.005	10.19
MQV 2048 Key-Pair Generation	411	10.024	24.39
MQV 2048 Key-Pair Generation with precomputation	1348	10.005	7.42
MQV 2048 Key Agreement	228	10.034	44.01
ECIES over GF(p) 168 Encryption	701	10.004	14.27
ECIES over GF(p) 168 Encryption with precomputation	1407	10.004	7.11
ECIES over GF(p) 168 Decryption	389	10.005	25.72(*)
ECNR over GF(p) 168 Signature	1393	10.004	7.18
ECNR over GF(p) 168 Signature with precomputation	2795	10.005	3.58
ECNR over GF(p) 168 Verification	763	10.014	13.12
ECNR over GF(p) 168 Verification with precomputation	1704	10.004	5.87
ECDHC over GF(p) 168 Key-Pair Generation	1404	10.005	7.13
ECDHC over GF(p) 168 Key-Pair Generation with precomputation	2822	10.004	3.55
ECDHC over GF(p) 168 Key Agreement	1366	10.005	7.32
ECMQVC over GF(p) 168 Key-Pair Generation	1398	10.004	7.16
ECMQVC over GF(p) 168 Key-Pair Generation with precomputation	2818	10.004	3.55
ECMQVC over GF(p) 168 Key Agreement	736	10.025	13.62
ECIES over GF(2^n) 155 Encryption	466	10.004	21.47

ECIES over GF(2^n) 155 Encryption with precomputation	1394	10.005	7.18
ECIES over GF(2^n) 155 Decryption	740	10.004	13.52
ECNR over GF(2^n) 155 Signature	927	10.004	10.79
ECNR over GF(2^n) 155 Signature with precomputation	2731	10.005	3.66
ECNR over GF(2^n) 155 Verification	752	10.004	13.30
ECNR over GF(2^n) 155 Verification with precomputation	1605	10.005	6.23
ECDHC over GF(2^n) 155 Key-Pair Generation	932	10.004	10.73
ECDHC over GF(2^n) 155 Key-Pair Generation with precomputation	2789	10.004	3.59
ECDHC over GF(2^n) 155 Key Agreement	896	10.005	11.17
ECMQVC over GF(2^n) 155 Key-Pair Generation	931	10.004	10.75
ECMQVC over GF(2^n) 155 Key-Pair Generation with precomputation	2800	10.005	3.57
ECMQVC over GF(2^n) 155 Key Agreement	734	10.024	13.66

(*) This number is higher than it should be because of a performance bug in Crypto++ 4.0. It will be fixed in the next release.

Notes on Table 6

- RSA and LUC use 17 as the public exponent.
- DH and ElGamal encryption and decryption use short exponents to save time. The sizes of the secret exponents were chosen so that a meet-in-the-middle attack would be slower than the general discrete log algorithm (NFS). The sizes used were:

modulus	**exponent**
512	120
1024	164
2048	226

- Blum-Goldwasser is timed on a 16-byte plaintext. The lowest log(log(n)) bits of each square are used.
- EC means elliptic curve. Operations in GF(2^n) are implemented using a trinomial basis.
- All tests were done by repeating the crypto operations over small blocks of random data. In practice you will likely see slower speeds because time is needed to transfer data to and from memory.
- The RSA, RW, DH, MQV, and elliptic curve schemes come from the IEEE P1363 standard. For more information see http://grouper.ieee.org/groups/1363/index.html.
- Precomputation means using a table of 16 precomputed powers of each fixed base to speed up exponentiation.

6.2.2.2 Communications and storage complexity assessments

In this section we provide information regarding the storage and communications complexity of a selection of signature and public key encryption schemes. Specifically, Table 7 provides information regarding the storage complexity of signature schemes, and Table 8 provides information regarding the storage complexity of public key encryption schemes. In both tables (and in subsequent tables) all sizes are in bits.

Table 7 – Storage complexity for signature schemes

Scheme	**System parameter size (S1)**	**Public key size (S2)**	**Private key size (S3)**	**Signature overhead (S4)**
RSA (*n* bit modulus)	0	Up to 2*n*	2*n*	*n*
RSA with message recovery – e.g. ISO/IEC 9796-2 (*n* bit modulus)	0	Up to 2*n*	2*n*	Around 200 (for a 160-bit hash).
DSA (with up to *k* bit primes)	0	Up to 4*k*	*k*	160
Rabin (with an *n* bit modulus)	0	*n*	2*n*	*n*

Table 8 – Storage complexity for public-key encryption schemes

Scheme	**System parameter size (S1)**	**Public key size (S2)**	**Private key size (S3)**	**Encryption overhead (S5)**
RSA (with an *n* bit modulus)	0	Up to 2*n*	Up to *n*	Up to *n*
El-Gamal (with a *k* bit prime)	0	Up to 3*k*	Up to *k*	Up to *k*
EPOC-2 (with an *n* bit modulus)	0	Up to 3*n*	Up to 2*n*	–
ACE-KEM (over *GF(p)* where *p* is a *k* bit prime)	Up to 5*k*+1	Up to 8*k*	Up to 4*k*+4	–
ECIES-KEM (over *GF(p)* where *p* is a k bit prime)	Up to 5*k*+1	Up to 2*k*	Up to *k*+1	–

6.2.2.3 Randomness assessments

Most asymmetric cryptographic algorithms use random numbers in some form but the generation of suitably random bits can be quite troublesome. It is possible to write most algorithms deterministically with some kind of fixed length random seed being taking as input. The following tables (Table 9 and Table 10) summarise the sizes of random seed required to operate signature schemes and public-key encryption schemes.

Table 9 – Random bit requirements for signature schemes

Scheme	**Random seed size for key generation (R1)**	**Random seed size for signing (R2)**[16]
RSA (*n* bit modulus)	*n*	0
RSA with message recovery – e.g. ISO/IEC 9796-2 (*n* bit modulus)	*n*	0
DSA (with up to *k* bit primes)	Approximately 4*k*	*k*
Rabin (with an *n* bit modulus)	*n*	0

Table 10 – Random bit requirements for public-key encryption schemes

Scheme	**Random seed size for key generation (R1)**	**Random seed size for encryption (R3)**
RSA (with an *n* bit modulus)	*n*	0
El-Gamal (with a *k* bit prime)	2*k*	*k*
EPOC-2 (with an *n* bit modulus)	Up to 2*n*	*n*
ACE-KEM (over *GF(p)* where *p* is a *k* bit prime)	4*k*	*k*
ECIES-KEM (over *GF(p)* where *p* is a k bit prime)	*K*	*k*

6.2.3 Using these criteria

When selecting an algorithm, it is necessary to consider very carefully the environment in which it is being used. For example, if the algorithm is being used as part of a protocol, the precise computational and communications requirements arising from that protocol need to be considered, and then the algorithms assessed against these particular requirements. Further, the particular constraints on the devices will need to be carefully assessed, since in some environments computational complexity may be the most critical factor, whereas in others storage and/or bandwidth complexity will be the most constrained resource.

The information provided in Sections 6.2.1 and 6.2.2 above should provide the building blocks for such assessments,

6.2.4 Conclusions and recommendations

A set of criteria for assessing the suitability of a public key algorithm have been provided. Information in how a large number of public key algorithms match up to these criteria has also been given. This should enable informed choices regarding algorithm selection to be made.

6.3 Analyses of public key operations

6.3.1 Digital signature generation

The generation of a digital signature is performed in two steps: in the first step the data to be signed is hashed to a fixed-length value (the hash-code) by means of a cryptographic hash function, and in the second step a private-key operation is applied to this hash-code. To be able to guarantee a sufficient

[16] All sizes are in bits

security level, it is mandatory that the private key is stored on a Security Module (SM) and never leaves it. This means that the private key operation needs to be executed on the SM. The hash function computation may be performed on the SM or on the device itself. A detailed analysis of the function split between the device and the SM can be found in SHAMAN Deliverable D08 [41].

Due to the very different capabilities of the components of a distributed device it may be the case that the component that needs to compute a signature in order to meet the security requirements of some external entity lacks the ability to perform the signature computations. In this case the signature generation can be delegated to another component.

Another aspect of signature generation is the provision of the associated public key certificate, containing the binding of the public key part of the key-pair to the identity of the signer, to the communicating entity. The limited device can send its certificate to the communicating peer or can use a reference to a database where the certificate is stored and can be retrieved. It is expected that future wireless technology will not put a restriction on the sending of certificates over the wireless link. For interoperability the use of X.509v3 certificates is recommended.

6.3.2 Digital signature verification

To verify a signature the public-key of the signing identity, certified by a common trusted party, is needed. The public key certificate can be sent by the signed party along with the message, or it can be retrieved from a global repository. It is expected that future mobile networks will offer sufficient bandwidth to provide the certificate along with the signed data. After retrieval of the certificate, the certificate needs to be validated.

If the certificate is valid, a signature verification algorithm is executed. This consists of hashing the signed data, applying the public-key based signature verification operation to the signature, and comparing the output of the verification operation with the calculated hash. The signature is accepted if the two values match, and the verification calculations do not fail for any other reason. As these operations are computational expensive, the signature verification algorithm may be delegated to another device within the PAN.

6.3.3 Public key encryption

To encrypt data to be sent to a communicating peer, the public key certificate of the communicating peer has to be retrieved from a global repository, based on its identity. Although retrieval of certificates from a global repository is memory and bandwidth expensive, it is expected that future mobile networks offer sufficient bandwidth to provide this function. After retrieval of the certificate, the certificate needs to be validated.

If the certificate is valid a public-key encryption algorithm is executed, which is computationally expensive. Therefore, the public-key encryption algorithm may be delegated to another device within the PAN. Delegation of the encryption algorithm implies that the data is revealed to the component performing the encryption. This may be allowed in a PAN, where both components belong to the same user.

6.3.4 Public key decryption

To decrypt the data received from a communicating peer, a public key decryption algorithm is executed, which is computationally expensive. As for public key encryption, a component may do the decryption on behalf of another component within a PAN, for example, when the component is not able to host a SM that contains the private key for decryption. The user should be aware that the data is revealed to the component to which the decryption is delegated.

6.3.5 Asymmetric key establishment

The precise set of steps for this operation varies widely depending on the nature of the key establishment protocol used, but the steps common to a significant number of such protocols and which are resource demanding, include the exchange of public key certificates for the respective public key agreement keys and the validation of the exchanged certificates. Additional steps concern the preparation and exchange of key establishment messages and the computation of the shared key.

6.4 Analyses of PKI operations

6.4.1 Certificate validation

Certificate validation is required by signature verification and public key encryption. Certificate validation comprises the following major steps:

(1) Check the certificate integrity and the validity period stated in the certificate.

(2) Path acquisition and validation, i.e. establishing and verifying the chain of trust.

Checking the certificate validity period requires a reliable, resilient time source at the verifying party. The validity period normally can be checked locally from the certificate data.

The complexity of path acquisition and validation depends strongly on the PKI topology: in hierarchically structured PKI topologies it is comparatively simple, but it can get quite complex in meshed or mixed PKI topologies.

Therefore, a hierarchically structured PKI topology is preferable, but probably not feasible, because this implies an independent trusted third (regularity) party to build a complete hierarchical trust model. Providing the limited device with a list of trusted roots is also not feasible, as a mobile device cannot store the roots for all possible access networks.

For subscription-based access the access network can store a certificate for each home network it has a roaming agreement with. The MN passes its home ID to the visited network, which sends the corresponding certificate to the MN, showing that it has a roaming agreement with the MN's home network (the home network has signed the certificate, and confirmed that the identity of the home network is bound with the public key). Thus the MN only has to contain the root of his home network to validate the certificate. This home root key can be securely installed on the mobile phone during the registration process. However, this is not a solution in case there is no home network or the home network is inaccessible.

Therefore, path construction and validation can be very resource demanding. If the very limited device is not capable of performing the validation, it can make use of another component in the PAN that acts as a "certificate validation" server.

6.4.2 Certificate revocation check

In a global communication environment the certificate revocation status check requires a repository of CRLs or an online certificate status check using a protocol such as OCSP.

A CRL is a list identifying revoked certificates, which is signed by a CA and made available in a public repository. Each revoked certificate is identified by its certificate serial number. Certificates that have expired are removed from the list. The certificate extension field 'CRL Distribution Points' identifies how CRL information can be obtained.

Delta CRL lists can be used to improve processing time for applications and reduce the size of each newly produced CRL. This allows changes to be added to the local database while ignoring unchanged information that is already in the database.

Online revocation checking may significantly reduce the latency between a revocation report and the distribution of the information to the relying parties. However, these methods impose new security requirements; the certificate validator needs to trust the on-line validation service, whereas the repository does not need to be trusted (because the revocation list is signed by a trusted CA).

For global PKIs, the use of OCSP is recommended – see chapter 3. 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 regular updates of CRLs, and infrequent CRL update considerably reduces the effectiveness of CRL use.

For local PKIs, the use of CRLs can be effective, because there are far less bandwidth constraints within the PAN.

6.5 Methods enabling distributed security

As the PAN components have different and sometimes very restricted capabilities, an obvious solution for solving the resource problem is to delegate operations to other components in the PAN. Another possibility is that the PKI client operations are distributed over the PAN components. One can imagine that the SM is located only in the main component of the PAN. Functions to be executed by the mobile equipment may be performed on a component that makes use of the SM located on another component. For the user of the distributed terminal the location of the SM should be transparent. The exchange of information with regard to PKI client operations shall be done in a secure environment and therefore the components shall belong to the same PAN Security Domain (PSD). A PSD can contain components of different users / owners. The configuration of a PSD is described in Chapter 6 of D10 [42].

6.5.1 Server aided signature generation

The Server Aided Signature Generation (SASC) protocol, described in [13] enables a client to execute a RSA signature algorithm with the aid of a server without revealing its private key. The protocol is designed to enable a smart card or devices with limited capabilities to perform private key RSA computations faster with the aid of a server. The SASC protocol has already been presented in section 3.4 of D08 [41] as a method for distributing RSA operations between a tamper resistant device (TRD) and an insecure device. The devices may be physically co-located.

6.5.2 Delegated certificate processing

Components within the PAN that do not have sufficient resources to process certificates can delegate the processing to a server. Protocols that allow a component to outsource all or some of certificate validation, certificate revocation and certificate path construction to a server, are specified in [16] and [18].

6.6 Trust implications for limited devices

If the operations are outsourced to a server, the question is raised of how the server can be authenticated. This is necessary because of the critical operations the server has to perform. Because it is those devices that are not able to perform public operations that will use the services offered by the server, the authentication of the server has to be done by means of symmetric cryptographic methods. Components with restricted capabilities have therefore to be imprinted for later use of a symmetric cryptographic system, as explained in section 4.2 of D10 [42]. The master component can distribute the common PAN key, separately encrypted for each component, including the server. It is assumed that the master component has sufficient computing power to establish a shared key for each of the PAN components, and can derive and distribute the common PAN key. From this point on all PAN components can communicate securely with each other, including the PAN server component. One device within a PAN acts as the PK server, responsible for performing all PK and PKI computations for all other devices within the PAN. As an example the server could be a PDA computing a signature for another mobile device within the same PAN (and worn by the same individual).

6.7 Conclusions

In the near term mobile devices and smart cards will become more powerful, and most of them will be able to perform public key and PKI operations, providing the PKI and the certificates are optimised for a mobile environment. However, within a PAN some components may not be able to perform public key or PKI operations. We have proposed methods to delegate these operations to more powerful devices in the PAN.

The proposed methods need further investigation with respect to their performance. Their pros and cons have to be evaluated. Another issue for further consideration is whether it is feasible to have the server located outside the PAN.

7 Supporting security services with a PKI

7.1 Authentication

The issue of authentication has already been discussed in a general way in D04 [39]. 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 [39] have shown the general need for authentication and the different scenarios this feature is needed for.

This part of D13 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.

7.1.1 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.

7.1.1.1 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 will be granted on the basis of ad hoc 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 ad hoc payment could 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 annex 1 of D13.

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.

7.1.2 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 unclear how the use of certificates will develop in the near future. As the technical restrictions on mobile devices disappear, the protocols and formats will most likely converge to the established internet-standards. One example of this development is the emergence of WAP 2.0.

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 [39], chapter 7.3).

To underline this, Table 11 provides 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.

Table 11 – Technical restrictions on client/server authentication

We now look at the issues mentioned in the table in more detail:

7.1.2.1.1 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 depends 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.

7.1.2.1.2 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 handling 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.

7.1.2.1.3 Verification and revocation handling

The situation concerning this issue leads to the same conclusions as the previous topics. 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 3.

7.1.2.1.4 <u>Root key issues</u>

Root-keys as anchors to verify subordinately issued 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.

7.1.2.1.5 <u>Interoperability issues</u>

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:

7.1.2.1.6 <u>Certificate formats:</u>

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. However, 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 in which 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 fairly open standard, and time has shown that there are many degrees of freedom for defining an X.509 certificate.

It is not just the vaunted certificate extensions, as introduced in the third version of X.509, that are a potential threat to interoperability; other examples include 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 few parameters as possible, so as 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.

- The introduction of private extensions which are set critical is not recommended.

7.1.2.1.7 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.

7.2 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 [37].

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.

7.2.1 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.

7.2.1.1 Use of Access Control Lists (ACLs)

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 provider’s side can either be stored locally or on a central server. The solution to be chosen 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 preferable 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 central storage one has to consider this database as a bottleneck and also a single point of failure.

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.

7.2.1.2 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.

7.2.2 Certificate based access control

This section 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 their ease of implementation.

7.2.2.1 Use of virtual CAs for authorisation

A pragmatic approach to implementing a solution for authorisation with standard methods is the following.

Suppose you have an 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.

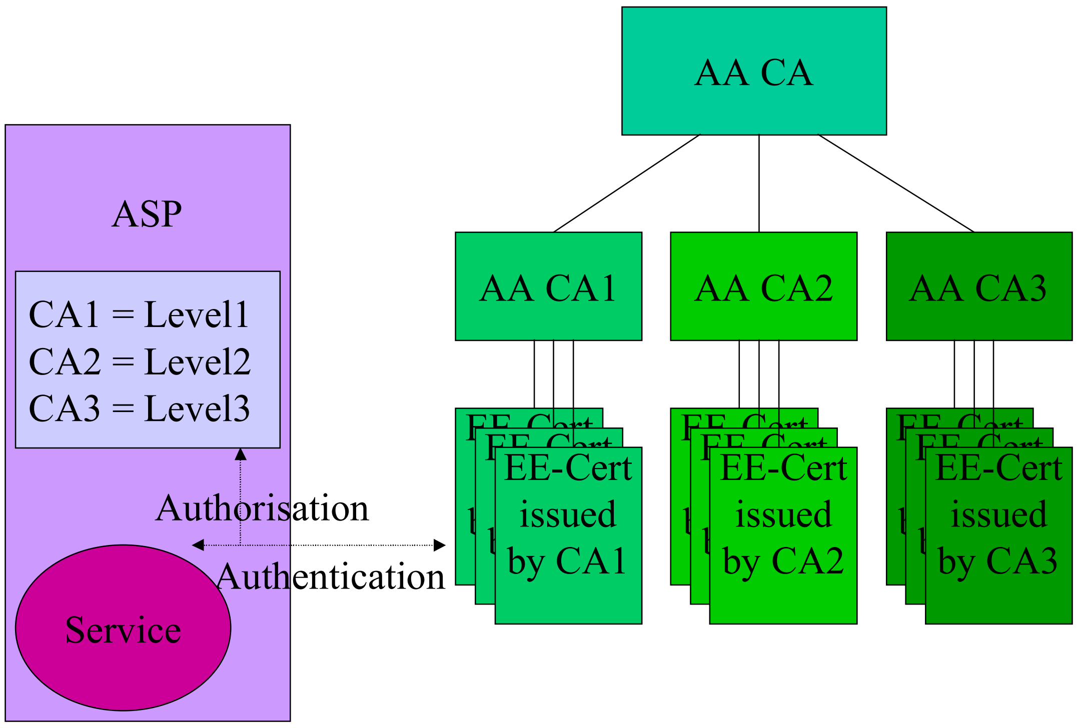

Figure 3 – Use of authorisation-related extensions in Public Key Certificates

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, as 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 [39], an extension is a triple consisting of the extension-identifier, the extension-value and the criticality-bit.

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

Another important point to consider is how to ensure that only accepted CAs will issue certificates with the authority-related extensions. To ensure the proper granting of rights, 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 do not 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.

7.2.2.2 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 mechanism has both 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.

7.2.2.3 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 an identity, the intention of the developer's of SPKI was to bind a public key to certain attributes. The concept is to identify a key-holder 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 also offers the binding of a key to a 'local' name.

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.

One 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.

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.

8 Conclusions and future research areas

The concluding discussions are divided up according to the main chapters of this document.

8.1 Public key based network access

In chapter 2, network access based on public key techniques was discussed. First, the subscription case where the user has a long-term trust relationship with a home network was described and analysed. Second, the alternative access case where there is no home network available was described. Instead, the user pays ad hoc for the services he/she requires. In the latter case, public key techniques are used to secure the (partly) wireless link between MN and AR.

The main goal of this work was to identify the requirements for public key mechanisms for network access, first for the subscription and second for the alternative access case (see sections 2.1.1 and 2.1.2 respectively). Two public key authentication and key agreement protocols were then reviewed, namely

- JFK, and
- IKEv2

The protocols were evaluated with respect to the requirements identified previously. The assessment for the subscription case yielded that JFKi fulfils all requirements for network access. However, JFKr and IKEv2 do not protect the user from active attacks from bogus access networks and therefore cannot be recommended for the network access scenario.

In the alternative access case, the set of requirements differs because the MN may use preliminary authentication or even remain anonymous initially. This implies that “authentication of the MN towards the access network” and “MN confidentiality” must be considered differently to the subscription case. However, the MN requires the access network to authenticate itself towards the MN. The assessment of the public key protocols for the alternative access case yielded that JFKi, JFKr, and IKEv2 are all suitable for the network access scenario.

Future work in public key based network access involves following the work of the IETF regarding a son-of-IKE protocol and possibly highlighting the requirements for our application, namely public key based network access, to the IETF. Moreover, other public key authentication protocols could be investigated.

Also, the role of the MN in the alternative access scenario needs to be further refined. Two cases, namely preliminary authentication and no authentication, have been identified and need to be investigated further. For example, what are the consequences of no authentication on the public key authentication protocol? Is this approach acceptable to network operators considering that it may open the door for DoS attacks on the access network?

8.2 Public key revocation in a mobile environment

In D04 [39] 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 were considered. These four techniques/protocols were described but not compared to any great extent. In chapter 3 the two most promising of these four protocols, namely OCSP and XKMS, were subjected to further analysis, compared and recommendations derived as to which should be used for the applications under consideration within SHAMAN.

SCVP was excluded 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 [39] 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.

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 be 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 [13] 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.

8.3 The personal PKI

The investigation of Personal PKI issues in chapter 4 showed that there are differences in the requirements and potentials for new mechanisms to be implemented compared to a more globally scaled PKI. Especially the issue of revocation in the PAN offers the possibility to look at new approaches. Furthermore it became clear that these new approaches need to be discussed and respective standards have to be established.

The following results have been established.

- Personal CAs: The most interesting aspect of CAs regarded in the context of Personal PKIs is the concept of multiple CAs. That means, due to the restricted availability and the danger of lost or stolen devices, mechanisms have to be specified that enable CAs to hand over possibility and to synchronise after a period of absence from the PAN. On the client side this concept has also to be introduced as a hand-over of CA-duties is not common in traditional PKIs.
- Device Initialisation: Protocols for device initialisation on very limited devices can be specified. One protocol for devices without keypad but with a display was derived from the protocol shown in 4.4.1. Another approach for devices without display and keypad was introduced, where new credentials have to be sent to the CA after using the credentials available after shipment for a first initialisation.
- Proof of possession: The discussion on proof of possession has shown, that the requirements and the methods known from traditional PKI in fixed networks can mostly be transferred to the PAN scenario.

- Revocation in PANs: Due to the nature of PANs, the considerably small number of participants and the fact, that no ‘offline-mode’ is necessary, new mechanisms for revocation can be introduced. These mechanisms are not yet standardised and may therefore be of interest for standardisation groups.
- ID-based cryptographic systems: In ID-based cryptographic systems the public key is not distributed in certificates but derived from a unique User-ID. Therefore certificates don’t have to be distributed. This results in less communication overhead. The computational complexity of both approaches is about the same, as mostly the same operations are performed. In contrast to traditional PKI the ID-based solution needs a much higher trust into a third party, as this party knows all the private keys.

8.4 PKI for secure execution environments

Chapter 5 began with a review of the SEE requirements defined in D03 [38], in particular by investigating which were not met, or not met well, by existing standards or by work of the Shaman project. Examining the requirements which were not met (well) we see that all of them are to do with **authorisation** issues. This confirms what has been said about PKI techniques for some time, namely that they can be used well to provide commercially and organisationally neutral security services such as authentication, integrity protection and confidentiality but that they do not provide any easy solutions to problems of authorisation. In fact, compared to symmetric techniques, in many cases public key techniques are poorer than symmetric with regard to authorisation.

It was found that there are some “out of band” solutions to the authorisation issues with the use of public key techniques for SEE and we believe that Shaman’s approach of taking a look at the problem that was wider than the pure cryptographic/security engineering view has been fruitful here. However, we believe that standardisation bodies should not give up on standards-based solutions to these problems. At the time of writing, the 3GPP MExE group looks like it will be disbanded. However, though this document finds the MExE work imperfect, the MExE specifications provide a good basis for future work.

Having seen that authorisation provided issues for SEE, it was then described how SEE, or rather, the post-manufacture flexibility in terminal functionality that SEE gives rise to, presents issues for authorisation and authentication. More precisely, if authentication is being carried out to establish whether a device is a particular type or not and therefore whether or not the behaviour of the device can be relied upon or not, post-manufacture flexibility of terminal behaviour can mean the behaviour of the device perhaps cannot be relied upon, because it has been adjusted to give unacceptable behaviour (for example, to ignore the rights information put upon DRM protected content). To cope with this situation, this document recommends that device authentication methods provide at least the identity of the base OS or kernel of the device, and that this base OS be in charge of which aspects of the device behaviour can be changed. Further, it is then required that this base OS itself cannot be changed. Further recommendations also follow, to do with the future development of SEE. It is recommended that SEE designers take note of the capabilities that they give to SEEs and when giving capabilities that could be misused with regard to the compliant (in some context) behaviour of the device, should only give those capabilities to authorised parties.

We then considered some alternatives to the use of PKI and code signing for achieving secure execution environments. It seems that there is promise in alternative methods of designating code sources as trusted and untrusted, such as methods similar to the ways that humans differentiate between trusted and untrusted people. People trust others that they have known for some time and have good experiences of, and analogues of duration of knowledge and quality of experience (e.g. number of times called, previous code received from another device) can be found for inter-device communication. Further research into this area is encouraged.

8.5 PKI for limited devices

In the near term mobile devices and smart cards will become more powerful, and most of them will be able to perform public key and PKI operations, providing the PKI and the certificates are optimised for a mobile environment. However, within a PAN some components may not be able to perform public key or PKI operations. We have proposed methods to delegate these operations to more powerful devices in the PAN.

The proposed methods need further investigation with respect to their performance. Their pros and cons have to be evaluated. Another issue for further consideration is whether it is feasible to have the server located outside the PAN.

References

[1] 3GPP TS23.057, "Mobile Station Application Execution Environment (MExE)".

[2] W. Aiello et al., "Just Fast Keying (JFK)". IETF Draft (work in progress), draft-ietf-ipsec-jfk-03.txt, April 2002.

[3] D. Boneh and M. K. Franklin, 'Identity-Based Encryption from the Weil Pairing'. In: J. Kilian (ed.): *Advances in Cryptology – CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pp. 213-229. Lecture Notes in Computer Science **2139**, Springer, 2001.

[4] M. Burnside, D. E. Clarke, T. Mills, A. Maywah, S. Devadas, and R. L. Rivest, 'Proxy-based security protocols in networked mobile devices'. In: *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC), March 10-14, 2002, Madrid, Spain*, pp. 265-272. ACM, 2002. [Available at http://theory.lcs.mit.edu/~rivest/BurnsideClarkeMillsMaywahDevadasRivest-ProxyBasedSecurityProtocolsInNetworkedMobileDevices.pdf].

[5] M. Burnside, D. E. Clarke, T. Mills, A. Maywah, S. Devadas, and R. L. Rivest, *Proxy-Based Security Protocols in Networked Mobile Resources*.
[Available at http://csg.lcs.mit.edu/pubs/reports/search4.pdf].

[6] 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.

[7] W. Diffie, P. C. van Oorschot and M. J. Wiener, 'Authentication and authenticated key exchanges', *Designs, Codes and Cryptography* **2** (1992) 107-125.

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

[9] 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.

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

[11] C. Gehrmann, K. Nyberg and C.J. Mitchell, 'The personal CA – PKI for a Personal Area Network', in: *Proceedings – IST Mobile & Wireless Communications Summit 2002, Thessaloniki, Greece, June 2002*, pp.31-35.

[12] D. Harkins, C. Kaufman, et al., 'Proposal for the IKEv2 protocol'. IETF Draft (work in progress), draft-ietf-ipsec-ikev2-02.txt, April 2002.

[13] S-M. Hong, J.-B. Shin, H. Lee-Kwang and H. Yoon, 'A new Approach to Server-Aided Secret Computation'. In: *The 1st International Conference on Information Security and Cryptology, ICISC '98, December 18-19, 1998, Seoul, Korea, Proceedings*. Korea Institute of Information Security and Cryptology (KIISC), 1998.
[Available at http://citeseer.nj.nec.com/hong98new.html].

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

[15] G. Horn, K. M. Martin and C. J. Mitchell, 'Authentication protocols for mobile network environment value-added services', *IEEE Transactions on Vehicular Technology*, **51** (2002) 383-392.

[16] Internet draft (June 2002), *Certificate Validation Protocol.*
[Available at http://www.ietf.org/internet-drafts/draft-ietf-pkix-cvp-00.txt].

[17] Internet X.509 Public Key Infrastructure, *Online Certificate Status Protocol – OCSP.* [Available at http://www.ietf.org/proceedings/02mar/I-D/draft-ietf-pkix-rfc2560bis-01.txt].

[18] Internet draft (June 2002), *Simple Certificate Validation Protocol.* [Available at http://www.ietf.org/internet-drafts/draft-ietf-pkix-scvp-09.txt].

[19] ISO/IEC 11770-3: 1999, *Information technology – Security techniques – Key management – Part 3: Mechanisms using asymmetric techniques*.

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

[21] ISO/IEC 15945, *Information technology – Security techniques – Specification of TTP services to support the application of digital signatures*. ISO/IEC, 2002.

[22] JSR 82. [Available at http://www.jcp.org/jsr/detail/82.jsp].

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

[24] A. J. Menezes, M. Qu and S. A. Vanstone, 'Some new key agreement protocols providing implicit authentication'. In: *Workshop record of the 2nd Workshop on Selected Areas in Cryptography (SAC '95)*, Ottawa, Canada, May 1995.

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

[26] MIDP 2.0. [Available at http://www.jcp.org/jsr/detail/118.jsp].

[27] RP for MIDP 2.0. [Available at http://www.jcp.org/jsr/detail/118.jsp].

[28] C. J. Mitchell and M. Walker, 'Solutions to the multidestination secure electronic mail problem', *Computers and Security* **7** (1988) 483-488.

[29] C. J. Mitchell, 'Multi-destination secure electronic mail', *The Computer Journal* **32** (1989) 13-15.

[30] M. Myers, R. Ankney, A. Malpani, S. Galperin and C. Adams, *Online Certificate Status Protocol – OCSP*, Internet RFC 2560, June 1999. [Available at www.ietf.org/rfc.html].

[31] G. Necula and P. Lee, 'Safe Kernel Extensions Without Run-Time Checking'. In: Proceedings of the 2nd Symposium on Operating System Design and Implementation (OSDI '96), Seattle, October, 1996, 229-243.
[Available at: http://www-2.cs.cmu.edu/~petel/papers/pcc/pcc-osdi96.ps].

[32] OMA DRM. [Available at http://www.openmobilealliance.org/documents.html].

[33] RFC2510 (March 1999). *Internet X.509 Public Key Infrastructure: Certificate Management Protocols*.
[Available at: http://www.ietf.org/rfc/rfc2510.txt?number=2510].

[34] RFC3029 (February 2001). *Internet X.509 Public Key Infrastructure: Data Validation and Certification Server Protocols*.
[Available at: http://www.ietf.org/rfc/rfc3029.txt?number=3029].

[35] RFC3280 (April 2002). *Internet X.509 Public Key Infrastructure: Certificate and Certificate Revocation List (CRL) Profile*.
[Available at: http://www.ietf.org/rfc/rfc3280.txt?number=3280].

[36] RFC3379 (September 2002). *Delegated Path Validation and Delegated Path Discovery Protocol Requirements*.
[Available at: http://www.ietf.org/rfc/rfc3379.txt?number=3379].

[37] SHAMAN Deliverable D02, *Intermediate Report: Results of Review, Requirements and Reference Architecture*. 29th June 2001. [Available at http://www.ist-shaman.org].

[38] SHAMAN Deliverable D03, *Intermediate Report: Security Architecture for Future Terminals and Applications.* 29th June, 2001. [Available at http://www.ist-shaman.org].

[39] SHAMAN Deliverable D04, *Initial report on PKI requirements for heterogeneous roaming and distributed terminals.* 10th September 2001. [Available at http://www.ist-shaman.org].

[40] SHAMAN Deliverable D07, *Intermediate specification of PKI for heterogeneous roaming and distributed terminals*. [Available at http://www.ist-shaman.org].

[41] SHAMAN Deliverable D08, *Intermediate specification of Security Modules*. [Available at http://www.ist-shaman.org].

[42] SHAMAN Deliverable D10, *Detailed specification of distributed mobile terminal system security*. [Available at http://www.ist-shaman.org].

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

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

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

[46] M. J. Wiener, 'Performance comparison of public-key cryptosystems', *CryptoBytes* **4 no 1** (Summer 1998), pp. 1-5.
[Available at `http://www.rsasecurity.com/rsalabs/cryptobytes/`].

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

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