EAP-Support in smartcard

1 Abstract

This document will describe the interface to the EAP protocol in smartcards, which could store multiple identities associated to Network Access Identifiers.
Overview

All technologies derived from 802.11 specifications such as 802.11a, 802.11b, 802.11g need strong security protocols for data privacy, integrity and network access. The 802.1X [8] specification describes the risks and the protocols for the protection of the exchanged data during the network connection.

802.1X specification requires the Extensible Authentication Protocol (EAP) to be used as the framework for application dependent authentication processes with a mutual authentication between the supplicant and the authenticator. It is obvious that the role of the supplicant in this specification could partly be implemented in the smart card as an authentication processing mean. The flexibility of EAP (RFC 2284) specification does not provide a Mandatory-to-implement solution. The structure of the EAP frames allows the applications to identify the EAP type of consequently to operate the appropriate authentication.

This draft describes a standard interface to EAP implementation embedded in a smart card.

Terms

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119.

Authentication Agent: A piece of software implemented in the supplicant that processes the authentication sequence.

AS
Authentication Server

Authenticator: See the IEEE 802.1X specification for a definition of this concept.

EAP
Extensible Authentication Protocol

GSM
Global System for Mobile communications

IMSI
International Mobile Subscriber Identifier, used in GSM to identify subscribers.

NAI
Network Access Identifier

PIN
Personal Identification Number

SK
Session Key

SIM
Subscriber Identity Mobile

Supplicant: an IEEE 802.1X concept, which in the context of IEEE 802.11 represents a STA (station) seeking to attach to an IEEE 802 LAN via an IEEE 802.1X Port. See the IEEE 802.1X specification for a complete definition.

4 Identification label

802.1X specification [5] requires an authentication between the authenticator or the authentication server (AS) and the supplicant. The authentication is embedded in the Extensible Authentication Protocol (EAP) RFC2284 [1] specification. The authentication consists of a challenge response between both parties without consideration of the involved crypto-suite. Before starting the mutual authentication, the AS needs the supplicant identity to establish the session. The AS or the authenticator sends an EAP Request Identity to the supplicant that returns its system identity. A user may own several identities likely associated to the network operators.

The identification label is a pointer to a system identity (the EAP-ID value returned in the EAP-Identity.response message) stored in smartcard; it may be of various types:

1. A network SSID as described in the 802.11 standard [4].
2. A user’s identification (userid) e.g. an ASCII string. A network access identifier, NAI [6] may be used as userid.
3. A pseudonym, e.g. a friendly name.

According to the network environment, the supplicant software needs to set the appropriate identity and verifies if the smart card is able to mirror the authenticator.

If the smart card is not able to process the authentication related to the identity then any setting process is rejected by the NAK code.

The subsequent sections give the description of the methods used by a supplicant for processing an 802.1X authentication using the smart card.

Annex one provides a reference implementation example for a SIM based authentication. Annex two provides a reference implementation example for a MD5 based authentication. Annex three provides a reference implementation for a TLS based authentication.

5 Identification Label Coding Rules

The Get-Next-Identity section didn’t define the coding rules of the identification label. This section describes the structure and the architecture of the userid.

A userid consists of 2 fields separated by the Internet symbol "@". The right hand side of the "@" symbol is the userid realms while the left hand side is an application dependent and unique identification number. EAP/SIM has defined the userid where the application
Identification is "IMSI". Other userid such as email address can be used by the application.

6 Mandatory and optional services

Mandatory services must be implemented in any smart card that claims conformance with this draft. Optional services are not required by basic authentication operations.

6.1 Add-Identity

Status: Optional.
This command and the Delete-Identity are part of the user's identity management protocols. The smart card is initially manufactured without any identification label. The personalization or the supplicant software adds in the smart card user's identification label that can be retrieved by other smart card command.

If the smart card manages pseudonyms the command does not allows setting the user pseudonyms. The smart card command only adds permanent identification label in the list.

6.2 Delete-Identity

Status: Optional
This command and the add-Identity are part of the user's identity management protocols. The smart card contains a list of one or several identification labels that can be retrieved by the supplication software. The command deletes one entry of the smart card list.

6.3 Get-Preferred-Identity

Status: Optional
The smart card contains at least one user's identity related to the user's network subscription. The supplicant software gets from the smart card the initial and preferred identification label. If the user has more than one identities the supplicant software uses the Get-Next-Identity to read all the available other user's identities. If the smart card manages pseudonyms and a pseudonym is available as preferred identity, the Get-Preferred-Identity shall return the pseudonym.

6.4 Get-Current-Identity

Status: Mandatory
The smart card contains at least one user's identity related to the user's network subscription. The supplicant software gets from the smart card its current identification label.

6.5 Get-Next-Identity

Status: Mandatory
The smart card may contain one or more user's identities according to the user's network subscriptions. The supplicant software should prompt the user's identity and a subsequent selection allows the smart card to process the appropriate EAP authentication type. The method Get-Next-Identity allows the supplicant software to read all the available user's identities.
The Get-Next-Identity method may inform the supplicant software when all user's identities have been read. Otherwise the supplicant software detects the identity list end when it gets again the first identity.

If the smart card contains a pseudonym management and the pseudonym is (are) available the Get-Next-Identity returns the appropriate pseudonym. If the pseudonym management is not supported, the smart card returns the permanent Identity according to the previous section.

6.6 Get-Profile-Data
Status: Optional
The Authentication Agent or the authenticator may request the subscriber profile information. The Get-Profile-Data returns all related information available in the smart card. This specification does not provide the detail of the subscriber profile information. The implementation of the information may be ruled but ASN.1 BER coding specification [9] or by an XML dialect [10].

6.7 Set-Identity
Status: Mandatory
Once the Identity selection is processed, the supplicant software needs to set the smart card EAP framework according to the selected user's identity. The Set-Identity sets or restarts the smart card EAP framework state machine for further processing using the EAP-Packets method.

The supplicant software can set the EAP framework using the pseudonym if available in the smart card. If the pseudonym is not available the supplicant software uses the permanent identity to set the EAP framework according to the previous section.

6.8 Process-EAP
Status: Mandatory
The EAP process is described in the RFC 2284 specification [1] and involves several EAP requests and responses packets,

1) EAP request/response Identity;
2) A suite of EAP request/response related to a particular authentication scenario; and
3) EAP success or failure.

The Set-Identity restarts the smart card EAP framework state machine for further processing using the EAP-Packets method.

The smart card receives the RFC 2284 frames. It retrieves the appropriate EAP authentication type in the frame and the identifier. The smart card maintains the EAP state machine and returns an EAP NAK packet if the state sequence is broken. Any EAP request is silently ignored if the state machine was not started.

The last step of the protocol retrieving the session Key from the smart card can be accomplished only if the last EAP packet received from the authentication is an EAP success packet.
6.9 Get-Session-Key (SK)

Status: Mandatory.
At the end of a successful authentication the supplicant needs to update the appropriate crypto suite (if any) using the session key. The Get-Session-Key returns to the supplicant software the key to initialize radio security protocols like TKIP, WRAP or CCMP. For obvious security reasons this service is available only if the smart card has received an EAP success packet.

6.10 Authentication-Status

At any time, the smart card may return the authentication status. This status may reveal the following situations:

1) No authentication identity has been selected.
2) Authenticating
3) Authenticated
4) Held (Authentication failure)

6.11 Multiple EAP Identity selections

Multiple EAP authentications may be processed simultaneously in the same smartcard. If this capability is supported, the following rules apply:

1) Multiple EAP Identities may be selected at the same time.
2) The supplicant software shall indicate in the Set-Identity command short identifier to be associated with the selected EAP identity.

Note: If another EAP identity was associated with the same short identity this EAP identity becomes necessarily unlinked and is no longer more possible to accessible to it unless a new set-identity command is processed (in this case the state machine is reset) or unless a different short identity has been also associated with it.

The supplicant software shall include this short identifier within the EAP-Packets, Authentication-Status and Get-Session-Key commands to inform which of the selected EAP identities the command is targeted to.

The smartcard and the supplicant software shall maintain a separate EAP state machine for each of the different selected EAP identities.

Note: the EAP state machine is associated with each EAP identity: whether two or more different short identities are associated to the same EAP identity, the results of EAP-Packets, Authentication-Status and Get-Session-Key commands do not depend on the short identifier used to refer the EAP identity. In other words, there is only one state machine for selected EAP Identity dependently of the short identities associated with it.
The authentication agent is a piece of software implemented in the supplicant that processes the authentication sequence. This component must be able to detect a smart card. If this device is not present, or if it silently discards an EAP.request message, then authentication agent must reject all incoming request messages by the NAK code.

8 ISO 7816-4 APDUs

This section of the document provides an implementation of the previous descriptions for an ISO 78176-4 compatible smart card. The section does not preclude of the transport protocol used between the smart card and the reader. Thus, this specification does not mandate-to-implement any transport protocol such as T=0 or T=1, which are not in the scope of this document. It should be noted that all values are in hex representation.

The restriction and security related descriptions are not present in the document. Annexes of this document give implementation examples.

Note: Class byte value defined in this section (‘A0’) shall be interpreted as an implementation example. Other values may be used respecting conventions defined in ISO 78176-4.

8.1 ISO 7816 Status Word

According to ISO 7816, the status word SW1,SW2 is a two bytes word, giving information about current operation either success or failure.

‘90’ ‘00’ indicates an operation success
‘98’ ‘04’ indicates one of the following events,
   - Access Condition not fulfilled, e.g. a pin code presentation is required.
   - Unsuccessful user PIN verification, at least one attempt left.
‘98’ ‘40’ indicate one of the following events
   - Unsuccessful user PIN verification, no attempt left
   - Smart card blocked
‘67’ ‘XX’
   - Incorrect parameter P3
‘6B’ ‘XX’
   - Incorrect parameter P1 or P2
‘6D’ ‘XX’
   - Unknown instruction code (INS) given in the command
‘6E’ ‘XX’
   - Wrong instruction class (CLA) given in the command
‘6F’ ‘XX’
   - Technical problem, not implemented
‘61’ ‘XX’
   - Operation result must be fetched by the ISO Get Response APDU (CLA = ‘C0’, P3= ‘XX’)
‘6C’ ‘XX’
   - Operation must be performed again, with the LE parameter value sets to ‘XX’.
‘70’ ‘00’
   - Packet silently discard.
8.2 PIN Management

Some services may require that the smartcard’s bearer presents its PIN code.

Smartcard returns the ‘98’ ‘04’ status word when it’s necessary to check the PIN code, before accessing to a particular service (see previous section).

A PIN code is typically a four digits decimal number, ASCII encoded, and ranging between ‘0000’ and ‘9999’.

8.2.1 Verify PIN
The ISO APDU Verify is used when a PIN code presentation is required.

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify</td>
<td>A0</td>
<td>20</td>
<td>00</td>
<td>00</td>
<td>08</td>
<td>00</td>
</tr>
</tbody>
</table>

Le is the PIN code length, typically height ASCII encoded bytes.

8.2.2 Change PIN
This APDU modifies the user PIN code.

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change</td>
<td>A0</td>
<td>24</td>
<td>00</td>
<td>00</td>
<td>08</td>
<td>00</td>
</tr>
</tbody>
</table>

The old PIN (8 bytes) and new PIN (8 bytes) are presented.

8.2.3 Enable PIN
This APDU enables the user PIN function.

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>A0</td>
<td>26</td>
<td>00</td>
<td>00</td>
<td>08</td>
<td>00</td>
</tr>
</tbody>
</table>

The user PIN code (8 bytes) is presented.

8.2.4 Disable PIN
This APDU disables the user PIN function.

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable</td>
<td>A0</td>
<td>28</td>
<td>00</td>
<td>00</td>
<td>08</td>
<td>00</td>
</tr>
</tbody>
</table>
8.2.5 Unblock PIN

This APDU unblocks a smart card, blocked after three wrong PIN code presentations.

```
+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
| Unblock| A0  |  2C | 00 | 00 | 10 | 08 |
+--------+-----+-----+----+----+----+----+
```

The user PIN code (8 bytes) and an unblock code (8 bytes) are presented.

8.3 Multi-Applications smart card considerations

A smart card may store several applications, each of them being identified by a set of bytes referred as the Application IDentifier (AID).

The ISO APDU Select is used when it is necessary to select an application, able to process one or more EAP authentication scenarios.

```
+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
| Select  | 00  | A4  | 04 | 00 | XX | 00 |
+--------+-----+-----+----+----+----+----+
```

Le is the AID length.

According to ISO 7816-7 AID is made of two parts:
- RID, a mandatory 5 bytes field that identifies a company or a standardization body.
- PIX, up to 11 bytes, which identifies an application.

8.4 Add-Identity

This command adds an identification label as described in the section: Identification Label Coding Rules. The smart card list is managed by the smart card. The identification label is appended as the last element of the list.

```
+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
|        | A0  |  17 | 00 | 81 | xx | 00 |
+--------+-----+-----+----+----+----+----+
```
8.5 Delete-Identity

This command deletes the identification label as described in the section: Identification Label Coding Rules. The command parameter gives the identification label to be deleted and the smart card leave the space empty.

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>17</td>
<td>00</td>
<td>82</td>
<td>xx</td>
<td>00</td>
<td></td>
</tr>
</tbody>
</table>

8.6 Get-Preferred-Identity

This command returns the user's preferred identification label as described in the section: Identification Label Coding Rules

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>17</td>
<td>00</td>
<td>02</td>
<td>00</td>
<td>XX</td>
<td></td>
</tr>
</tbody>
</table>

8.7 Get-Current-Identity

This command returns user's current identification label as described in the section: Identification Label Coding Rules.

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>17</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>XX</td>
<td></td>
</tr>
</tbody>
</table>

8.8 Get-Next-Identity

This command returns a user identification label as described in the section: Identification Label Coding Rules.

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>17</td>
<td>00</td>
<td>01</td>
<td>00</td>
<td>XX</td>
<td></td>
</tr>
</tbody>
</table>

8.9 Get-Profile-Data

The command returns the related subscriber profile information according to the application requirements and format.
### 8.10 Set-Identity

The command resets and initializes the state machine for processing the EAP Packets. The first step after this command is an EAP request identity packet. If a different EAP packet is sent to the smart card the smart card returns an EAP NAK response.

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>1A</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>YY</td>
<td></td>
</tr>
</tbody>
</table>

### 8.11 Set-Multiple-Identity

The command resets and initializes the state machine for processing the EAP Packets. The first step after this command is an EAP request identity packet. If a different EAP packet is sent to the smart card the smart card returns an EAP NAK response.

When "multiple EAP Identity selection" is supported, then the first status byte is ‘90’ and the second one indicates the short identifier (coded in 1 byte) to be associated with the selected identity.

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>16</td>
<td>00</td>
<td>80</td>
<td>XX</td>
<td>00</td>
<td></td>
</tr>
</tbody>
</table>

### 8.12 Process-EAP

The command is the method for EAP packet management. The smart card identifies the EAP packet type and processes the EAP authentication according to current state machine. The state machine sequences have to be respected and the smart card enforces the EAP sequence processing.

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>80</td>
<td>00</td>
<td>AA</td>
<td>XX</td>
<td>YY</td>
<td></td>
</tr>
</tbody>
</table>

The EAP request or response packet lengths are represented by the unknown value XX and YY. The supplicant software should set these elements in accordance with the EAP packet types.
If "multiple EAP Identity selection" is not supported, P2 (AA value) shall be set to ‘00’.

If "multiple EAP Identity selection" is supported, P2 (AA value) shall indicate the short identifier associated with the selected EAP identity to which the command is targeted. These short identifiers are coded as described in Set-Identity command.

Most EAP request packets will produce an EAP response packet from the smartcard. If no response is to be produced (e.g. packet silently discard because invalid sequence) the smartcard shall inform the client software with an alert status word.

Success and failure packets do not require any response from the EAP client. A "successfully ending of command" status word shall be send from the smartcard once a success EAP packet is processed. An alert status word shall be send from the smartcard once a failure EAP packet is received.

EAP Identity packets are independent of the authentication type and can be the same for any type of authentications. This section of the document provides the packet details. The rest of the EAP packet being authentication protocol dependent, they are detailed in the informative annex of this document.

The description of the EAP/Request/identity is detailed according to the IETF RFC 2284 [1].

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Request    |  Identifier   |          Length = 5            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 01   |                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The description of the EAP/Response/identity is detailed according to the IETF RFC 2284 [1].

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Response   |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 01   |                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        User Identity                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
Note: Command chaining and extended length

1) When an incoming EAP packet exceeds 255 bytes, the transport mechanisms for Extended APDU described in ISO/IEC 7816-3 for T=0 and T=1 may be used.
   For T=0 the APDU Command (APDU-C) is split into data strings of at most 255 bytes and transported in the Data Field of a series of consecutive APDU ENVELOPE.
   For T=1 the APDU-C is split into data strings of at most 254 bytes and transported in the Information Field of chained I-blocks. In both cases, on reception of the TPDU the smartcard has to concatenate the successive data strings in order to obtain the original APDU.

2) When an outgoing EAP packet exceeds 256 bytes, the smartcard may use the mechanisms described in ISO/IEC 7816-4, i.e. extended length field (ISO/IEC 7816-4 2002) for T=0 and T=1.
   For T=0 the APDU response (APDU-R) is split into successive data strings of at most 256 bytes by the card. The Terminal can retrieve them by a series of consecutive GET RESPONSE APDU.
   For T=1 the APDU-R is split into data strings of at most 254 bytes by the card and transported in the Information Field of chained I-blocks. On reception, the Terminal performs the concatenation of the Information Field of the successive I-blocks to get the APDU-R. The supplicant software shall then reassemble the complete EAP packet before sending it to the authenticator.

8.13 Get-Session-Key

Once the state machine has received the EAP Success packet the smartcard process is able to send the Session Key used by the 802.1X specification for the crypto-suite.

As an illustration the EAP SIM authentication [2] specifies the Session Key usage according to the system cryptographic suite.

If "multiple EAP Identity selection" is not supported, P2 (AA value) shall be set to 00.

If "multiple EAP Identity selection" is supported, P2 (AA value) shall indicate the short identifier associated with the selected EAP identity to which the command is targeted. These short identifiers are coded as described in Set-Identity Command.
8.14 Get-Current-Version

This command returns the EAP-Type protocol version and the WLAN-SCC version.

+--------+-----+-----+----+----+----+----+
| Command | Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
|        | A0  | 18  | xx | yy | 00 | 02 |
+--------+-----+-----+----+----+----+----+

P1=00, Reserved
P1 is the current EAP-Type
P2=0, gets the EAP-Type version
P2=1, gets the WLAN-SCC version

9 State Machine Sequence

9.1 Supplicant software state machine sequence

+-----------------------+   +-----------------------+
|A-Get user's identity |>>>|B-Set user's identity  |>>>
+-----------------------+   +-----------------------+

C-sends/receive EAP packets |>>>|D-Get-Session-Key       |

Transitions:
A-B : All available identities received by Get-Next-Identity commands
B-C : Set-Identity command successfully performed

9.2 Smartcard EAP framework state machine sequence

+---------------------+   +---------------------+
| Z-Identity not set  |>>>| Y-Authenticating    |>>>
+---------------------+   +---------------------+

X-Authenticated       W- Not authenticated

Transitions:

10 Security Considerations

10.1 General Considerations

As a reference implementation the previous section provides the details of the EAP authentication using the GSM SIM. This section of the document highlights the new potential risks providers of application may face by re-using deployed networks for other purposes. From the document [7] fatal flaw does exist when have physical access to the smart card.

The nature of the Internet network does no longer require getting physical access to the smart card. Worms, Trojan horses or viruses can move to the computing platforms and performs the jobs. It is important for a reference implementation to provide the relevant level of protection for the new applications but not to create other flaws.

Other consideration have been introduced in [2] to protect the smart card against crypto attack and recommends the authentication should take place in a PROTECTED ENVIRONMENT.

10.2 PEAP Consideration

Protected Extensible Authentication Protocol (PEAP) [12] is a pre-processing protocol that allows the privacy of data when processing EAP [1] protocol. EAP protocol, as defined in [1], starts by an EAP packet request/Identity. The EAP packet response Identity returns the user's identification label with no privacy being not part of [1].

PEAP protocol allows both part of the EAP packet exchange creating a session key that can be for privacy over the subsequent execution of the EAP protocol.

This implementation of EAP in the smart card shall allow performing a PEAP tunnel for privacy. Once PEAP first phase has been successfully preformed, the EAP protocol has defined shall be performed according the EAP smart card requirements.

11 Intellectual Property Right Notice

To be specify according to the author and participant.

12 Annex 1 (Informative) - EAP/SIM packet detail.

The protocol implementation is out of the scope of this document but as a reference implementation this section gives details using the SIM as specified by [3]. Other protocol can be implemented using ISO
7816-3 TPDU. This section of the document gives the APDU syntax and coding which makes the specification protocol free. The first EAP packet is the EAP Request Identity. This initial packet format complies with [1]. The smart card returns an EAP response identity according to the IMSI length and the supported version according to [2].

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>80</td>
<td>00</td>
<td>00</td>
<td>05</td>
<td>YY</td>
<td></td>
</tr>
</tbody>
</table>

The description of the EAP/Request/identity is detailed according to the IETF RFC 2284 [1]. This EAP packet doesn't respect the EAP/SIM format since it is only part of [1].

<table>
<thead>
<tr>
<th>Request</th>
<th>Identifier</th>
<th>Length = 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = 01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The second EAP Packet is the EAP request SIM start as represented in the IETF draft document [2].

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note the EAP/Response/Identity when returning the user's identity that includes the real coded IMSI in the EAP packet and not the IMSI coded for GSM network. Further information can be retrieved in [3] for the IMSI coding in the SIM during the SIM setting.

The user Identity field can contains the user's permanent pseudonym or re-authentication identity.
The description of the EAP/Request/SIM/Start is detailed according to [2] incoming SIM data where further information can be retrieved.

The third EAP Packet is the EAP request SIM Challenge as represented in the IETF draft document [2].
The description of the EAP/Request/SIM/Challenge is detailed according to [2] incoming SIM data where further information can be retrieved.

The description of the EAP/Response/SIM/Challenge is detailed according to [2] outgoing SIM data where further information can be retrieved.
The last EAP Packet is the EAP success notification as represented in the IETF RFC 2284 [2].

The first EAP packet is the EAP Request Identity. This initial packet format complies with the RFC 2284. The smart card returns an EAP response identity according to the NAI length.

The description of the EAP/Request/identity is detailed according to the IETF RFC 2284 [1].

The description of the EAP/Response/identity is detailed according to the IETF RFC 2284 [1].
The second EAP Packet is the EAP/request/MD5/challenge as represented in the IETF RFC 2284 [1].

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>80</td>
<td>00</td>
<td>00</td>
<td>XX</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>
```

The description of the EAP/Request/MD5/challenge is detailed according to the IETF RFC 2284 [1].

The third EAP Packet is the EAP success notification as represented in the IETF RFC 2284 [1].

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A0</td>
<td>80</td>
<td>00</td>
<td>00</td>
<td>04</td>
<td>00</td>
</tr>
</tbody>
</table>
```

The description of the EAP/Response/MD5/challenge is detailed according to the IETF RFC 2284 [1].
Further information can be retrieved from the IETF draft document [2].

14 Annex 3 (Informative) TLS support

14.1 Fragment maximum size.

A single TLS record may be up to 16384 octets in length, but a TLS message may span multiple TLS records, and a TLS certificate message may in principle be as long as 16MB. The group of EAP-TLS messages sent in a single round may thus be larger than the maximum RADIUS packet size of 4096 octets, or the maximum 802 LAN frame size.

The chaining and extended length mechanisms identified in this document provide enough extension to manage incoming and outgoing EAP-TLS packets. Then, authenticator shall not necessary follow a specific fragment policy regarding whether EAP-TLS is provided by the smartcard or not.

However, in order to prevent multiple segmentation and re-assembly operations, the maximum EAP message length of a no fragmented packet shall be set to 240 bytes. For a fragmented EAP message, the maximum length value shall be 240 bytes.

As defined in EAP-TLS, when the smartcard receives an EAP-Request packet with the M bit set, it MUST respond with an EAP-Response with EAP-Type=EAP-TLS and no data. This serves as a fragment ACK.

14.2 EAP/TLS messages format.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------------------------------------+--------+------------------------+
|     Code      |  Identifier   |      Length  <= 240           |
+------------------------+------------------------+------------------------+------------------------+--------+------------------------+
|   Type = 13   |     Flag      |        TLS Message Length     |
+------------------------+------------------------+------------------------+--------+------------------------+------------------------+------------------------+
|       TLS Message Length      |          TLS DATA             |
+------------------------+------------------------+------------------------+--------+------------------------+------------------------+------------------------+
|                                                               |
+------------------------+------------------------+------------------------+--------+------------------------+------------------------+------------------------+
```

Flags

```
0 1 2 3 4 5 6 7 8
+------------------------+
|L M S R R R R R |
+------------------------+
L = Length included.
M = More fragments
S = EAP-TLS start, set in an EAP-TLS Start message.
R = Reserved
```
14.3 Example of EAP/TLS Authentication

Smartcard          Authentication Server
                  <- EAP-Request/
                    Identity
                  <- EAP-Request/
                    Identity (MyID) ->
                  <- EAP-Request/
                    EAP-Type=EAP-TLS
                    (TLS Start)
EAP-Response/
EAP-Type=EAP-TLS
TLS client_hello) ->
                  <- EAP-Request/
                    EAP-Type=EAP-TLS
                    (TLS server_hello,
                     TLS certificate,
                     TLS certificate_request,
                     TLS server_hello_done)
EAP-Response/
EAP-Type=EAP-TLS
(TLS certificate,
 TLS client_key_exchange,
 TLS certificate_verify,
 TLS change_cipher_spec,
 TLS finished) ->
                  <- EAP-Request/
                    EAP-Type=EAP-TLS
                    (TLS change_cipher_spec,
                     TLS finished)
EAP-Response/
EAP-Type=EAP-TLS ->
                  <- EAP-Success

15 Annex 4 (Normative) ASN.1 BER Tag coding for the subscriber profile information
To be defined according to the EAP type.

16 Annex 5 (Informative) APDUs exchange example
This annex shows ISO 7816 (T=0) TPDUs exchanged between the smart card and the authentication agent

Select (AID=11223344556601)
Select.request: 00 A4 04 00 07 11 22 33 44 55 66 01
Select.response: 90 00

Get-Next-Identity()
Get-Next-Identity.request: A0 17 00 01 00
Get-Next-Identity.response: 6C 04
Get-Next-Identity.request: A0 17 00 01 04
Get-Next-Identity.response: 61 62 63 64 90 00
Set-Identity()
Set-Identity.request: A0 16 00 80 04 61 62 63 64
Set-Identity.response: 90 00

EAP-Packets()
EAP-Packet.request: 00 80 00 00 05 01 A5 00 05 01
EAP-Packet.response: 61 0E
GetResponse.request: A0 C0 00 00 0E
GetResponse.response: 02 A5 00 09 01 61 62 63 64 90 00

PIN code verification (0000)
EAP-Packet.request: 00 80 00 00 05 01 A5 00 05 01
EAP-Packet.response: 98 04
Verify.request: A0 20 00 00 08 30 30 30 30 FF FF FF FF
Verify.response: 90 00

17 References


[2] EAP SIM Authentication draft version 8 (NORMATIVE)


[7] "Can you Clone a GSM Smart Card (SIM)? " From Charles Brookson Chairman GSM Association Security Group


18 Author’s Addresses
Pascal Urien
ENST
46 rue Barrault
75013 Paris Phone: NA
France Email: Pascal.Urien@enst.fr

Augustin J. Farrugia
Impasse des CAMEGIERS Phone: NA
Ceyreste, 13600 France Email: afarrugia@csi.com

Max de Groot
Gemplus
Avenue du Pic de Bertagne Phone: +33 4 42 36 50 36
BP 100, 13881 Gemenos Email: max.de-groot@gemplus.com
France

Guy Pujolle
LIP6 - University Paris 6
8 rue Capitaine Scott Phone: NA
Paris 75015 France Email: Guy.Pujolle@lip6.fr

Jorge Abellan
SchlumbergerSema
50, Av Jean Jaures Phone: +33 1 46 00 59 33
Montrouge 92542 France Email: Jorge.abellan@slb.com