The EAP Smartcard, a tamper resistant device dedicated to 802.11 wireless networks.

Pascal Urien[1], Marc Loutrel[2]

[1]Ecole Nationale Supérieure des Télécommunications 46, Rue Barrault.
75013 Paris France
Pascal.Urien@enst.fr

[2]Laboratoire d'Informatique de Paris 6.
8 rue du Capitaine Scott
75015 Paris.France
Marc.Loutrel@lip6.fr

Abstract

This paper introduces the architecture of a new type of smartcards dedicated to security in wireless networks. Their functionalities have been described in a recent internet draft in which we propose an interface to integrate the **E**xtensible **A**uthentication **P**rotocol (EAP) into smartcards. These devices ensure security features, such as authentication or visitor's credentials needed for services deployment in 802.11 wireless environments.

1 Introduction

Coming years are expected to be stellar for Wireless LAN market growth, driven by the increasingly cheap and reliable 802.11 technology. Demand for wireless access to LANs has raised due to mobile computing devices, such as laptops and personal digital assistants, and a desire for seamless and permanent connections to the network without using cables. However security still remains the *Achille heel* of these emerging ubiquitous networks.

With wireless-virtual plugs new threats appear. Sometimes they are referred as *parking lot attacks* [1]. In wired technologies, security is mainly ensured by restricting network access, to authorized staff. As radio links can't be confined in a safety area, it's necessary to create more well-fitted security mechanisms like,

- **Network accesses control**; visitors are authenticated, and according to their credentials use services , to which they have subscribed.
- **Packet signature**; an authenticated visitor exchanges frames that are identified by its 802 MAC address and its IP address. Messages' signature, prevents spoofing attacks, and supports non repudiation features, that may be mandatory for service billing.
- **Information privacy**; visitor's data is ciphered in order to ensure protection against eavesdroppers, like packet sniffers. However, in some cases encryption is realized at application level.

The WEP protocol, defined in the IEEE 802.11 [5] standard aims at supporting these three services, but many threats have been highlighted,

- Authentication doesn't efficiently work; it's easy to record a key and to use it again [2].
- Packet signature mechanism (CRC encryption) is weak. It's possible to modify a byte in a ciphered packet, and to compute a right CRC value [2].

- Information privacy architecture is not scalable as static secret values (40 or 104 bits) are shared between access point and wireless devices. Furthermore these shared secrets can be recovered by recording about one million enciphered frames [4].

Next generation of 802.11 networks will support the Temporal Key Integrity Protocol (TKIP) currently under definition at the IEEE 802.1i committee. TKIP works with a master key (MK), from which enciphering keys are derived and used for each incoming or outgoing packet. A strong 64-bit signature called *Message Integrity Code* (MIC) is appended to every frame. MK is obtained from the IEEE 802.1x [6] authentication process. An ephemeris key (*Transient Key*) is deduced from this parameter and can be updated via a *re-keying process.* Other keys (packets encryption and signature) are obtained from this transient key.

IEEE 802.1x [6, 18] architecture (see figure 1) deals with three main entities,
- ***The supplicant***, a computer that intends to exchange IP traffic over a wireless network,
- ***The authenticator*** in WLANs is typically an access point which runs WEP or TKIP protocols. A master key (MK) is shared between the access point (AP) and the supplicant. This key used for radio security purposes is generated by the AP, which may securely send it to the supplicant via an EAP-Key frame, whose content is encrypted by another key called the session key (SK),
- ***The authentication server*** (AS). This server is linked to an AAA (Authentication, Authorization, Accounting) infrastructure that typically stores network user profiles (subscriptions, credentials …) and authentication parameters (symmetric cryptographic keys, X509 certificates …). Authentication data is shuttled by the EAP protocol, which is carried in 802 frames between the supplicant and the authenticator, and encapsulated by the RADIUS [14] protocol (over IP) between authenticator (acting as a NAS entity) and the authentication server. The Authenticator transparently forwards EAP messages so it acts as a relay between the supplicant and the RADIUS server. At the end of this process a session key (SK) is effectively shared by both parties and is (securely) sent by the RADIUS server to the access point via a procedure that is not standardized. Either computing an encryption key from the RADIUS shared secret or working with an IPSEC link are possible ways of transmitting the SK.

EAP has emerged as a standard layer to support various authentication mechanisms, such as challenge access protocol [8], SSL/TLS [13], SIM [9], without having to pre-negotiate a particular one. One major benefit of EAP is to perform user authentication, and therefore to enable services, in various *AAA* environments [3] like,
- Microsoft platforms. Most companies use these platforms to deliver basic business services to their employees (email, web services …). By 2003, 33 percent of windows 2000 corporate users are predicted to use a smartcard to logon[1] .
- Mobile phone operators' authentication centre (AuC). Many operators see 802.11 hotspots as an extension to their existing wireless networks (like for example GPRS). As a consequence, subscribers will be authenticated and will be billed through SIM smartcards.

[1] source: Gartner 2001

- Public Key Infrastructure. The use of certificates, allows off line authentication (according to delegation mechanisms), and doesn't imply a link to a central server.

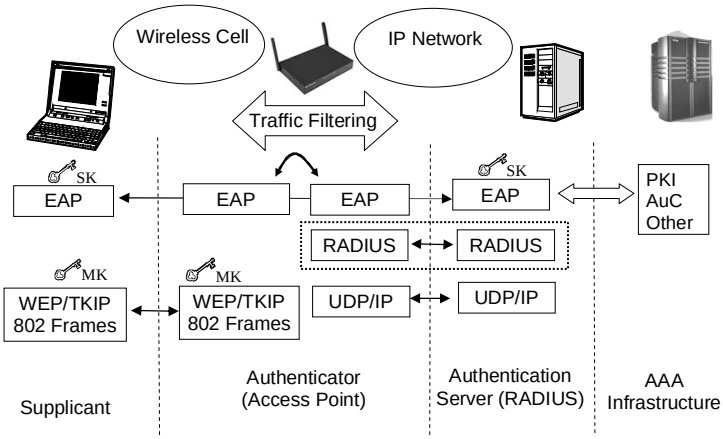

Figure 1: 802.1x architecture.

The EAP message structure [11] is quite simple and consists of two parts:
- A header that includes the following parameters: Code, Identifier, Length and Type.
- A payload that includes information relative to a particular authentication scheme. The standard suggests organizing the payload with one-byte prefix indicating the length of data followed by the data-byte stream.

The *Code* is a one-byte field that identifies EAP packet functions. The *Code* has following values: 1 for request, 2 for response, 3 for Success, and 4 for failure. The *Identifier* is a one-byte field and aids in matching responses with requests. The *Length* field is a two-byte field and indicates the total length of the EAP packet. The *Type* field is one octet and identifies the structure of an EAP Request or Response packet. All EAP implementations support types 1-4, 1 for identity, 2 for notification, 3 for NAK (Response only) and 4 for MD5 challenge.

An EAP authentication scenario begins by an Identity request message, followed by one or more request/response messages relative in order to agree on the authentication type to be used. It normally ends by a notification indicating either Success or Failure.

Keying materials could be deduced from these messages and stored in smartcard files. The station uses these keys for packet signing or enciphering. In many cases, wireless services providers need that subscribers don't know the secret keys that are requested for their authentication. This feature is particularly useful to avoid cloning as it will imply a lack of revenues. As smartcards are considered as the best tamper resistant devices, we believe that they are the cornerstone for secure user authentication, in conjunction with the EAP protocol.

2 How to deploy smartcards in wireless networks

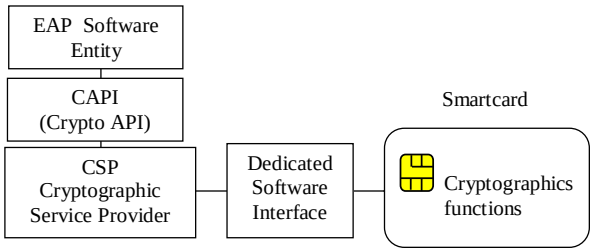

Figure 2: A classical procedural concept, working with win32 systems

There are two main approaches for deploying smartcards in wireless networks.

- **Classical procedural concept**. Cryptographic functions located in smartcards are invoked according to traditional remote procedure call paradigms. The EAP protocol is instantiated by a software entity, which manages all network information exchanged and uses smartcard services when it is necessary. A first illustration of such an architecture is shown in figure 2. An EAP application uses services delivered by the crypto application programmatic interface (CAPI) which in turn runs the cryptographic service provider (CSP) facilities in order to effectively compute cryptographic functions. Smartcard details are hidden by this last interface and no new standard is needed for smartcard integration in wireless networks. Another alternative is to directly interact with existing smartcards, like the GSM SIM [20]. An EAP entity, that implements the EAP-SIM [9] protocol, could use PC/SC [19] to directly interact with the subscriber's smartcard (see figure 3).

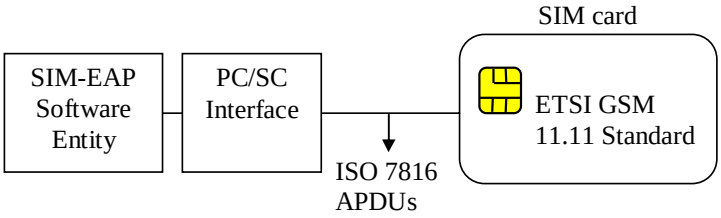

Figure 3: SIM-EAP entity working with a SIM card

- **Protocol approach**. An EAP engine runs in the smartcard (see figure 4), which processes EAP requests and produces corresponding responses [16, 17]. A light EAP entity is located on the terminal; it broadcasts EAP messages exchanged between the network and the smartcard. This software entity also manages a possible smartcard absence. For instance when the tamper device is not present a NAK message is generated upon each incoming request.

One of the major advantages of this second architecture is to introduce a direct communication channel between the embedded EAP entity and the RADIUS authentication server (see figure 5). This link can be used to securely exchange information like:

- *The subscriber profile*. A collection of data may include certificates giving the wireless visitor credentials and preferences.
- *The visited network profile*, a list of parameters including server names or cryptographic keys. As an illustration many internet protocols (RSVP [21] COPS [22]…) deal with symmetric shared secrets to compute message signatures (typically a digest value obtained from the message content and a password). A dedicated EAP authentication protocol, integrating privacy, integrity and digital signature features, could transport keys, needed by local services like QoS.
- *Data Management*. The EAP channel is used for smartcard management; this process is very similar to SIMs remote management that is currently implemented in GSM networks, by means of encrypted SMS exchanges [24]. Smartcard management means updating data (subscriber profile …) or downloading new applications (for example class files in javacards).

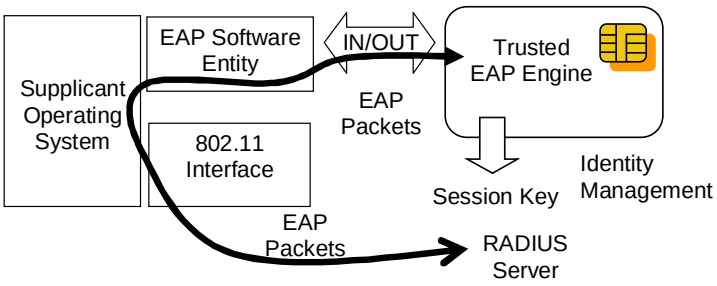

Figure 4: The EAP smartcard, software architecture.

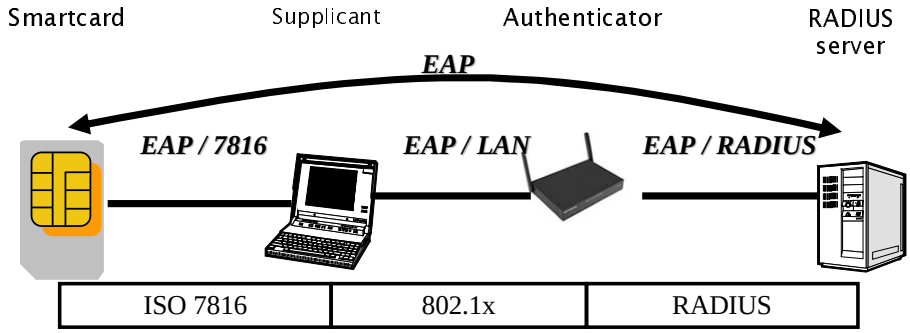

Figure 5. The EAP smartcard, logical architecture

3 The EAP-Smartcard

A first internet draft [23] defines four basic services (see figure 6), Get-Next-Identity, Set-Identity, EAP-Packets, Get-Session-Key shuttled by specific ISO 7816 [7] APDUs.

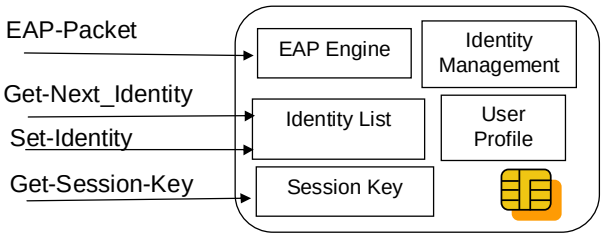

Figure 6. Basic EAP smartcard services

3.1 Smartcard Identity

The first key concept is smartcard identity. As we previously mentioned it, EAP authentication process begins by an identity request message received by the supplicant (the EAP smartcard in our approach). The supplicant delivers a response message that contains the system identity value (EAP-ID). This parameter is interpreted by the RADIUS server as a user login or a NAI [12] that could also be forwarded to another RADIUS server. The EAP smartcard may deal with several classes of wireless networks, managed by different administrative authorities. Therefore multiple system identities can coexist in this tamper resistant device.

Smartcard identity points to a particular system identity (EAP-ID), linked to an EAP type (working with a given authentication scenario) and its associated cryptographic keys (see figure 8). It may be of various types like:

- Service Set Identity (SSID) element, a 32-byte string that identifies an 802.11 network [5]. This parameter is obtained by the terminal operating system, and can be used as discriminator to set the EAP smartcard identity.
- A user identifier (UserId) that can be interpreted as a network access identifier [12].
- A pseudonym (LocalId), e.g. a friendly name. This parameter is for example a user account in a particular computing environment.

3.2 Get-Next-Identity

This service extracts a smart identity from a circular list. The terminal operating system browses the EAP card content and selects an identity according to pre-defined rules or displays the list of available identities.

3.3 Set-Identity

This service sets the smartcard current identity. It resets the EAP state machine managed by the smartcard and fixes the triplet (System Identity, EAP-Type, EAP-keys) to be used with the EAP protocol.

3.4 EAP-Packets

This service forwards an EAP request message or a notification to the smartcard. As EAP messages maximum length is 65535 (2^{16}-1), an incoming message can be segmented, according to the ISO 7816 standard, in a series of 255-bytes fragments. In a symmetric way, an outgoing message may be segmented in a set of 256-bytes fragments. Therefore the terminal EAP light entity performs all needed segmentation/reassembly operations.

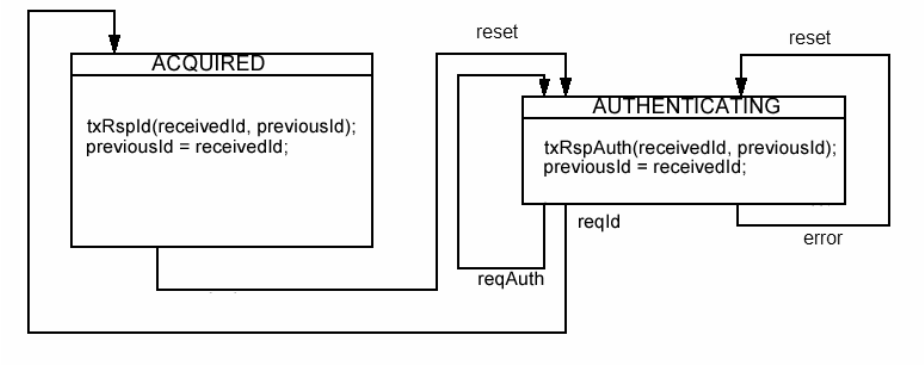

Figure 7. Simplified embedded EAP state machine.

The supplicant state machine, as described in 802.1x standard is split between the terminal and the smartcard. In fact smartcard only implements the AUTHENTICATING state (see figure 7). Upon reception of the Set-Identity command smartcard unconditionally transits in the AUTHENTICATING state. Upon reception of the EAP Identity-Request message, smartcard unconditionally moves in the ACQUIRED state, delivers an Identity response message and re-enters the AUTHENTICATING state. In agreement with the 802.1x state machine all EAP requests are processed in the AUTHENTICATING state. The final EAP notification message (either success or failure) indicates the end of the authentication process. If any error occurs during the authentication procedure (reception of NAK or failure messages …) the smartcard restarts at the AUTHENTICATING state where it will wait for an identity request or the first EAP-Type request.

If the EAP smartcard support security features like PIN code or biometric identification, all EAP messages will be silently discard before the occurrence of a successful bearer authentication.

3.5 Get-Session-Key

This service returns the session key (SK) that is sometimes available at the end of the authentication scenario. This key is used by the terminal to decrypt the WEP or TKIP master key. Some services provider could require that this parameter remains safely store in the EAP smartcard. For that purpose this function should be replaced by a procedure, for example named RUN_SK_ALGO that decrypts an input value.

4 Optional EAP card features

These functionalities are not described by the first version of the internet draft, but are issues under consideration that could be supported by upcoming versions.

4.1 Identity management

An EAP smartcard may be manufactured as an EAP engine that doesn't store any identity. However the smartcard operating system supports several authentication protocols, like EAP-SIM, EAP-TLS and so on. Identity management (see figure8) is performed by three services, Add-Identity, Update-Identity and Delete-Identity either

off-line, by means of specific ISO 7816 APDUs, or on-line (*over the air*). Off-line management means that needed operations are executed in a safe place. On-line management means that information is exchanged with a remote server, for example via the EAP protocol.

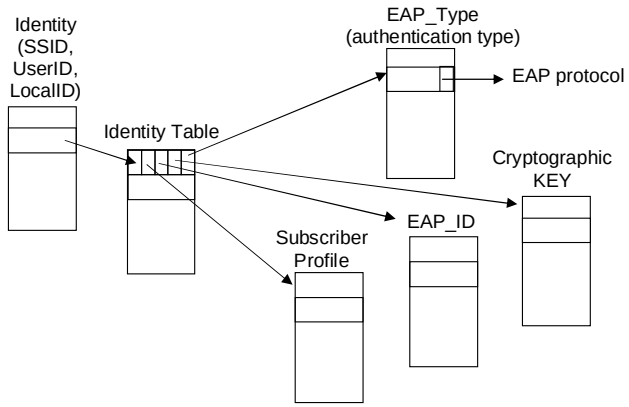

Figure 8. Overview of the identity management.

4.2 User profile

The user profile is a set of information which defines the subscriber credentials for services supported by a specific wireless internet service provider. This collection of data may be expressed in various syntax like ASN.1 or XML. Usually this information is stored in a LDAP repository, queried at each authentication procedure. EAP smartcards may store this profile (for example in the form of an X509 certificate), in order to support a distributed architecture that facilitates roaming with different WISPs (especially when PKI infrastructure is used) or that doesn't require connection to a central corporate repository.

5 Experimental platform

We have implemented the first internet draft in javacard, which natively includes an USB interface [15]. Consequently the tamper resistant device can be used in a classic PC, without any additional reader. In win32 platforms the notion of *EAP Provider* [10] has been introduced. Typically it is a software dynamic library (DLL) that implements the supplicant 802.1x AUTHENTICATING state. The operating system is in charge of all the remaining operations. The Remote Access Service entity (RAS) manages all resident *EAP provider* objects (see figure 9).

Upon system boot, the EAP provider object is invoked via the method RasEapGetInfo that returns three pointers to additional methods, RasEapBegin, RasEapMakeMessage and RasEapEnd.

Human user interacts with the EAP provider through a dialog box, started by the *RasEapInvokeConfigUI* procedure. In our experimental implementation this graphical interface is used to get the smartcard identity list and to select the appropriate one.

When the operating system detects a wireless cell, identified by its SSID, it performs the association process with the access point, sends an EAP-Start frame and activates the EAP provider that had been previously associated to this particular SSID. Then it calls the RasEapGetIdentity method, which in turn, sends an identity-request message to the EAP smartcard.

When the EAP identity request message is received, an EAP session is started and the system calls the RasEapBegin function. At this point the smartcard application may be selected if needed and subscriber can be asked to give his PIN code.

During an EAP session all EAP requests or notifications are sent to this RasEapMakeMessage method, which forwarded them to the EAP smartcard. At the end of the authentication scenario the RasEapEnd method is invoked.

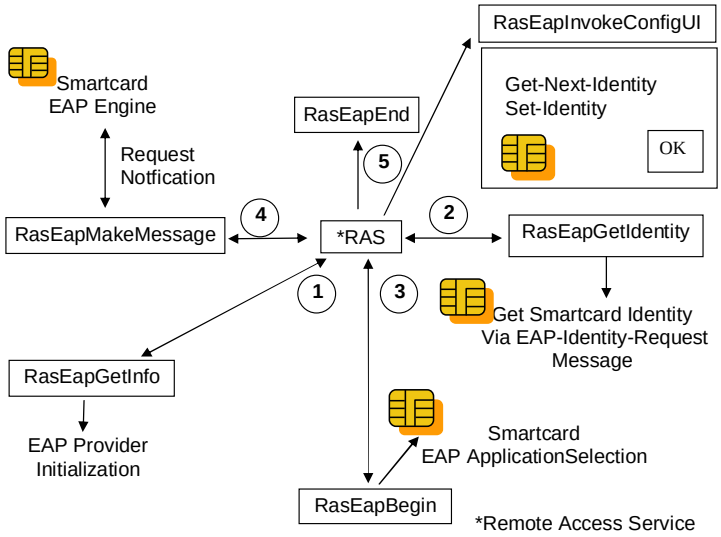

Figure 9: win32 experimental platform, EAP protocol sharing between smartcard and Operating System.

6 Conclusion

We have successfully introduced a new kind of smartcards dedicated to wireless networks. In this first step we have designed an interface that could work with existing operating systems, and that gives access to a secure EAP engine. We have clearly identify two working plans, the user plan used for identity selection and EAP messages processing, and an administrative plan eventually used for smartcard personalization and update. Although many issues are still under consideration, this work will be pursued through the WLAN smartcard consortium[2] that aims at defining smartcard specifications for wireless environments.

[2] www.wlansmartcard.org

7 References

[1] Arbaugh W, Shankar N, and Wan Y, “Your 802.11 Wireless Network has No Clothe” http://www.cs.umd.edu/~waa/wireless.pdf, 2001.

[2] Borisov N, GoldBerg I, Wagner D, “Intercepting Mobile Communications: The Insecurity of 802.11”, Proceeding of the Eleventh Annual International Conference on Mobile Computing And Network, July 16-21, 2001.

[3] Cisco, “A Comprehensive Review of 802.11 Wireless LAN Security and the Cisco Wireless Security Suite”, White Paper, http://www.cisco.com, 2002

[4] Fluhrer S, Mantin I, Shamir A, “Weakness in the key scheduling algorithm of RC4”, 8th Annual Workshop on Selected Areas in Cryptography, August 2001.

[5] IEEE 802, Part 11 « Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications » 1999.

[6] IEEE P802.1X *Approved Draft*, “Port based Network Access Control”, june 2001.

[7] ISO 7816, “Identification Cards - Integrated Circuit(s) Cards with Contacts”.

[8] Kamath V, Palekar A, “Microsoft EAP CHAP Extensions”, draft-kamath-pppext-eap-mschapv2-00.txt, September 2002.

[9] Haverinen H, Salowey J, “EAP SIM Authentication”, draft-haverinen-pppext-eap-sim-09.txt, January 2003

[10] Microsoft, “About Extensible Authentication Protocol”, http://msdn.microsoft.com

[11] RFC 2284, "PPP Extensible Authentication Protocol (EAP)", March 1998.

[12] RFC 2486, "The Network Access Identifier", January 1999.

[13] RFC 2716, « PPP EAP TLS Authentication Protocol », October 1999.

[14] RFC 2865, "Remote Authentication Dial In User Service (RADIUS)", June 2000.

[15] SchlumbergerSema, e-gate, http://www1.slb.com/smartcards/infosec/egate.html.

[16] Urien P, Tizraoui A, Loutrel M, “Integrating EAP in SIM-IP smartcards”, ASWN IEEE workshop on Applications and Services in Wireless networks, July 2002, Paris.

[17] Urien P, Loutrel M, Lu K, “Introducing Smartcard in Wireless LAN Security “, 10th International Conference on Telecommunication Systems, Monterey, California, October 3-6 2002.

[18] William A. Arbaugh, Arunesh Mishra, “An Initial Security analysis of the 802.1X standard”, www.cs.umd.edu/~waa/1x.pdf.

[19] PC/SC (1996) "Interoperability Specification for ICCs and Personal Computer Systems", *© 1996 CP8 Transac, HP, Microsoft, Schlumberger, Siemens Nixdorf.*

[20] ETSI - GSM 11.11 "Digital cellular telecommunications system (Phase2+); Specification of the Subscriber Interface Identity Module – Mobile Equipment (SIM_ME) interface".

[21] RFC 2205, "Resource Reservation Protocol RSVP", September 1997

[22] RFC 2748, "The COPS (Common Open Policy Service) Protocol" January 2000.

[23] Urien P, Farrugia A.J, Pujolle G, Groot M, “EAP support in smartcards“, draft-urien-eap-smartcard-00.txt, 55th IETF, Atlanta, November 2002.

[24] ETSI - GSM 3.48 "Digital cellular telecommunications system (Phase2+); Security Mechanisms for the SIM application toolkit".