The personal PKI*

Chris J. Mitchell (Royal Holloway, University of London, UK) and Ralf Schaffelhofer (T-Systems, ITC Security, Germany)

Abstract

The term personal PKI was devised within the SHAMAN project to describe a public key infrastructure specifically designed to support the distribution of public keys in a Personal Area Network. In this paper a variety of issues relating to the operation and management of a personal PKI are discussed. Following a general discussion of requirements for personal PKIs, the main topics covered are: the operation of personal CAs, device initialisation, proof of possession, and revocation.

1. Introduction

This paper is concerned with methods for the deployment of Public Key Infrastructure (PKI) techniques to support secure communications between devices in a Personal Area Network (PAN). One major issue dealt with in this paper is the development of methods for two PAN components to securely exchange their public keys, as required to support the pairing of "second party components" (as defined in the PAN reference model within Annex 2 of [8]). 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.

The term 'Personal PKI' is used throughout for a PKI deployed to support communications in a PAN. The idea is that by deploying a PKI in such a limited environment, many of the problems associated with PKI deployment in a much larger and less well-defined environment can be avoided, whilst the advantages of use of a PKI can be retained. The PAN is assumed to contain at least one device acting as a 'Personal Certification Authority (Personal CA)', which is responsible for generating public key certificates for all devices within the PAN.

Sections 2 and 3 of this paper contain a discussion of requirements and issues. Section 4 provides a detailed discussion of the Personal CA and the corresponding issues. In section 5 a protocol for device initialisation is introduced and analysed. Proof of possession as a means of assuring the certifier of the possession of the private key related to the public key to be certified is discussed in section 6, and section 7 lists requirements and ideas about revocation in PAN-environments. Finally, conclusions and issues for further research are provided in section 8.

2. 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 below.

* The work described in this paper has been supported by the European Commission through the IST Programme under Contract IST-2000-25350. The information in this document is provided as is, and no guarantee or warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

- *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 certificate may be issued 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.

2.1 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 has generated 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 simplified 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, i.e. details of the device and the public key, to the user for a decision regarding whether or not the certificate should be renewed. 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.

2.2 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.

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.

Another approach is *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. A more detailed discussion of theses issues can be found in section 7.

2.3 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 5.1.1 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 5.1.1 above will need to be used.

3. 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 Annex 2 of [8]):

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:

e. no third party passive interceptor of communications can learn any secret information;

f. 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);

g. 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 Annex 2 of [8], or a method of equivalent strength, should be used.

Additional and optional functional requirements are:

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

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

4. Personal CAs

4.1 Operation of a personal CA

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

4.1.1 CA initialisation

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

The requirements for the personal CA functionality listed in Section 3 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/USIM device.

4.1.2 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).

4.1.3 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 [8].

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 5.1 below.

4.1.4 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? Solutions devised for conventional PKI-scenarios, e.g. OCSP, may not be appropriate within the PAN environment. Therefore a discussion on revocation in PANs is provided (see section 7).

4.2 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:

1. There are two or more devices capable of acting as CAs. One is nominated as the Primary CA, and the others are Secondary CAs.
2. 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.

4.2.1 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 first be put into the state that existed prior to any initialisation. Thus, either a new primary CA has to be set up and used to imprint every device in the PAN (just as happened originally), or 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). In the latter case the list will obviously not contain the corrupted CA, and a new CA is nominated as primary, and the devices are initialised one by one with the new primary CA.

4.2.2 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.

4.2.3 Synchronisation issues

Finally 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.

5. Device initialisation

5.1 A protocol for device initialisation

The security requirements for the device initialisation process have been listed in Section 3 above. A protocol has been proposed by Gehrmann and Nyberg to meet the identified requirements – note also that this protocol has been previously described in annex 2 to [8]. We sketch this protocol below. The results in [8] provide in full detail examples of 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 5.2.

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

5.1.1 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.

5.1.2 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 5.1.1, 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 5.1.4 below).

5.1.3 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 5.1.1 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 given in section 6.

5.1.4 Analysis of protocol

The purpose of the protocol described in Section 5.1.1 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}.

5.2 Initialisation methods for limited devices

In section 5.1 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.

5.2.1 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 5.1 is used. 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 $\mathrm{MAC}_K(X_A, Y_B)$ 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 $\mathrm{MAC}_K(Y_A, X_B)$, 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 $M = \mathrm{MAC}_K(Y_A, X_B)$, 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.

5.2.2 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 5.2.1 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 if 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 or compromises the CA, preventing the secure re-initialisation of device *A*.

6. Proof of possession

Proof of possession is required to demonstrate the knowledge of the private key corresponding to 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 need to analyse whether the 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 in a PAN 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 **[3]**.

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

- An attacker may request a certificate for a public key of another person, at the same time spoofing the other person's identity.
- 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.

6.1 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[†], taken from Note 12.54 (pages 518/519) of the Handbook of Applied Cryptography [5]. 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 using

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

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 objective, this time 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*.

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 ([5], page 537) also points out that active attacks related to the above attack are considered by Diffie, van Oorschot, and Wiener [1], and Menezes, Qu, and Vanstone [6].

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, [9]. 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.

6.2 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 whether the user is able 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 and can be linked to the requesting device, proof of possession might not be necessary.
- The asymmetric key-pair is generated by the certifying party when the certificate is requested. Proof of possession is not 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 supported.

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. Use of the wrong kind of PoP technique 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 various PoP techniques, as discussed in 6.4 below, must therefore be mapped to the kind of key which 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 the nature of the algorithm specified to be used with a certain key makes a difference as to how proof of possession should be achieved. One possible criterion might be whether or not 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 depending on the PoP mechanism in use. Whereas in some case the informations may be non-critical, there may be mechanisms where the proving party is required 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 must therefore be further investigated with respect to zero-knowledge properties.

6.3 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.

As already mentioned, it is essential to consider where a key is generated. When key-generation is performed by the certifying party, and the private key is sent to the client together with the certificate, a PoP mechanism is certainly not necessary. If a key is generated by the manufacturer and pre-installed on a device before shipping, proof of possession may be necessary if attacker could get hold of the public key before the device reaches its owner. Proof of possession is not necessary if the authenticity of the key sent to be certified is secured with another mechanism. One example of such a mechanism could be as follows.

A manufacturer of smartcards pre-installs cryptographic keys on his cards. In addition, a non-personal certificate is generated and included on tbe card simply to prove that the public key sent for certification was generated by the manufacturing party. During the personal PKI certification 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 a request, and will get suspicious if he does not receive a proper certificate from the certifying party. That means that the only risk left is a man-in-the-middle attack, that is likely to be discovered very quickly 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 key pair 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 key pair is not intended to be used by the customer for purposes other than proving the possession of a key or the possession of the device.

Scenarios in which 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, even in this case there may be scenarios where PoP is unnecessary. An example of the latter 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.

6.4 Proof of possession mechanisms

We now consider a number of different mechanisms for establishing PoP. All these mechanisms are in some way 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 here 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 here 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 used with care, as the user is giving away information by decrypting a value selected by the CA. This means that, in standard cryptograohic terminology, the user is acting as an oracle. As the proof of possession process is only performed once, i.e. 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, if an attacker wants to use the user to decrypt an arbitrary challenge, the decrypted value will with very high probability not match 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 input to 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 output from the one-way-function, and this will almost certainly not be useful to 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 sending any information related to the private key but yet prove possession of the private key to the requesting party, a zero-knowledge protocol may be used. Since no information about the secret (i.e. the decrypted value) is given away to the requesting party, a potential attacker gets 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 with respect to 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 only be used in 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 will be able to retrieve the certificate from the directory. If the certificate is only distributed by pushing it to the requester, and is not available to pull from some directory, then this method is potentially very efficient.

6.5 Proof of possession in standards

Proof of possession is also an issue addressed in standards. The IETF has produced two RFCs which 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, and signing and decrypting are essentially the proposed mechanisms. As a third alternative the computation of a MAC on the certificate-request with a key derived from a secret, shared between the CA/RA and the requesting party, is proposed. This approach may not be of 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 depending on whether or not they use information concerning the receiver. 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 which everyone 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 purely from this perspective. 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).

6.6 Efficiency of POP-mechanisms

In the table below, the main properties of the mechanisms described in Section 6.4 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 issued certificate	CA: encrypt Client: decrypt	Two PK operations

6.7 Suitability of the methods for mobile environments

The most convenient mechanism to prove possession of a private signing key is probably 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 selection of mechanism will depend on a careful analysis of the precise implementation evironment. 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, although the number of PK-operations to be performed does not change and is therefore two.

7. Revocation in personal PKIs

7.1 Assumptions and requirements for Personal PKIs

- **Structure of the PAN**

  A PAN is usually a relatively small structure so it may be possible 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 greater than in conventional PKI environments, as the number of users is likely to be much smaller. This means all users can be reached at a certain time, or the CA can keep track of which users have not 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 the particular requirements of a PAN.

7.2 PAN-specific versus general revocation mechanisms

In conventional PKIs 'pull mechanisms', such as the provision of revocation lists or online status mechanisms, are typically used to provide access to revocation information. This is the case for the following reasons.

- Often a large number of clients take part in the PKI. Thus 'push' services would lead to a bandwidth problem and associated management problems for the CA. A

broadcast service could be implemented, but then clients not logged in during the broadcast would not receive the most recent revocation information.

- Large PKIs are usually long-term PKIs, where revocation information has to be provided covering a lengthy time period.
- Not all clients are always online, and so some will not receive all revocation information that is distributed.
- Not all clients are interested in revocation information at all times. This would make a broadcast potentially inefficient.
- The revocation information gets so large that not all clients are able to cache the information. Thus it must be possible to load the information when required.

In contrast to conventional PKIs, push services can probably be implemented in Personal PKIs, as the following assumptions hold.

- Personal PKIs may be short-term 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 consequence one might introduce short term certificates that are only valid for a short period of time and have to be renewed frequently, e.g. every day, as is already done in WAP in the SSL-server-certificate context. This could result in the removal of any need for revocation mechanisms.
- Only a restricted number of parties take part in the Personal PKI. Therefore more time or resource consuming mechanisms (such as the issuing of short term certificates as described above), that do not scale in large PKIs, might be feasible for implementation 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 that they have already received.

7.2.1 Mechanisms adapted from conventional PKIs

Revocation mechanisms used in traditional PKIs can be adapted to perform revocation management in Personal PKIs as well. The two general approaches, i.e. using CRLs or requesting status information online, can be implemented. A short summary of the discussion of relevant issues follows.

CRLs

A major fault with CRLs is that, in the mobile domain, they 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. (Of course, this does not necessarily rule out the use of delta-CRLs, but they carry their own significant management overhead).

Therefore we focus on the online status protocols OCSP and XKMS:

OCSP

OCSP, as an online revocation method, uses signed messages from the OCSP responder to the client (in our case a mobile handset) to convey revocation information. The purpose of OCSP is to provide revocation status and nothing else. OSCP has been developed by the IETF PKIX group. Several vendors, such as Baltimore, Valicert, VeriSign, Entrust, provide OCSP client and server implementations. 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 can then 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.

XKMS

XKMS, like OCSP, provides an online certificate revocation checking method. XKMS, however, offers more than just certificate revocation; it can also check the certificate validity and process a certificate chain path. It also allows for key registration. 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. 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 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.

If the two online status protocols are compared, the following conclusions can be drawn. Signed OCSP responses and requests are 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). The encoding method for OCSP messages is preferred to that of XKMS as the mobile world has already 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. Overall, OCSP seems to be the preferable solution for certificate revocation in the short-term future because of the significantly smaller size of its messages, and the fact that it can be implemented more easily on the client side.

7.2.2 PAN-specific mechanisms

Because of the particular characteristics of a personal PKI, new mechanisms to manage revocation, not appropriate for traditional PKIs, can be used. Generally the new situation is that the PKI is a relatively small structure. Therefore it may be possible to implement some kind of push mechanism. That means that a member of the PKI gets informed automatically about recent revocation incidents. Several possible models for this can be devised.

- **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 approach, 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 mechanism is only efficient when there are not too many revocations.
  2. *Automatic distribution of new revocation incidents*
     This solution requires the introduction of 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 is a more efficient way to distribute only recent incidents, but as past experience indicates, 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 only makes sense when the number of participants is not too big. The time at which white-lists are published must be specified. 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 mechanism just announcing the new member, as for the proposed mechanisms to distribute revocation incidents only (see above).
- **Ad-hoc-distribution of CRLs** (as already discussed in section 2.2)
  4. The idea behind this approach is that the distribution of CRLs is done between the clients, so that the necessity to contact a directory is removed. Determining the currency of CRLs can be handled by introducing serial-numbers or time-stamps, and clients always update to the most recent version. That is, whenever mobile devices communicate, they exchange the serial number (or time-stamp) of the CRL 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, 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.

7.2.3 Issues with local caching in clients

An important question for revocation in PAN scenarios is whether very limited devices can cope with certain revocation information at all. In the case 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 large 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. One example of this latter case may arise in 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 many certificates.

7.2.4 Support of multiple mechanisms

As discussed in the previous section, mechanisms exist that are attractive for a certain class of devices, whereas they may be less attractive or even infeasible for another class. Thus there many be a need to integrate at least two revocation mechanisms in one PAN. New issues may arise from the introduction of different mechanisms.

- It could be necessary to implement additional logic 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.

7.3 Suitability of the methods for mobile environments

PANs offer many possibilities for implementing revocation mechanisms differing from the ones familiar in fixed network scenarios. Currently these novel mechanisms are only proposals, and have not yet been implemented. It will be difficult to implement mechanisms that relate to the client software, as this functionality must first be standardised. Nevertheless, as PAN scenarios get more and more attention in the mobile world, and given that PANs will be an essential part of tomorrow's mobile infrastructure, it is necessary to consider these concepts and to develop solutions that may be more efficient than the ones available today.

8. Summary and Conclusions

After defining the requirements for PKI in a personal area network, we have looked at concrete issues such as imprinting devices, management of certification authorities and revocation mechanisms. The discussions in this paper have shown that whilst many mechanisms and protocols from the fixed network environment may be used or adapted for the PAN environ,ent, new mechanisms, not feasible in conventional fixed network PKI scenarios, may be advantageous in PANs. The latter category of mechanism is probably best represented by the new imprinting protocols where the user has to act as a trusted channel and by the various scenarios proposed for revocation checking based on push-mechanisms.

Acknowledgements

The authors would like to thank all their colleagues in the IST SHAMAN project who have contributed to the development of the results described in this paper, especially Jozef Dankers, Theo Garefalakis, Heiko Knospe, Scarlet Schwiderski-Grosche and Tim Wright. Further details on the results presented in this paper may be found in the SHAMAN project deliverables, available at http://www.ist-shaman.org.

References

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

[2] 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.

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

[4] D. Maher, *Secure communication method and apparatus*. US Patent number 5,450,493, September 1995 (filed December 1993).

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

[6] 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.

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

[8] SHAMAN Deliverable D13, *Final technical report – results, specifications and conclusions*. [Available at http://www.ist-shaman.org].

[9] 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.