CPFP: An Efficient Key Management Scheme for Large Scale Personal Networks

Shahab Mirzadeh, Rahim Tafazolli
Centre for Communication Systems Research (CCSR)
University of Surrey
Guildford, UK
{S.Mirzadeh, R.Tafazolli}@surrey.ac.uk

Frederik Armknecht
Chair for System Security
Ruhr-University Bochum
Bochum, Germany
frederik.armknecht@trust.rub.de

Jordi Jaen Pallares
FOKUS Fraunhofer-Institute
Berlin, Germany
jordi.jaen.pallares@fokus.fraunhofer.de

Hossam Afifi
Institut National des Télécommunications (INT)
Evry cedex, France
hossam.afifi@int-evry.fr

***Abstract*— This paper provides an efficient key management scheme for large scale personal networks (PN) and introduces the Certified PN Formation Protocol (CPFP) based on a personal public key infrastructure (personal PKI) concept and Elliptic Curve Cryptography (ECC) techniques.**

***Keywords*—certified PN formation protocol (CPFP), key management, personal networks (PN)**

I. INTRODUCTION

The concept of Personal Networks (PNs) was introduced and studied during the IST-MAGNET project [1]. Fig. 1 depicts its communication architecture which consists on the one hand of a dynamic collection of personal nodes and devices around a user (Private Personal Area Network or P-PAN) and on the other hand of remote personal nodes and devices in different clusters (home cluster, office cluster, car cluster…) that are connected to each other through an infrastructure or ad hoc networks.

Security and privacy are one of the major concerns in the development and acceptance of PN technologies. For this purpose, well-established cryptography to protect the integrity and confidentiality of the data should be used as much as possible. Symmetric key algorithms like the 128-bit Advanced Encryption Standard (AES) are expected to provide strong protection beyond the year 2031 [3]. However, the use of AES requires the establishment of shared keying material in advance.

With lack of permanent access to a common trusted third party in PN environments and also user unwillingness to delegate her trust to a centralized entity outside her personal territory, classical network security mechanisms based on the conventional public key infrastructure (PKI) and certificate authorities (CA) cannot be directly applied to the PN. In MAGNET phase 1, a key agreement protocol based on an authenticated Diffie-Hellman (DH) protocol, named PN Formation Protocol (PFP), was developed, which fulfills the security needs of small networks [4]. In this paper, we introduce a new key agreement protocol based on a personal public key infrastructure (Personal PKI) [5] and Elliptic Curve Cryptography (ECC), which is scalable to larger PNs and provides an enhanced level of authentication and non-repudiation with ease of key revocation and key update.

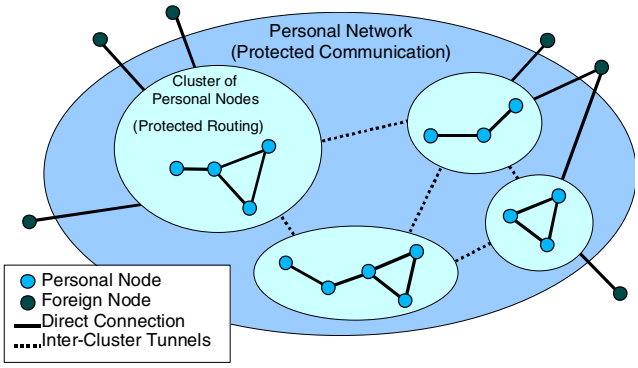

Figure 1. PN Communication Architecture [2]

II. CERTIFIED PN FORMATION PROTOCOL (CPFP)

CPFP is based on a personal public key infrastructure (Personal PKI) in which instead of global certificates issued by a trusted third party, the local certificates issued by the PN certificate authority (PNCA) will be applied. CPFP has two different stages. In the first stage, all PN devices get imprinted with the PNCA i.e., establish the PNCA signature public key as the PN root key and get a certificate on their own long term Diffie-Hellman public key. In the second stage, PN nodes use their certificates to authenticate each other and establish pairwise keys based on the Elliptic Curve Menezes-Qu-Vanstone (ECMQV) [6] protocol.

The ECMQV is the elliptic curve variant of MQV [7] key establishment protocol which is incorporated in the public key standard IEEE P1363 and is based on two sets (public and

978-1-4244-1653-0/08/$25.00 ©2008 IEEE

private) of long term (static) and ephemeral (dynamic) Elliptic Curve Diffie-Hellman (ECDH) keys. As a prerequisite in ECMQV, peers should a priori possess authenticated copies of each other's long term public keys which will be done through the issued certificates within the first stage of CPFP.

A. CPFP Stage 1 – Initializing and imprinting with PNCA

PN security depends on the security of the imprinting procedure which is subject to the following assumptions:

- The user is in full control of the imprinting procedure and determines when and how new devices get imprinted with PNCA and taken as members of her PN.
- The personal devices share two different communication interfaces with PNCA including Proximity Authenticated Channel (PAC) and usual (insecure) wireless communication channel.

A proximity authenticated channel is a communication interface between two devices, which is authenticated by physical means of user. Typically, proximity authentication is performed by touching the device, or by reading from or entering to a device's interface. We distinguish between two types of PAC channels, private and public PAC channels, with respect to the level of security the PAC channel can provide. A private PAC channel provides authenticity, integrity and confidentiality, while a public PAC channels provide authenticity and integrity only.

A typical example of a private PAC channel is realized by a user, who reads an alphanumeric string from the display of one device and then enters it to the other device using the keypad. Clearly such a channel sets some limits to the length of the string that can be transferred from one device to another, e.g., typically 32-40 bits, which is feasible to be transferred using devices' user interfaces by the user. Typical realizations of public PAC are RFID tags, Infrared communication, and public displays on the devices (such as an overhead display over a cashier, printer, or network access point [8]). If the PAC is public, the protocol requires that at least 160 bits of information can be transferred over it.

The user starts CPFP by choosing one device with keypad and display as the PN certificate authority (PNCA) and imprints all PN devices with it. The PNCA initializes itself with the generation of a pair of public and private ECDSA (Elliptic Curve Digital Signature Algorithm) signature keys and other PN components initialize themselves with the generation of their long term ECDH public and private keys. The parameters are based on a fixed elliptic curve with standardized coefficients e.g. P-192 recommended by NIST.

PNCA and PN components exchange their public keys (signature and long term) over the insecure wireless channel and the user authenticates the procedure with help of the complementary PAC channel. The outcome of this stage is that the PNCA issues certificates for the long term public key of each paired component which can be at the same time verified by all PN components. Based on the used PAC, there are two different procedures for this stage of the protocol:

1) Imprinting over Private PAC

In this version of the protocol, after the public keys are exchanged over the insecure wireless channel, the PNCA generates a key K which is suitable to be used in a Message Authentication Code (MAC) function which is shared by all PN components. Using this key K, the PNCA computes a MAC of the exchanged public keys. Both the key K and the MAC value should be feasible to be transferred by the user interfaces of the devices over the private PAC (at most 8 digits). This means that the MAC value should be truncated to 4 digits. One possible way of doing it is to take the 32 least significant bits of it, turn it to an integer, and then take the 4 least significant digits of it.

There are different scenarios, depending on the types of available interfaces. For example, if the PNCA has a display and the PN device has a keypad, then the key K and the truncated MAC are displayed by the PNCA to the user who enters them into the pairing device. The pairing device uses the received key value K to compute the truncated MAC value on the exchanged public keys (received over the insecure wireless channel) on its own. In a second step, it compares the result with the entered information and shows an accepted or rejected signal (peeps or blinks a light) to the user who updates the PNCA (Fig. 2a).

As the key K is chosen randomly each time and the private PAC provides confidentiality, an attacker gets no knowledge on the key K or on the MAC from the protocol runs. Hence, the only possible attacks are to block the messages over the Private PAC to prevent that the imprinting stage from finishing or to replace the key of the PN device with its own key for impersonation and to hope that the MAC value remains valid by coincidence. Assuming a message size of 8 digits, the probability for success is less than 2^{-16}.

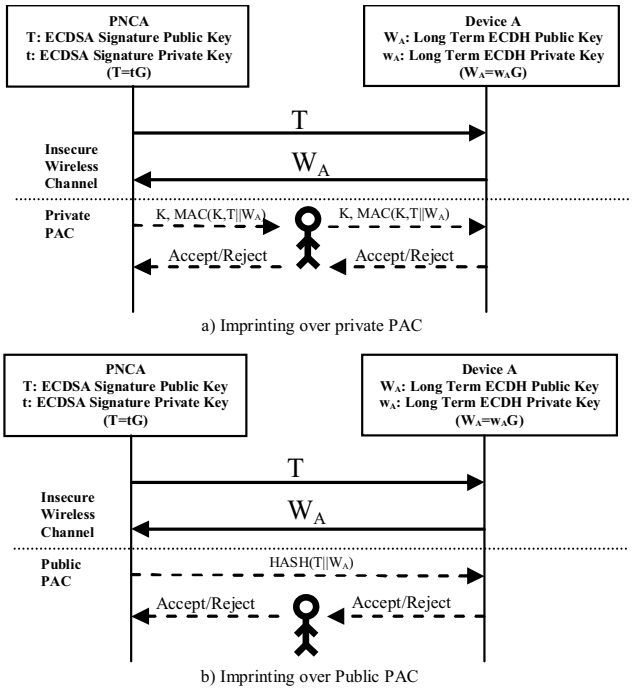

Figure 2. CPFP first stage - Imprinting

2) Imprinting over Public PAC

In this version of the protocol (Fig. 2b), after exchanging the signature and the long term public keys over the insecure wireless channel, the PNCA generates a hash of the exchanged public keys and sends it to the pairing device over the public PAC. The pairing device calculates the hash of the exchanged public keys, compares the result with the received data over the public PAC, and shows an accepted or rejected signal to the user who updates the PNCA.

As the public PAC provides integrity and authenticity, an attacker can either again block the messages to prevent the completion of the imprinting stage or replace the PN key W_A with another key to achieve impersonation. The replacement remains only undetected if the hash value would be the same. However, if the hash function is collision resistant, this is possible only with a negligible probability.

B. CPFP Stage 1- Getting Certificates from PNCA

The use of digital certificates is an established method to generate trusted identities in network communications. A certificate provides a binding between identity information and a public key; a key pair can subsequently be used for key exchange to set up secured communications as well as for digital signatures to validate transactions. In CPFP, certificates are used to bind the user friendly identities of PN components to their long term ECDH public keys. This ensures that once the certificates are issued by the PNCA and while they are not revoked or expired, the identities and their long term ECDH public keys are trustable by all PN components.

The PN components' identities are locally chosen in our key management system and can be any unique name in the PN environment. Because of the dynamic and heterogeneous nature of the PN and also because of the distribution of PN nodes in different clusters (fixed or mobile), MAC address or IP address (main candidate for homogeneous static network) can not be used as identities in the considered scenario. On the other hand, in CPFP all PN devices get certificates on their long term ECDH public keys and rarely change them, so a hash value of these long term ECDH public keys is a good candidate for a PN identity in MAGNET. To make the recognition of different components as easy as possible for the user, she will choose a user friendly name (UFN), including the PN name and/or the owner name, for each component during the imprinting and use these UFNs as their identities.

RSA, DSA and ECDSA are three standard algorithms that are usually used for digital signatures [9]. The use of ECC-based signatures with digital certificates provides both size and performance advantages. ECC-based signatures on a certificate are smaller and faster to create; and the public key that the certificate holds is smaller too.

The process of issuing certificates by the PNCA is as follows:

- After receiving the authenticated copy of the device's long term public key (during the imprinting procedure), the PNCA asks the user for extra information which should be included in the certificates like a user friendly name (UFN) and a validity period. Based on the received information and on the device's long term public key, the PNCA constructs a message m.
- The PNCA selects an ephemeral random secret private key k from the interval [1, n-1] which has an inverse modulo n.
- Then, it computes R = kG with G being the generator of the used elliptic curve and converts its x-coordinate to an integer x_1.
- Next, it computes $r = x_1 \bmod n$. If r=0, it goes back to step 2.
- Otherwise, it computes e = h(m) with h being a hash function
- Then, it calculated $s = k^{-1}(e+tr) \bmod n$, where t is its ECDSA signature private key. If s=0, it goes back to step 2.
- Finally, it outputs the message m with its signature (r, s) as the issued certificate for the paired device.

Each PN component is equipped with the PNCA's public key during the imprinting procedure. Given a certificate m and a signature (r,s), a PN component verifies its validity by performing the following procedure:

- Verify if r and s are from the interval [1, n-1]. If they are not, stop and reject the signature.
- Compute e = h(m)
- Compute $w = s^{-1} \bmod n$
- Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$
- Compute $R= u_1G + u_2T$, if $R = \infty$ reject signature
- Convert x-coordinate of R to an integer, x1 and calculate $v = x1 \bmod n$
- The device accepts the signature if v = r.

Observe that the algorithms described above are the established ECDSA algorithm (e.g., see [10]). It is believed to be secure according to the current state of knowledge if the parameters are appropriately selected.

C. CPFP Stage 2 – Using ECMQV to Drive Shared Key

In the last stage of CPFP, the Elliptic Curve Menezes-Qu-Vanstone (ECMQV) [6] key agreement protocol is used to establish a shared secret key between PN components which have already imprinted and have got PNCA' certificates on their long term public keys. The PNCA itself participates in this stage to establish shared pairwise keys with other PN components, with issuing a self signed certificate on its long term ECDH public key.

While based on ECDH, ECMQV offers attributes – such as key-compromise impersonation resilience and unknown key-share resilience – that are not found with ECDH. ECMQV has many desirable performance attributes, including the fact that the dominant computational steps are not expensive while the protocol also has low communication overhead, is role-symmetric, non-interactive and does not use encryption or

time-stamping. This makes it ideal in the development of security protocols and systems that require efficient and authenticated key agreement protocol and was chosen as a one of the three recommended key management protocols in NSA Suite B cryptographic primitives to be used to protect classified and unclassified sensitive information. For example, ECMQV is proposed for securing US Federal government communications up to the TOP SECRET classification (for more information, see [11]).

We are using a three-pass version of ECMQV (Figure 3) with the following protocol messages [12]:

1) $A \rightarrow B: R_A, Cert\_A$

A generates its ephemeral (dynamic) public and private keys (r_A, R_A) and sends its ephemeral public key (R_A) along with its long term public key certificate (Cert_A) to B.

2) $B \rightarrow A: R_B, Cert\_B, MAC(k1, 2||UFN_B||UFN_A||R_B||R_A)$

Upon receipt the first message, B does the following:

- Performs an embedded public key validation of R_A to verify it possesses certain arithmetic properties.
- Generates its ephemeral public and private keys (r_B, R_B).
- Computes an implicit signature "$s_B = (r_B + \check{R}_B w_B) \bmod n$" and a shared key "$K = hs_B(R_A + \check{R}_A W_A)$" and verifies that $K \neq \infty$ ($\check{R}_B$ and $\check{R}_A$ are the first "$L=[((\log 2\ n)+1)/2]$" bits of the first component of the point R_B and R_A).
- Using shared key derivation function (KDF), B derives k_1 and k_2 from the x-coordinate of the shared key K.
- Compute $MAC(k1, 2||UFN_B||UFN_A||R_B||R_A)$ and send the result along with its ephemeral public key R_B and its long term public key certificate Cert_B to A.

3) $A \rightarrow B: MAC(k1, 3||UFN_A||UFN_B||R_A||R_B)$

With receiving the second message, A does the following:

- Perform an embedded public key validation of R_B to verify it possesses certain arithmetic properties.
- Compute an implicit signature "$s_A = (r_A + \check{R}_A w_A) \bmod n$" and a shared key "$K = hs_A(R_B + \check{R}_B W_B)$" and verify that $K \neq \infty$.
- Using shared key derivation function (KDF), derive k_1 and k_2 from the x-coordinate of the shared key K.
- Compute $MAC(k_1, 2||UFN_B||UFN_A||R_B||R_A)$ and verify it based on the received message 2
- Compute $MAC(k_1, 3||UFN_A||UFN_B||R_A||R_B)$ and send the result to B.

B computes $MAC(k_1, 3||UFN_A||UFN_B||R_A||R_B)$ and verifies it based on the message 3. The session key is k_2.

This part of the CPFP protocol makes use of ECMQV as an established protocol which is secure according to the current state of knowledge. However, if this assumption should turn out to be unjustified at some point in time, one could imagine variations of CPFP which use other key agreement protocols.

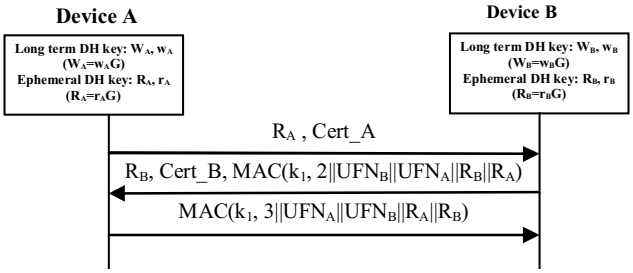

Figure 3. CPFP Stage 2 – Using ECMQV to derive shared keys

ECMQV was mainly chosen because of performance issues and not because it differs in its functionalities from other key exchange protocols.

III. KEY REVOCATION MECHANISM

Like a certificate authority in a normal PKI, the PNCA is not only in charge of inviting nodes into the PN but also to revoke them in the case of need. Since the user is the centre of the PN architecture, only the user herself should be able to decide whether a node has to be revoked or not. In practice, we envision the following procedure from a user's point of view to revoke one node.

Whenever the user logs into one PNCA device, he can choose to have a list of the currently valid PN members displayed. Given the list of current nodes, a user can select one or several devices and choose the REVOKE option to revoke these nodes.

When the revocation procedure is initiated, the actually used PNCA updates the Certificate Revocation List (CRL). The CRL is a file which contains all necessary information on the nodes that need to be revoked. This information include at least:

- PN's node identifier
- A time stamp and/or a CRL version number
- Serial number identification of the revoked certificate
- A code implying the reason of revocation

PNCA keeps a record of revoked certificates in a CRL up to their expiry date (each certificate has a specified expiry date). Each CRL has either a version number and/or a time stamp. With every revocation procedure, the PNCA updates the CRL and changes its version number and/or refreshes the time stamp. The CRL is signed with the private key of the PNCA (SK_{PNCA}) to ensure the non-repudiation, integrity, and message authenticity. As the revocation list is signed, each node can check its validity with the public key of the PNCA (PK_{PNCA}) obtained at imprinting time.

The new CRL is either distributed whenever a new revocation has happened and/or periodically (even if nothing has changed except the version number/time stamp). Nodes keep a record of the CRL locally, update it with revocation messages, and check its version with other communicating peers. If a node does not have the latest version of the CRL (or if it is overdue), it will update it.

It goes without saying that the CRL has only its value if it is ensured that every node has at any point in time the actual version. If two nodes exchange data, both must be sure that the other one has not been revoked since the last time they communicated. Thus, we envision that each node checks the current version of the CRL (either stored locally or retrieved from appropriate places) before a new communication starts (or at least in regular time intervals). This requires the following functionalities in the context of the CRL:

- The updated CRL can be distributed within the whole PN in a reliable way.
- The current version of the CRL can be provided upon request.

This can be realized in several ways. One possibility is to make use of the existing upper layer facilities in MAGNET, e.g., by using the Secure Context Management Framework (SCMF) [13] which is able to distribute and provide data on demand or adding an extra service to a MAGNET PN, a kind of 'revocation list service' which is discoverable through the MAGNET Service Management Platform (MSMP) [13]. The other approach is going for an ad-hoc CRL distribution scheme, where PN nodes ask each other for the latest version of the CRL and in case of difference, both nodes update to the latest CRL version.

IV. PNCA Resilience

The fact that the PNCA plays a central role in the PN's key management brings the problem of resilience. If the PNCA is broken or out of reach, the basic operations as inviting new devices and revoking keys should not be abandoned.

In the currently discussed approach, we use the fact that in principle the difference between the PNCA and an ordinary PN node is that the PNCA knows the secret key SK_{PNCA} that is mandatory for the operations mentioned above. This means that PNCA is rather a functionality than a certain device and if other devices share its knowledge of SK_{PNCA}, they can take over its functionality if necessary. Therefore, we propose to store SK_{PNCA} on different devices on several, strategically well-chosen locations. Each of these devices can act as the PNCA in the case of need, e.g., if the previous PNCA is unreachable or broken. As a device acting as PNCA has in principle full control over the PN as it can invite or revoke devices, it is of utmost importance that SK_{PNCA} is stored only in encrypted form to prevent an attacker to take over control of the PN if she steals a PNCA. If the value SK_{PNCA} is only protected by a key chosen by the user, e.g., derived from a password, this requirement is unfortunately most probably not fulfilled. Observations have showed that humans rather tend to choose insecure password which would compromise the security of the whole PN. A possible countermeasure could be to force the user to choose a strong password by refusing weak ones. Alternatively, one could imagine that the encryption is additionally protected by a piece of hardware. As it is already common practice for mobile phones, one could require the usage of a smart card together with a password to decrypt SK_{PNCA}.

Observe that decrypting SK_{PNCA} is only necessary once in the beginning of an epoch to turn a device into the PNCA. As long as the same device keeps this functionality, no user interaction is required in this point. After this epoch, the unencrypted SK_{PNCA} need to be erased from the memory so that only the encryption of SK_{PNCA} remains. At this time, the device looses its "superior knowledge" and becomes an ordinary node again.

Of course, knowing SK_{PNCA} is only half of the battle. It is likewise required that the PNCA has an actual list of PN members and revoked nodes. Therefore, the "old" PNCA and the "new" PNCA have to synchronize their lists to provide full functionality. If synchronization cannot be handled by the PN itself, one could think of storing this information on a portable medium like an SD card, possibly encrypted as wel. Thus, at the end of one epoch, when a device looses its PNCA role, it stores the current data on the medium. The device which becomes the next PNCA should have access to this medium to restore the actual data.

Acknowledgment

The authors are grateful of all the partners involved in the projects MAGNET and MAGNET-Beyond, especially work-packages 4, for their fruitful discussions and collaborations.

References

[1] IST MAGNET Beyond Project http://www.ist-magnet.org/

[2] M Petrova et al., "Conceptual Secure PN Architecture", IST MAGNET deliverable D2.1.1, December 2004, http://www.ist-magnet.org/GetAsset.action?contentId=939330&assetId=939344

[3] NIST Special Publication 800-57, "Recommendation for Key Management – Part 1: General (Revised)", May 2006, http://csrc.nist.gov/publications/nistpubs/800-57/SP800-57-Part1.pdf

[4] S. Mirzadeh et al. , "Final version of the Network-Level Security Architecture Specification", IST MAGNET deliverable D4.3.2, Feburary 2005, http://www.ist-magnet.org/GetAsset.action?contentId=942902&assetId=943012

[5] C. J. Mitchell and R. Schaffelhofer. The personal PKI. In C. J. Mitchell, editor, Security for Mobility, chapter 3, pages 35-61. IEE, London, UK, 2004.

[6] L. Law, A. Menezes, M. Qu, J. Solinas, & S. Vanstone, An efficient protocol for authenticated key agreement, Designs, Codes and Cryptography, 28(2), 2003, 119–134

[7] http://en.wikipedia.org/wiki/MQV

[8] D.Balfanz, D.K.Smetters, P.Stewart and H.Chi Wong. "Talking To Strangers: Authentication in Ad-Hoc Wireless Networks". Technical report, Xerox Palo Alto Research Center, Palo Alto, 2002, http://www2.parc.com/csl/members/balfanz/publications/loclim.pdf

[9] NIST FIPS PUB 186-2, DIGITAL SIGNATURE STANDARD (DSS), January 2000, http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf

[10] Don Johnson, Alfred Menezes, Scott Vanstone, The Elliptic Curve Digital Signature Algorithm (ECDSA), International Journal of Information Security, 2001 - Springer

[11] Fact Sheet NSA Suite B Cryptography, http://www.nsa.gov/ia/industry/crypto_suite_b.cfm?MenuID=10.2.7

[12] Darrel Hankerson, Alfred Menezes, and Scott Vanstone, "Guide to Elliptic Curve Cryptography", Springer-Verlag New York Inc., 2004

[13] M Jacobsson et al., "Specification of PN networking and security components", IST-MAGNET Beyond deliverable D2.3.1, December 2006, http://www.ist-magnet.org/GetAsset.action?contentId=1111179&assetId=2253549