Introducing Smartcard Enabled RADIUS Server

Pascal Urien[1], Mesmin Dandjinou[2]
[1]*Ecole Nationale Supérieure des Télécommunications (ENST), 37/39 rue Dareau 75014 Paris France*
[2]*Universite Polytechnique de Bobo-Dioulasso, Burkina Faso*
Pascal.Urien@enst.fr, Mesmin.Dandjinou@voila.fr

ABSTRACT

This paper introduces an innovative concept of smartcard enabled RADIUS server. We design RADIUS servers in which EAP messages are fully processed by smartcards, called EAP-Servers. When the well known TLS protocol is used as authentication method, this architecture becomes scalable. In that case, concurrent authentication sessions are simultaneously handled by different EAP-Servers, each of them securely embeds an unique X509 certificate and its associated private key. We presents experimental results obtained with commercial components, and demonstrate that system performances, about 5 seconds per RADIUS session, are compatible with today network constraints.

KEYWORDS: Smartcard, AAA, Security, EAP, TLS, RADIUS, WLAN.

1. INTRODUCTION

There is a trend in the smartcard industry to design tamper resistant devices supporting IP connectivity [1]. As an illustration, a recent paper [2] introduces USB full speed enabled smartcards, whose operating system supports classical communication OSI layers, such as Ethernet over CDC (*Communication Device Class*) TCP/IP, SSL and HTTP. Although USB interfaces are widely available in personal computers, an other class of secure components, sometimes referred as MegaSIM[1] also offers networking facilities (e.g. TCP/IP stack) working over MMC (*MultiMedia Card*) physical interfaces. This technology is natively adapted to mobile phones, that already include MMC memories used for the storage of multimedia contents. However USB or MMC interfaces only realize point to point link to hosts, and are usually supported by peripheral devices (printers, modems, network boards…), plugged to computers by means of these IO ports. Therefore IP enabled smartcards required docking hosts (see figure 1), which in most cases, access to this devices by means of standard Ethernet drivers (called NDIS in Windows platforms) used to manage LAN interfaces or modems.

Due to that, most of foreseen applications, used IP smartcards as personal WEB servers, plugged to computers, linked to its bearer by PIN code or biometric identification, and possibly offering mobile WEB services.

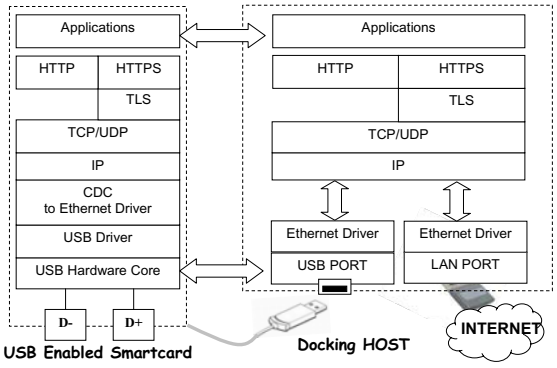

Figure 1. Example of USB Enabled Smartcard.

This paper is focused on server applications, in which smartcard acts as standard network device that doesn't offer services to human bearers, but rather enhances the security of communication infrastructures. For that, we have designed a scalable RADIUS server, based on smartcards. We deal with asymmetric cryptographic which authorizes elegant solutions; every smartcard holds an unique X509 certificate, and parallel computing techniques balance the modest computing capacities of smartcards. We demonstrate that our proposal is working in today networks, and could be improved by the introduction of components equipped with common wired (Ethernet) or wireless LAN interface.

[1] http://www.m-sys.com

This paper is organized according to the following plan. Section 2 describes classical RADIUS servers. Section 3 introduces smartcard enabled RADIUS server. Section 4 presents experimental results. Section 5 discusses about future directions of this work.

2. CLASSICAL RADIUS SERVERS

The RADIUS protocol [3] was introduced in the nineties, in order to solve authentication issues induced by millions of users, that intended to reach their Internet Service Provider through modems, dealing with the PPP protocol. It was then re-used by the IEEE 802.1x standard [4], that defines access control mechanisms for wired and wireless (IEEE 802.11) infrastructures. It deals with two entities, the Network Access Server (NAS) and the authentication server (AS). In a telephony context the NAS runs in a Point Of Presence (POP) , while in Wi-Fi applications it is located in Access Points (AP), and blocks all frames that are sent/received by unauthenticated users. Clients (called supplicants in IEEE 802.1x) are authenticated before allocations of their IP addresses. Authentication messages are exchanged between user and NAS over PPP or LAN frames. These messages are encapsulated in RADIUS packets exchanged between NAS and AS entities; RADIUS works over an UDP/IP stack, and therefore is routable through the Internet.

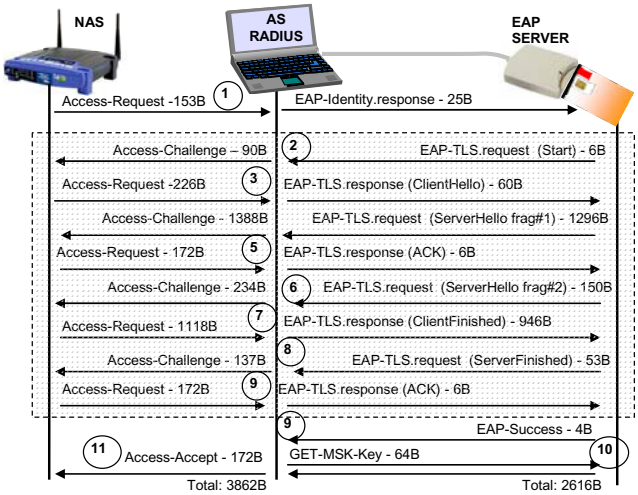

Figure 2. Example of Information Exchange, During a RADIUS Session.

The Extensible Authentication Protocol (EAP) was introduced in 1999 [5], in order to define a flexible authentication framework. Before its definition, it was necessary to standardize every new authentication procedure used by PPP (such as PAP, CHAP, MSCHAP), what slowed down security enhancements and leaded to a lack of interoperability. EAP is a powerful umbrella, able to transport many authentication scenari, between EAP client and EAP server. It basically works with three kinds of messages; requests delivered by servers; responses returned by clients; and notifications produced by servers in order to indicate success or failure of authentication procedures.

EAP packets are transparently shuttled in RADIUS payloads [6]. RADIUS is a key player in AAA (Authentication Authorization Accounting) infrastructures, it is widely deployed in Wi-Fi networks and should likely control accesses to emerging WiMax Metropolitan Area Networks [7]. EAP protocol is in charge of user's authentication, it should securely manage and store cryptographic credentials, needed by these operations.

Figure 2 illustrates a RADIUS dialog between a NAS and an authentication server (AS). EAP messages received/sent from/to network client (intending to reach Internet resources through an access point) are exchanged with the AS according to the following scenario:

- The *EAP-identity* message (see figure 2, item 1), produced by an EAP client is encapsulated by the NAS in an Access-Request packet. In the case of the EAP-TLS [8][9] authentication procedure, this identity is the subject field of the user's X509 certificate.

- AS extracts this EAP message. By analyzing the user's identity it deduces what class of authentication method is needed. Typically user's account parameters are stored in a LDAP directory logically working with the authentication server. This event begins an EAP session. In usual implementations the EAP-server software is merged with the RADIUS module. In our proposal EAP servers run in smartcards, and EAP messages are dispatched by the AS.

- The EAP session (see figure 2, shadowed area) is handled via couples of *Access-Challenge* packets, transporting EAP requests and *Access-Request* packets carrying EAP responses.

- The EAP server ends the authentication session by a notification (see figure 2, item 9), either *Failure* or *Success*, giving the procedure result. Upon success, the EAP server computes a master session key (MSK), which is afterwards forwarded by the AS to the NAS in an *Access-Accept* packet. MSK is a secret shared by access point and network client, in order to calculate all cryptographic materials needed for radio security purposes.

RADIUS security is based on a secret (called the RADIUS secret) shared between NAS and AS, and works with two mechanisms.
- Firstly, *Access-Challenge*, *Access-Reject*, *Access-Accept* packets include an MD5 authentication digest, calculated from the concatenation of a packet content, the RADIUS secret, and a random value (the authenticator field) included in a previous Access-Request .

- Secondly an optional security attribute (the Authenticator-Message, RADIUS attribute number 80) e.g. an HMAC-MD5 function (whose key is the RADIUS secret) computed over the whole packet content.

Most of RADIUS servers software implementations, use the well known *OpenSSL* library, in order to support the EAP-TLS [9] authentication procedure, a quite transparent encapsulation of the TLS protocol [8].

We believe that EAP server smartcards enhance the RADIUS security, specially in EAP-TLS case, for the following reasons,

- The server private key is securely stored and used by the smartcard.

- The client's certificate is autonomously checked by the EAP server.

- If the EAP client also runs in a smartcard, the EAP session is then fully processed by a couple of tamper resistant devices, working as *Secure Access Module* (SAM), a classical paradigm deployed in highly trusted architectures.

We will emphasis later that it's possible to embed RADIUS authentication servers in smartcards. But even if they include a TCP/IP stack, the lack of Ethernet connectivity in today components, implies the use of a docking host, offering USB or MMC interfaces and classical network interfaces (such IEEE 802.3 or IEEE 802.11).

However experimental results (presented in section 4) show that EAP-servers are slower than classical, all software based, RADIUS servers. One advantage of EAP-TLS is to elegantly allow parallel architecture, in which the slowness of smartcards is compensated by an acceleration factor induced by the scheduling of concurrent EAP sessions to multiple smartcards; each of them independently managing a particular session. Finally, from a security point of view, the benefits of RADIUS processing in smartcards is reduced by the constraint of sharing a secret with Access Points, whose security policies are not usually well defined or even specified.

In the next section we will introduce a smartcard enabled RADIUS server, working with realistic performances in today networks. Future developments are explored in section 5.

3. SMARTCARDS ENABLED RADIUS SERVERS

Our proposed *smartcard enabled RADIUS server* is made of two parts:

- A RADIUS authentication server, running in a docking host. It offers the Ethernet connectivity and IP services. It receives and sends RADIUS packets over UDP sockets. It builds or parses RADIUS messages, handles the RADIUS secret, checks or generates authentication attributes. EAP messages, transported by RADIUS payloads are forwarded to smartcards, running EAP-Servers.

- EAP servers. Each smartcard runs an EAP-server, and fully handles an EAP-TLS authentication procedure. Each component stores an unique X509 certificate and its associated RSA private key. It computes EAP responses and produces EAP requests. At the end of a successful authentication session, a MSK is calculated and delivered to the RADIUS entity

3.1. Authentication Server Design

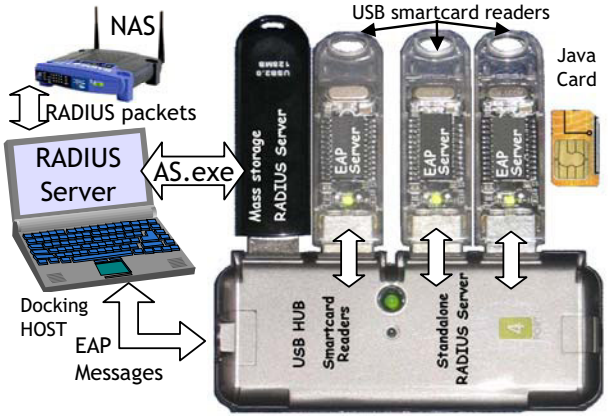

Figure 3. Practical Realization of a Smartcard Enabled RADIUS Server.

An EAP session is a set of messages associated to an unique Session-Id value, which is obtained by the concatenation of two values, the NAS-Identifier (RADIUS attribute n°32) and the Calling-Station-Id (the

client's MAC address, corresponding to RADIUS attribute n°31) as follows:

Session-Id = NAS-Identifier | Calling-Station-Id (1)

A session begins with an *EAP-Identity response* and ends with an EAP notification (either *Success* or *Failure*). It is associated to an unique smartcard. When no devices are available, the incoming RADIUS packet (starting a session) is silently discarded.

Due to smartcard slowness, each EAP message is handled by a thread that forwards EAP response to the appropriate smartcard, waits for its response, builds a RADIUS packet and finally transmits it towards the NAS.

An EAP server processes only once a particular message. The associated RADIUS packet is recorded, and sent again when an incoming duplicated RADIUS packet is detected.

AS is also in charge of session retries. If no activity is detected during a given timeout, a retransmission occurs. After a few retries the session is released, and its associated smartcard is ready for new allocations.

Figure 3 shows a plug and play realization of a *smartcard enabled RADIUS server*. Several USB smartcard readers, equipped with EAP servers, are plugged to an USB hub. A mass storage device stores the AS code. The system works in a standalone way, and is used by the docking host without any previous set-up.

3.2. EAP-Server design

We introduced in [10] the *OpenEapSmartcard* platform, an open Javacard environment, that realizes EAP clients in standard JC2.2 javacards [11]. Its code sources may be freely downloaded through the WEB. This platform has been slightly enhanced in order to support EAP servers, and includes four main components:

1- *The Engine object* manages the APDUs interface (a set of ISO 7816 [12] commands) with the AS software. The two main offered services are input/output of EAP messages, and MSK key reading. It is also in charge of EAP messages segmentation and reassembly; EAP-TLS messages maximum size is about to 1300 bytes, that is less than the limit length of Ethernet frames (around 1500 bytes). These messages are split in ISO 7816 payloads, whose maximum size is 255 bytes for input data and 256 for output data. As mentioned above the *EAP-Identity.response* message resets the EAP-TLS state machine.

2- The *Credential objec*t stores all credentials required by EAP-TLS method. This includes the *Certification Authority* (CA) certificate, the server certificate and its private key. An EAP-TLS method is initialized with appropriate credentials every time when an *EAP-Identity.response* message is received.

3- The *Authentication Interface object* describes all services fulfilled by EAP-TLS methods. Basic procedures are method initialization, packet processing and MSK key downloading.

4- The *EAP-TLS object* is in charge of packets processing as specified in [9]; since TLS packets size may exceed the Ethernet frame capacity, EAP-TLS supports internal segmentation and reassembly mechanisms. It includes an ASN.1 parser for certificates checking and provides all cryptographic facilities (*Pseudo Random Function*, HMAC, RC4) that not are available through standard javacard APIs.

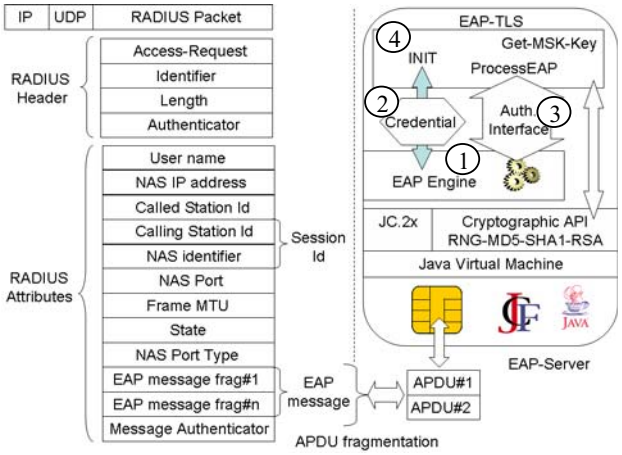

Figure 4. EAP Messages Encapsulation in RADIUS Packets.

4. EXPERIMENTAL RESULTS.

We have implemented EAP-Servers on commercial SIM cards, compatible with the JC 2.2 specification and offering 64 KB of E^2PROM space. The computing time consumed, by an EAP session is about 5 seconds.

In [13] we made a proposal for analysis of smartcards performances. We split the time required for EAP-TLS operations in three categories, data transfer ($T_{Transfer}$), cryptographic resources (T_{Crypto}) and others factors (T_{Other}), typically the software overhead.

$$T_{EAP\text{-}TLS} = T_{Transfer} + T_{Crypto} + T_{Other} \quad (2)$$

4.1. Data Transfer

It's the time consumed by the RADIUS application, running on a docking host, to read and write data to/from a smartcard. The experimental measurement of this parameter is easy, and is the sum of multiple delays induced by the reader crossing (if a smartcard reader is used), IO operations handled by the virtual machine, and finally reading and writing to memories such as RAM or E^2PROM.

$$T_{Transfer} = T_{Reader} + T_{IO} + T_{Memories} \quad (3)$$

Due to the small amount of embedded RAM, EAP-TLS messages, whose size are typically a few KB, are stored in the non volatile memory. A practical way to estimate the $T_{Transfer}$ value is to perform a simple test that transfers 255 bytes between HOST and smartcard non volatile memory. Figure 5 illustrates the observed results of our SIM component, with two types of readers. It clearly highlights that readers significantly influence the information throughput, between computers and smartcards. In the best case we observe a data rate of about 0,15 ms/bytes (50 Kbits/s).

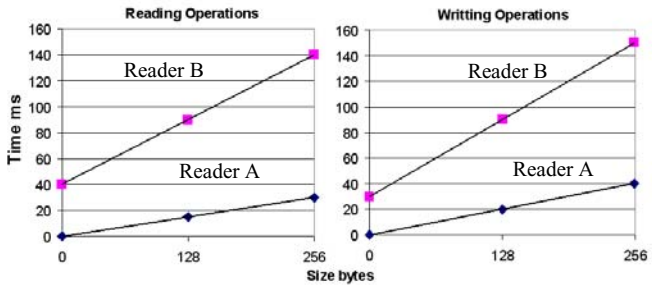

Figure 5. Reading and Writing Operations, with Two Different Readers.

During a TLS authentication, working with 1024 bits RSA keys around 2600 bytes of information (see figure 2) are exchanged between HOST and EAP-Servers. Therefore we expect a transfer duration equal to:

$$T_{Transfer} = 2600 \times 0{,}15 = 390 \text{ ms}$$

4.2 Cryptographic resources.

Cryptographic resources are protected by various countermeasures and are used via dedicated Application Programmatic Interfaces (APIs). In the EAP-server entity we need three basic facilities MD5, SHA1 and RSA.

MD5 and SHA1 are digests functions that work with blocs, whose size is 512 bits. Therefore the computing time is proportional to the size of the input message. In TLS authentication scenario [8], dealing with 1024 RSA keys sizes, approximately 266 blocs of 512 bits (about 17,000 bytes) are processed by MD5 and SHA1 functions. If we call T_{Digest} the average time for computing a bloc ($T_{MD5}/2 + T_{SHA1}/2$), these calculations cost 532 times T_{Digest}.

Figure 6 shows the variation of digest computing times with the size of the input messages. MD5 costs 1,4 ms/bloc, SHA1 needs 2,2 ms/bloc, which leads to an average value of 1,8 ms/bloc and a digest computing cost of 960 ms (532 x 1,8)

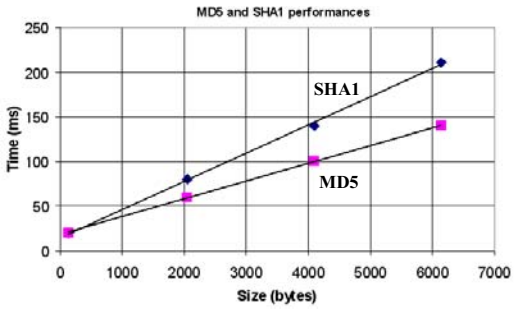

Figure 6. MD5 & SHA1 Computing Times

During an authentication session, the EAP-Server performs three RSA calculations.

- It checks the signature of client's certificate with the Certification Authority public key (public key decryption).

- It decrypts the *PreMasterSecret* value with its private key (private key decryption).

- It checks the verify message, a hash value encrypted with the client's private key (public key decryption).

So, the total time consumed by RSA operations is

$$T_{RSA} = T_{PubKD} + T_{PubKE} + T_{PrivKD} = 890 \text{ ms} \quad (4)$$

Table 1 presents the measured RSA costs of our SIM card.

Private Key Encryption	Public Key Decryption	Public Key Encryption	Private Key Decryption
750ms	70ms	60ms	760ms

Table1. RSA Computing Times.

Finally we get the following value, for the cryptographic costs of EAP-TLS:

$$T_{Crypto} = T_{RSA} + 532 \times T_{Digest} = 1850 \text{ ms} \quad (5)$$

4.3. Other resources

Table 2 presents delays associated with eleven messages exchanged during the EAP-TLS authentication, which are numbered according to figure 2. The duration of data transfer is approximately the sum of values T4,T6 and T7 (440ms), and we notice a good agreement with our previous evaluation.

T1- Rx; EAP-Identity.response	20ms
T2- Tx: EAP-TLS.request (Start)	10ms
T3- Rx: EAP-TLS.response (Client Hello)	90ms
T4- Tx: EAP-TLS.request (Server Hello fragment#1))	210ms
T5- Rx; EAP-TLS.response (EAP-TLS-ACK)	30ms
T6- Tx: EAP-TLS.request (Server Hello fragment#2)	20ms
T7- Rx: EAP-TLS.response(Client-Finished)	200ms
T8- Tx: EAP-TLS.request (Server-Finished)	3946ms
T9- Rx: EAP-TLS.response(EAP-TLS-ACK)	731ms
T10- Tx: EAP-TLS.request (EAP-Success)	20ms
T11- Rx: Get-PMK-Get	29ms

Table2. Timing Measurements of an EAP Session.

The total cost of this authentication (Σ Ti) is 5300ms, from which we deduced the T_{Other} parameter:

$$T_{Other} = T_{EAP\text{-}TLS} - T_{Transfer} - T_{Crypto} = 3050ms$$

As a conclusion EAP-Servers spend:

- 0,4s (8%) in data exchange with RADIUS server

- 1,9s (35%) in cryptographic APIs,

- 3,0s (57%) in other operations realized by Java software.

5. FUTURE WORK

In this first prototype we have chosen to implement EAP-Servers in smartcards. While the observed performances are rather slow, in comparison with classical software solutions, it's possible to improve them by introducing parallel processing of simultaneous sessions.

Several architecture could be considered in future prototypes, RADIUS smartcards, RADIUS smartcards clusters, and full RADIUS smartcards.

5.1. RADIUS smartcards

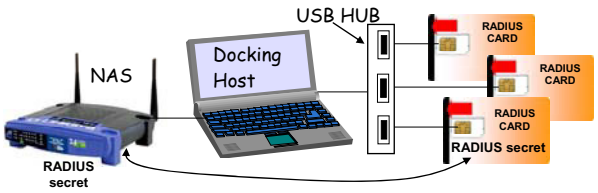

Figure 7. Docking Host and RADIUS Smartcards.

From a security point of view, the main benefit of (partial) implementation of RADIUS servers in smartcards, is to handle all calculations dealing with the RADIUS secret in a secure and trusted computing platform. For that, RADIUS *Access-Request* packets, including EAP messages, are forwarded to RADIUS smartcards, that return appropriate RADIUS packets (*Access-Challenge*, *Access-Accept*, …). The additional costs compared with EAP-servers are the following:

- MD5 and HMAC-MD5 functions used for security purposes; in a typical EAP-TLS session (dealing with key size of 1024 bits, as illustrated in figure 2) this cost is around 100 calculations of MD5 blocs (140 ms)

- About 4000 bytes of data are sent/received to/from the RADIUS card. This represents of difference of 1500 bytes by comparison with EAP-Servers, and required an estimated extra time of 225 ms (1500 x 0,15).

- RADIUS packets are parsed, checked and assembled by the embedded software. We believe that this task should consume less time than the EAP-TLS protocol computing (3,0s).

Although it seems realistic to compute a RADIUS session in less than 10s, other resources such as timeout management (packets retransmission, sessions release…) and physical accesses to Ethernet networks, will be still managed by the docking host (see figure 7).

5.2. RADIUS smartcards clusters

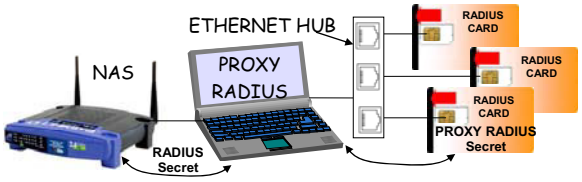

Figure 8. A Cluster of RADIUS Smartcards Based on a RADIUS Proxy

In the previous section we inquired about the pertinence and feasibility of RADIUS smartcards. Even if we foresee devices equipped with LAN interfaces such as Ethernet [1], we don't solve the performances problem induced by moderate computing capacities of smartcards. A possible and standardized solution of this issue is to implement RADIUS proxy server that will dispatch RADIUS sessions to multiple RADIUS smartcards (see figure 8). It could also manage retransmissions and timeouts and of course it supports a full Internet access. We should notice that in this case, the RADIUS secret (shared with the NAS) is stored in the docking station. RADIUS cards, logically working with the proxy, use one or several distinct secrets.

5.3. Full RADIUS Smartcard

A full RADIUS smartcard is the *Holy Grail* of highly secure and trusted RADIUS implementation. It requires smartcards equipped with LAN interfaces; also offering a high computing power, in order to handle multiple authentication sessions. This mythic component doesn't exist today, but it seems likely that emerging *System On Chip technologies* (SoC) could be used to obtain a two chip solutions.

6. CONCLUSION

In the paper we have presented an original realization of RADIUS server dealing with EAP servers. We observed interested performances, compatible with today networks constraints. Although individual authentication sessions are computed at moderate speed, the proposed architecture is scalable and could deal with realistic number of users. Future prototypes could support a subset of the RADIUS protocol, dealing with all security processing.

REFERENCES

[1] FP6-IST, "Integrated Secure Platform for Interactive Personal Devices", INSPIRED project, Information Society Technologies, 2004-2006,

[2] Tual J.P, Couchard A, Sourgen K, "USB Full Speed enabled smart cards for Consumer Electronics applications", Consumer Electronics, 2005 - ISCE 2005 - Proceedings of the Ninth International Symposium on 14-16 June 2005 Page(s):230 – 236

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

[4] Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks: Port-Based Network Access Control", IEEE Standard 802.1X, September 2001.

[5] RFC 3748, "Extensible Authentication Protocol, (EAP)", June 2004.

[6] RFC 3559, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", September 2003

[7] Institute of Electrical and Electronics Engineers ,"Draft IEEE Standard for Local and metropolitan area networks part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands", IEEE 802.16e/D12, October 2005.

[8] RFC 2246, "The TLS Protocol Version 1.0", January 1999.

[9] RFC 2716, "PPP EAP TLS Authentication Protocol". October 1999.

[10] Urien, P, Dandjinou, M, "The OpenEapSmartcard project", short paper, Applied Cryptography and Network Security 2005, ANCS 2005, Columbia University, June 7th - 10th, 2005 New York, NY, USA

[11] Chen, Z "Java Card Technology for Smart Cards: Architecture and Programmer's Guide", ADDISON-WESLEY Professional; 2000.

[12] ISO 7816, "Cards Identification - Integrated Circuit Cards with Contacts".

[13] Urien P, Dandjinou M "Designing smartcards for emerging wireless networks", Seventh Smart Card Research and Advanced Application IFIP Conference, CARDIS 2006, Tarragona-Catalonia, April 19-21 2006.