EAP-Support in Smartcard

1 Abstract

This document describes the interface to the EAP protocol in smartcards, which may store multiple identities associated to EAP methods and appropriate credentials.
Overview

Terms

Relationship with RFC 2284

4.1 EAP multiplexing model

4.2 EAP smartcards

Identification label

UserID Coding Rules

Mandatory and optional services

7.1 Add-Identity

7.2 Delete-Identity

7.3 Get-Preferred-Identity

7.4 Get-Current-Identity

7.5 Get-Next-Identity

7.6 Get-Profile-Data

7.7 Set-Identity

7.8 Process-EAP

7.9 Get-Session-Key (SK)

7.10 Get-802.1X-State

7.11 Reset-801.1X-State

7.12 Method Functions

7.13 Relationship with the 802.1X supplicant state machine

7.14 Multiple EAP Identity selection

Relationships with the Smartcard Interface

9.1 ISO 7816-4 APDUs

9.2 ISO 7816 Status Word

9.3 PIN Management

9.4 Multi-Applications smartcard considerations

9.5 Add-Identity

9.6 Delete-Identity

9.7 Get-Preferred-Identity

9.8 Get-Current-Identity

9.9 Get-Next-Identity

9.10 Get-Profile-Data

9.11 Set-Identity

9.12 Set-Multiple-Identity

9.13 Process-EAP

9.14 Method Functions

9.15 Get-Session-Key

9.16 Get-Current-Version

State Machine Sequence

10.1 Supplicant software state machine sequence

10.2 Smartcard EAP framework state machine sequence

Security Considerations

11.1 General Considerations

11.2 PEAP Consideration
2 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 smartcard 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 smartcard. This device is generally considered as the most secure computing platform.

3 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.

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 Identification Number

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 Relationship with RFC 2284

4.1 EAP multiplexing model

According to [14], EAP implementations conceptually consist of the three following components:

[1] Lower layer. The lower layer is responsible for transmitting and receiving EAP frames between the peer and authenticator. EAP has been run over a variety of lower layers including:
- PPP;
- Wired IEEE 802 LANs [IEEE-802.1X];
- IEEE 802.11 wireless LANs [IEEE-802.11];
- UDP (L2TP [RFC2661] and ISAKMP [PIC]);
- and TCP [PIC].

[2] EAP layer. The EAP layer receives and transmits EAP packets via the lower layer, implements duplicate detection and retransmission, and delivers and receives EAP messages to and from EAP methods.
[3] EAP method. EAP methods implement the authentication algorithms and receive and transmit EAP messages via the EAP layer. Since fragmentation support is not provided by EAP itself, this is the responsibility of EAP methods.

4.2 EAP smartcards

An EAP smartcard implements an EAP method and works in cooperation with a smartcard interface entity, that sends and receives EAP messages to/from this component. The simplest form of this interface is a software bridge that forwards EAP messages to smartcard. According to EAP methods complexity and smartcard computing capacities, protocols sub-sets, that don’t deal with security features may be computed by the smartcard interface entity.
802.1X specification [5] requires an authentication between 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 associated to corporate/personal networks or operators’ networks.

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 (as described in section 6). 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

6 UserID Coding Rules

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.

7 Mandatory and optional services
Mandatory services MUST be implemented in any smartcard that claims conformance with this draft. Optional services are not required by basic authentication operations.

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

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

7.3 Get-Preferred-Identity
Status: Optional

7.4 Get-Current-Identity
Status: Mandatory
The smartcard contains at least one user’s identity related to the user’s network subscription. The supplicant software gets from the smartcard its current identification label.

7.5 Get-Next-Identity
Status: Mandatory
The smartcard 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 smartcard 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.

7.6 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 smartcard. Details of the subscriber profile information are given in annex 4. The implementation of the information may be ruled but ASN.1 BER coding specification [9] or by an XML dialect [10].
7.7 Set-Identity  
Status: Mandatory  
Once the Identity selection is processed, the supplicant software needs to set the smartcard EAP framework according to the selected user's identity. The Set-Identity sets or restarts the smartcard EAP framework state machine for further processing using the EAP-Packets method.

7.8 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 smartcard EAP framework state machine for further processing using the EAP-Packets method.

An incoming EAP/Request/Identity restarts the smartcard EAP framework state machine for further processing using other EAP-Packets methods.

The smartcard receives the RFC 2284 frames. It retrieves the appropriate EAP authentication type in the frame and the identifier.

The smartcard maintains the EAP state machine and returns an EAP NAK packet if the state sequence is broken. In that case it restarts the AUTHENTICATING state.

Any EAP request is silently ignored if the state machine was not started.

The last step of the protocol retrieves the session Key from the smartcard.

7.9 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, or CCMP.

In an 801.1X [5] context, SK should be interpreted as the unicast key.

In an 802.11i or WPA context SK should be interpreted as the PMK (Pairwise Master Key).

7.10 Get-802.1X-State.  
Status: Optional.
This command returns the current smartcard 802.1X state:

1) IDENTITY-NOT-SET, no authentication identity has been selected.
2) AUTHENTICATING.
3) AUTHENTICATED, authentication success, AUTHENTICATING state restarted.
4) NOT-AUTHENTICATED, AUTHENTICATING state restarted.

7.11 Reset-801.1X-State.
Status: Optional.
This command forces the EAP smartcard in the NOT-AUTHENTICATED state. See section -Relationship with the 802.1X supplicant state machine-.

7.12 Method Functions
Status: Optional.
EAP smartcards that are not able to completely process an EAP method MAY support some essential security procedures, like for example,

- X.509 Certificate storage
- Random generator
- Private key encryption
- Private key decryption
- Public key encryption
- Public key decryption
- Symmetric key encryption
- Symmetric key decryption

7.13 Relationship with the 802.1X supplicant state machine

The supplicant state machine, as described in 802.1x standard is split between the terminal and the smartcard. The smartcard only implements the AUTHENTICATING state.

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. It’s optionally forwarded to the EAP smartcard.

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.

7.14 Multiple EAP Identity selection

Status: Optional.

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 the short identifier to be associated with the selected EAP identity.

The supplicant software shall include this short identifier when necessary, in order to inform which of the selected EAP identities the command is targeted to.

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

8 Relationships with the Smartcard Interface.

The Smartcard Interface is a piece of software implemented in the supplicant that processes the authentication method. This component must be able to detect a smartcard. If this device is not present, or if it silently discards an EAP.request message, then Smartcard Interface MUST reject all incoming request messages by the NAK code.

9 ISO 7816-4 APDUs

This section of the document provides an implementation of the previous descriptions for an ISO 78176-4 compatible smartcard. The section does not preclude of the transport protocol used between the smartcard 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 noticed 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.

9.1 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' indicates one of the following events
- Unsuccessful user PIN verification, no attempt left
- Smartcard blocked
9F' xx' indicates that xx bytes (mod 256) are ready for reading.
- The FETCH command (A0120000xx) is used for reading xx bytes
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 discarded.
70' 01'
- Authentication failure

9.2 Segmentation/Reassembly rules

9.2.1 Segmentation
When a command transfers a payload, whose size is greater than 255 bytes, the less significant bit of the P1 byte is used as a More flag.

- This bit is equal to zero for a non-fragmented payload or a last fragment (More = 0 = false).
- This bit is set to one (More = 1 = true) for a payload fragment.

See annexes for examples.

9.2.2 Reassembly

- When a command reads less than 256 bytes, or in the last bloc case, the returned payload ends by the 9000 Status Word.
- When a command returns more than 256 bytes, each payload bloc (excepted the last one) ends by the 9Fxx Status Word, in which xx indicates the length of the next bloc. The FETCH (INS=12) command (A0120000xx) is used to read the next bloc.
- See annexes for examples.

9.3 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/eight digits decimal number, ASCII encoded, and ranging between '0000' and '9999'.

9.3.1 Verify PIN

```
+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
| Verify | A0  |  20 | 00 | 00 | 08 | 00 |
```

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

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

9.3.2 Change PIN

This APDU modifies the user PIN code.

```
+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
| Change | A0  |  24 | 00 | 00 | 10 | 00 |
```
The old PIN (8 bytes) and new PIN (8 bytes) are presented.

9.3.3 Enable PIN

This APDU enables the user PIN function.

+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
| Enable | A0  |  26 | 00 | 00 | 08 | 00 |
+--------+-----+-----+----+----+----+----+

The user PIN code (8 bytes) is presented.

9.3.4 Disable PIN

This APDU disables the user PIN function.

+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
| Disable| A0  |  28 | 00 | 00 | 08 | 00 |
+--------+-----+-----+----+----+----+----+

The user PIN code is presented.

9.3.5 Unblock PIN

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

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

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

9.4 Multi-Applications smartcard considerations

A smartcard 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’s necessary to select an application, able to process one or more EAP authentication scenari.

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

Lc 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.

9.5 Add-Identity

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

Identity coding guidelines are not yet specified.

+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
|        | A0  | 17  | 00 | 81 | xx | 00 |
+--------+-----+-----+----+----+----+----+

9.6 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.

+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
|        | A0  | 17  | 00 | 82 | xx | 00 |
+--------+-----+-----+----+----+----+----+

9.7 Get-Preferred-Identity

This command returns the user’s preferred identification label.

+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
|        | A0  | 17  | 00 | 02 | 00 | XX |
+--------+-----+-----+----+----+----+----+

9.8 Get-Current-Identity

This command returns user’s current identification label.

+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
|        | A0  | 18  | 00 | AA | 00 | XX |
+--------+-----+-----+----+----+----+----+

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.

9.9 Get-Next-Identity

This command returns an user’s identification label.

<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>

9.10 Get-Profile-Data

The command returns the related subscriber profile information according to the application requirements and format. Profile coding rules are defined in annex 4.

<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>AA</td>
<td>00</td>
<td>YY</td>
<td></td>
</tr>
</tbody>
</table>

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

9.11 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 smartcard the smartcard 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>16</td>
<td>00</td>
<td>80</td>
<td>XX</td>
<td>00</td>
<td></td>
</tr>
</tbody>
</table>

9.12 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 smartcard the smartcard 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 one byte) to be associated with the selected identity.

9.13 Process-EAP

The command is the method for EAP packet management. The smartcard 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 smartcard 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>
</table>
+A0      | 80    | 00  | AA | XX | YY |

+----------------------------------+
<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>
</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.

Urien & All  Informational - Expires February 2005 17
EAP-Support in Smartcard  August 2004

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 (‘7000’).

Success and failure packets do not require any response from the EAP client. A "successfully ending of command (90 00)" Status Word shall be send from the smartcard once a success EAP packet is processed.

An alert status word (‘7000’) MAY be sent from the smartcard once a failure EAP packet is received.

EAP Identity packets are independent of the authentication type; 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].
The description of the EAP/Response/identity is detailed according to the IETF RFC 2284.[1]

9.14 Method Functions.

EAP smartcards that are not able to process a specific full EAP method may support some essential security procedures.

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.

xx is the length of the input value.

yy is the length of the returned value.

P1 identifies a particular function, and is organized according to the following scheme:

| 0 | X509 | Certificate reading |
| X509 | Certificate reading |
b0 reserved (More bit)

9.15 Get-Session-Key

Once the state machine has received the EAP Success packet the Smartcard Interface is able to send the Session Key used by the 802.1X or the 802.11i specification for the crypto-suite.

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

```
+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
|        | A0  | A6  | 00 | AA | 00 | 40 |
+--------+-----+-----+----+----+----+----+
```

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.

9.16 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

9.17 Get-802.1X-State

This command returns the current smartcard state.

```
+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
|        | A0  |  19 | 00 | AA | 00 | 01 |
+--------+-----+-----+----+----+----+----+
```

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.

Returned values:
01 IDENTITY-NOT-SET, EAP messages silently discarded.
02 AUTHENTICATING, Authentication in progress.
03 AUTHENTICATED, waiting for an EAP/Request
04 NOT-AUTHENTICATED, waiting for an EAP/Request

9.18 Reset-802.1X-State

This command forces the EAP smartcard to the 802.1X AUTHENTICATING state

+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
|        | A0  |  19 | 10 | AA | 00 | 01 |

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.

Returned values:
01 No Identity set, EAP messages are silently discarded.
04 NOT-AUTHENTICATED, waiting for an EAP/Request
### 9.19 Commands summary.

<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>Process-EAP</td>
<td>A0</td>
<td>80</td>
<td>00</td>
<td>ii</td>
<td>xx</td>
<td>yy</td>
</tr>
<tr>
<td>Method-FCT</td>
<td>A0</td>
<td>60</td>
<td>zz</td>
<td>ii</td>
<td>xx</td>
<td>yy</td>
</tr>
<tr>
<td>Get-802.1X-State</td>
<td>A0</td>
<td>19</td>
<td>00</td>
<td>ii</td>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>Reset-802.1X-State</td>
<td>A0</td>
<td>19</td>
<td>10</td>
<td>ii</td>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>Get-Session-Key</td>
<td>A0</td>
<td>A6</td>
<td>00</td>
<td>ii</td>
<td>00</td>
<td>xx</td>
</tr>
<tr>
<td>Get-Profile-Data</td>
<td>A0</td>
<td>1A</td>
<td>00</td>
<td>ii</td>
<td>00</td>
<td>yy</td>
</tr>
<tr>
<td>Get-Current-Identity</td>
<td>A0</td>
<td>18</td>
<td>00</td>
<td>ii</td>
<td>00</td>
<td>yy</td>
</tr>
<tr>
<td>Get-Next-Identity</td>
<td>A0</td>
<td>17</td>
<td>00</td>
<td>01</td>
<td>00</td>
<td>yy</td>
</tr>
<tr>
<td>Get-Preferred-Identity</td>
<td>A0</td>
<td>17</td>
<td>00</td>
<td>02</td>
<td>00</td>
<td>yy</td>
</tr>
<tr>
<td>Set-Identity</td>
<td>A0</td>
<td>16</td>
<td>00</td>
<td>80</td>
<td>xx</td>
<td>00</td>
</tr>
<tr>
<td>Set-Multiple-Identity</td>
<td>A0</td>
<td>16</td>
<td>00</td>
<td>83</td>
<td>xx</td>
<td>00</td>
</tr>
<tr>
<td>Add-Identity</td>
<td>A0</td>
<td>17</td>
<td>00</td>
<td>81</td>
<td>xx</td>
<td>00</td>
</tr>
<tr>
<td>Delete-Identity</td>
<td>A0</td>
<td>17</td>
<td>00</td>
<td>82</td>
<td>xx</td>
<td>00</td>
</tr>
<tr>
<td>Get-Current-Version</td>
<td>A0</td>
<td>18</td>
<td>xx</td>
<td>yy</td>
<td>00</td>
<td>02</td>
</tr>
<tr>
<td>Verify-PIN</td>
<td>A0</td>
<td>20</td>
<td>00</td>
<td>00</td>
<td>08</td>
<td>00</td>
</tr>
<tr>
<td>Change-PIN</td>
<td>A0</td>
<td>24</td>
<td>00</td>
<td>00</td>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>Enable-PIN</td>
<td>A0</td>
<td>26</td>
<td>00</td>
<td>00</td>
<td>08</td>
<td>00</td>
</tr>
<tr>
<td>Disable-PIN</td>
<td>A0</td>
<td>28</td>
<td>00</td>
<td>00</td>
<td>08</td>
<td>00</td>
</tr>
<tr>
<td>Unblock-PIN</td>
<td>A0</td>
<td>2C</td>
<td>00</td>
<td>00</td>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>Select-AID</td>
<td>A0</td>
<td>A4</td>
<td>04</td>
<td>00</td>
<td>xx</td>
<td>00</td>
</tr>
<tr>
<td>Get-Response</td>
<td>A0</td>
<td>C0</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>xx</td>
</tr>
<tr>
<td>FETCH</td>
<td>A0</td>
<td>12</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>xx</td>
</tr>
</tbody>
</table>
10 State Machine Sequence

10.1 Supplicant software state machine sequence

+-----------------------+   +-----------------------+
| A-Get user’s identity |-->| B-Set user’s identity |>>>|
+-----------------------+   +-----------------------+  |
                  +--------------------+   +-------------------+  |
| C-send/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
C-D : Successful ending of EAP-Packets command with no outgoing packet (Status word of the command equals ‘9000’). This can be also detected by ‘authenticated’ status following the Authentication-Status command.
D-C : An incoming EAP/Request packet

10.2 Smartcard EAP framework state machine sequence

+------------------|   +------------------|
| Z-IDENTITY-NOT-SET |-->| W-NOT-AUTHENTICATED |<--|
+------------------|   +------------------|  |
                  +--------------------+   +-------------------+  |
| Y-AUTHENTICATING |<--| X-AUTHENTICATED |  |
+------------------+   +-------------------+  |

Transitions:
Z-W : An available identity successfully set
Y-X : Authentication Successful.
X-Y : First Authentication Request received.
W-Y : First Authentication Request received

11 Security Considerations

11.1 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 smartcard.

The nature of the Internet network does no longer require getting physical access to the smartcard. 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 smartcard against crypto attack and recommends the authentication should take place in a PROTECTED ENVIRONMENT.

11.2 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 smartcard MAY allow performing a PEAP tunnel for privacy. Once PEAP first phase has been successfully preformed, the EAP protocol (or other protocol) has defined shall be performed according the EAP smartcard requirements.

12 Intellectual Property Right Notice

To be specify according to the Author and Participants.
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 smartcard returns an EAP response identity according to the IMSI length and the supported version according to [2].

+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
|        | A0  | 80  | 00 | 00 | 05 | YY |
+--------+-----+-----+----+----+----+----+

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].

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 the EAP/Response/Identity when returning the user’s identity that includes the IMSI 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.

Urien & All    Informational - Expires February 2005     25
EAP-Support in Smartcard        August 2004

The user Identity field can contain the user’s permanent pseudonym or re-authentication identity.

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

+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
|        | A0  | 80  | 00 | 00 | XX | YY |
+--------+-----+-----+----+----+----+----+
The description of the EAP/Request/SIM/Start is detailed according to [2] incoming SIM data where further information can be retrieved.

```
<table>
<thead>
<tr>
<th>Request</th>
<th>Identifier</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = 18</td>
<td>Subtype = 10</td>
<td>Reserved</td>
</tr>
<tr>
<td>AT_PERM..._REQ</td>
<td>Length = 1</td>
<td>Reserved</td>
</tr>
<tr>
<td>AT_FULL..._RES</td>
<td>Length = 1</td>
<td>Reserved</td>
</tr>
<tr>
<td>AT_ANY_ID_REQ</td>
<td>Length = 1</td>
<td>Reserved</td>
</tr>
<tr>
<td>AT_VERSION_L...</td>
<td>Length</td>
<td>Actual Version List Length</td>
</tr>
<tr>
<td>Supported version 1</td>
<td>Supported version 2</td>
<td></td>
</tr>
<tr>
<td>Supported version 3</td>
<td>Padding</td>
<td></td>
</tr>
</tbody>
</table>
```

The description of the EAP/Response/SIM/Start is detailed according to [2] outgoing SIM data where further information can be retrieved.

```
<table>
<thead>
<tr>
<th>Response</th>
<th>Identifier</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = 18</td>
<td>Subtype = 10</td>
<td>Reserved</td>
</tr>
<tr>
<td>AT_NONCE_MT</td>
<td>Length = 5</td>
<td>Reserved</td>
</tr>
<tr>
<td>NONCE_MT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Urien & All Informational - Expires February 2005

EAP-Support in Smartcard August 2004

```
<table>
<thead>
<tr>
<th>AT_SELECTED</th>
<th>Length = 1</th>
<th>Select Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT_IDENTITY</td>
<td>Length</td>
<td>Actual Identity Length</td>
</tr>
<tr>
<td>User Identity (Optional)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

The description of the EAP/Response/SIM/Start is detailed according to [2] outgoing 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].

---

<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>04</td>
<td>00</td>
<td></td>
</tr>
</tbody>
</table>

---

14 Annex 2 (Informative) - EAP/MD5 packet details

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

---

<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].

---

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

---

<table>
<thead>
<tr>
<th>Response</th>
<th>Identifier</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = 01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identity Value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The second EAP Packet is the EAP/request/MD5/challenge as represented in the IETF RFC 2284 [1].

```
+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
     | A0  | 80  | 00 | 00 | XX | 16 |
+--------+-----+-----+----+----+----+----+
```

The description of the EAP/Request/MD5/challenge 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              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 04   |                                               |
|---------------+---------------+     MD5-Challenge.Value     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The description of the EAP/Response/MD5/challenge 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 = 16          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 04   |  Type_Size=10 |                               |
|---------------+---------------+---------------+     MD5 Digest Value     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

```
+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
     | A0  | 80  | 00 | 00 | 04 | 00 |
+--------+-----+-----+----+----+----+----+
```

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Success    |  Identifier   |          Length = 04          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Further information can be retrieved from the IETF draft document [2].

15 Annex 3 (Informative) TLS support.

EAP-TLS smartcards securely store at least the following items
- Client X509 certificate
- Client Private RSA Key
- Certification Authority Public Key

According to the smartcard computing capacities, two modes are defined in this draft (see details in annex 6)

- Mode 1: smartcard supports five functions:
  - Public Key Encryption, with the server public key
  - Private Key Encryption, with the client private key
  - Public Key decryption, with the Certification Authority (CA) public key
  - Reading of the client’s certificate
  - Random Number Generator

- Mode 2: smartcard fully processes the EAP-TLS protocol

15.1 Unix Time issue.

As mentioned in [15] TLS RFC the client hello message includes a 32 byte random number, whose first 4 bytes are interpreted as the Unix Time. As smartcard is not able to maintain a clock, this parameter

Urien & All Informational - Expires February 2005 30
EAP-Support in Smartcard August 2004

MUST be added to the EAP-TLS Start message by the Smartcard Interface.

```
+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
|        | A0  | 80  | 00 | 00 | 0A | YY |
+--------+-----+-----+----+----+----+----+
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=01 | Identifier | Length = 6 |
+-----------------------------------------------
| Type = 13 | Flag=20 | Unix Time |
+-----------------------------------------------
| Unix Time |
+---------------------------
```

15.2 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 no fragmented packets
issued by smartcard SHALL be set to an adapted value.

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.

15.3 EAP/TLS messages format.

```
+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+
|     Code       |  Identifier    |      Length   |
|----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+
| Type = 13      |     Flag       |    TLS Message|
|                |                   |    Length     |
|                |                   |    DATA       |
```

Urrien & All Informational - Expires February 2005 31

EAP-Support in Smartcard August 2004

Flags

```
0 1 2 3 4 5 6 7
+-------------------+
| L M S R R R R R R |
```

L = Length included.
M = More fragments
S = EAP-TLS start, set in an EAP-TLS Start message.
R = Reserved

15.4 Example of EAP/TLS Authentication

```
Smartcard                   Authentication Server
<- EAP-Request/
   Identity
EAP-Response/
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)
   (Fragment 1: L, M bits set)
EAP-Response/
EAP-Type=EAP-TLS ->
<- PPP EAP-Request/
   EAP-Type=EAP-TLS
```
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,

Urien & All Informational - Expires February 2005 32

EAP-Support in Smartcard August 2004

EAP-Response/
EAP-Type=EAP-TLS ->

<- EAP-Success

16 Annex 4 (Normative) ASN.1 BER Tag coding for the subscriber profile information

The subscriber profile is a collection of data associated to every identity. It can be used by the operating system of a wireless terminal in order to get information about user credentials.

Various information (photos...) MAY be also available.

16.1 ASN.1 Subscriber Profile Encoding

16.1.1 EapID

EapID ::= OCTET STRING

The EAP-ID associated to the current identity.

16.1.2 EapType

EapType ::= INTEGER

The EAP type associated to the current identity.

16.1.3 Version

Version ::= INTEGER

The protocol version associated to an EAP type.

16.1.4 User Credential

UserCredential ::= SEQUENCE OF CredentialObject

CredentialObject ::= SEQUENCE {
ObjectValue SubscriberInformation
}

SubscriberInformation ::= CHOICE {
SSIDList [0] IMPLICIT SEQUENCE OF {

...}
SSIDName OCTET STRING

SubscriberCertificate [1] IMPLICIT SEQUENCE OF {
Certificate X509Certificate
}

Certificate X509Certificate
}

UserData [3] IMPLICIT SEQUENCE OF {
  { SubscriberFile UserFile
  }
}

UserFile SEQUENCE OF {
Name OCTET STRING,
Value BIT STRING Value
}

X509Certificate an ASN.1 definition, as described in [13].

16.1.5 UserProfile

UserProfile ::= SEQUENCE {
  ThisEapID   EapID,
  ThisEapType EapType,
  ThisVersion Version,
  ThisCredential UserCredential
}

16.1.6 UserProfile encoding example

30 82 xx yy
  04 05 31 32 33 34 35       EapID   = 1235
  02 01 0D                   EapType = EAP-TLS
  02 01 01                   Version = 1

A1 82 xx yy
  04 05 61 62 63 64 65       SSID = abcde
  04 05 66 67 68 69 6A       SSID = fghij

A1 82 xx yy
  First  X509Certificate
  Second X509Certificate

A2 82 xx yy
  First  Root X509Certificate
  Second Root X509Certificate

A3 82 xx yy
  30 82 zz tt
  04 05 61 62 63 64 65 // Name = abcde
  03 82 zz tt
  File content
Annex 5 (Informative) APDUs exchange example

This annex shows ISO 7816 (T=0) TPDUs exchanged between the smartcard and the authentication agent

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

// Get current identity
Get-Current-Identity.request: A0 18 00 00 00
Get-Current-Identity.response 98 04
// !Pin code is requested

// PIN code verification (0000)
Verify.request: A0 20 00 00 08 30 30 30 30 FF FF FF FF
Verify.response: 90 00

// Try again
Get-Current-Identity.request: A0 18 00 00 00
Get-Current-Identity.response: 6C 04
Get-Current-Identity.request: A0 18 00 00 04
Get-Current-Identity.response: 61 62 63 64 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

// Process EAP-Packets()
EAP-Packet.request: A0 80 00 00 05 01 A5 00 05 01
EAP-Packet.response: 61 09
GetResponse.request: A0 C0 00 00 09
GetResponse.response: 02 A5 00 09 01 61 62 63 64 90 00
EAP-Packet.request: A0 80 00 00 08 01 A6 00 08 04 02 12 34
EAP-Packet.response: 61 16
GetResponse.request: A0 C0 00 00 16
GetResponse.response: 02 A6 00 16 04 10 CF A5 2D CD 63 5F 5C 6D 55 B8 09 FD B7 BB EC 5C 90 00
18 Annex 6, EAP-TLS ISO7816 APDUs Trace (T=0 Protocol)

18.1 EAP-TLS session parameters

18.1.2 CA Public Key (2048 bits)

modulus:
c8:a6:35:3d:ab:d4:67:07:ff:c6:e8:f0:03:a5:f1:
5b:00:c8:8f:36:a1:f3:88:e8:23:f1:04:c6:d4:26:
63:95:00:ade:cf:57:5d:95:2d:01:f5:7b:ae:6c:b6:
a0:70:ac:b1:2c:b2:fe:6b:7d:82:42:1d:45:3d:15:
fa:6d
publicExponent: 65537 (0x10001)

18.1.3 Server Public Key (1024 bits)

modulus:
publicExponent: 65537 (0x10001)

18.1.4 Client Private Key (1024 bits)

modulus: // N
00:de:70:0e:ff:51:fd:17:16:cf:0:51:b0:4e:ef:2e:
f3:c8:ff:83:5c:2f:f5:2b:ff:80:0f:bd:89:86:6a:
00:3e:d1:8e:42:83:62:be:c3

publicExponent: 65537 (0x10001)
privateExponent:
cf:4e:7b:ff:0f:04:b1:fc:00:99:2d:9a:76:0a:2e:
18.2 Full EAP-TLS trace (mode 2)

// TLS-START + GMT-UNIX-TIME
Tx: A0800000000A 011400060D20 3FAA2B6A
Rx: 6150

Tx: A0C0000050 // Read Client Hello
Rx: 021400500D800000004616030100410100003D03013FAA2B6A08BDD285B43D1F
3BC9715FC9F85FC453FE58F3A9E07FF397CD653922000160004005000A0009

Urien & All Informational - Expires February 2005 37
EAP-Support in Smartcard August 2004

0064006200003000600130012006301009000

// Forward Server_Hello frag#1 1396 octets, total size = 4710 octets // eap.request#15
Tx: A0800100F0011505740DC00000126616030116E50000000004616030100410100003D03013FAA2B9BCC3D
6179E2D7E78460A2596342C5014289B753209CA02A31DEB91422061240000089
2B16D27FEBD10B93DEFC224C322B69B994C1A8FB2B5BD4094861A0004000B00
05A80005A50005A23082059E30820486A003020102020A613116E50000000000
0330D0692A864886F70D010105000305231123010060A0992268993FF2C64
864886F70D0109011619706572736F6E616C2D6261736963407468617774652E
636F6D0061305F310B3009060355040613025553117301506035504A130E56
657269566768620C20496E632E313730356035504B132E436C617373203320
5075626C6963205072
Rx: 6106
Tx: A0C0000006 // Read ACK#16
Rx: 02160006D0D009000

// Transfer Server Hello frag#3 1396 octets eap.request#17
Tx: A0800100F0011705780D40696D6172792043657274696663174696F6E2041
7574667F26297479061305F310B30090603550406130255531173015060355
04A1130E56657269536676E2C20496E632E3113730356035504B132E436C61
73732032205075626CC696320507269D6172792043657274696663174696F
6E20417574686F7269749061305F310B30090603550406130255531173015060355
06035504A130E56657269536676E2C20496E632E313730356035504B132E436C61
436C6173732031205075626CC696320507269D6172792043657274696663174696F
74696F6E20417574686F726974900C43081C1310B
Rx: 9000

Rx: 9000
Tx: A0800100F0309060355040613025553117301506035504A130E5665726953
6967662C20496E632E313C30A06035504B13333436C61737320332020575626C
696320507269D6172792043657274696663174696F6E20417574686F7269
7497220D204732313A303806035504B1312386320203139338205656726953
6967662C20496E632E320F7220617574686F72697A6E6420757365206F
6E6C79311F301D06035504B1316665672695369676E205472757374204E
5776F726BB09C308199310B30090603550406130245853111300F60355040713
0842756461706573743127302506035504A131E4E
Rx: 9000

Urien & All        Informational - Expires February 2005          40
EAP-Support in Smartcard              August 2004

Rx: 9000
Tx: A0800100F0657446C6F636B2048616C6F7A617462697A746F6E73616769204666
742E311A301806035504B1311541616E757369746661E796B69666F6B3132
3030603550431324E65744C6F636B205557A6C65746920283C6173732042
292054616E75736974661E796B961664F00473045310B300906035504013
255531130166035504A130F47544204362F72706F726174696F6E311C30
1A06035504031314574205397652654727573742052626F6F740077307531
0B30090603550406130255531301606035504A130F47544204362F72706F
726174696F6E3127302506035504B131E47544520
Rx: 9000

Rx: 9000
Tx: A0800100F04379626572547275737420536F6C67574696F6E732C0496E632E31
233021060355040313147454204237962652547275737420476C6F6216C20
5266746F7400C63081C3310B300906035504061302555311341301260355040A
0A456E74727573742E6E6574313B30390635504B1323777772E656747275
53742E65742F43505320696E636F72702E20627920726275662E20286C696D69
477326CC696122E923125302306035504B131C2863292033393392045674
727573742E66747240C696D6974656431A303806035504031313456E747275
73742E657240536563757256250365727667220
Rx: 9000

Rx: 9000
Tx: A0800100F0436572649666963174696F6E20417574686F726974900B23081
AF310B3009060355040613024855311300E603550408137048756E7617279
3111300F603550407130825647610657341327320506035504A131E4E65
744C6F636B2048616C6F7A617462697A746F6E73616769204666746E2311A3018
06035504B1311541616E757369746661E796B69666F6B313630340605504
03132D4E65744C6F636B204B6F7A6A6567797A6F692028436C61737320412920
54616E7573697476616E7969661646F00C43081C1310B300906035504061302
55531173015060355040A130E5665726953696766
Rx: 9000

Tx: A0800000C42C20496E632E313C0A060355040B13333436C6173732032205075
626C6963205072696D61727920436572746966636174666F6E20417574686F
72697479201204732313A0308060355040B1313286329203393393820556752
695369676E2C20496E632E202D20466F72220617574686F72697A656420757365
206F6E6C79311F301D060355040B1316566572695369676E20547272577374204E
6574776F726B00703063E11B03090603550406130255531183018060355040A
130F47545420436F72
Rx: 6106

Tx: A0C0000006 // Transfer ACK#17
Rx: 071200060D009000

// Read Server Hello frag#4 550 octets  eap.request#18
Tx: A0800100F011802260D00706F7261174696F6E31273025060355040B131E4754
5204379626572542757537420536F6C757469666F7232C20496E362E311C01A
06035504031334754452043796265725427575737420526F6F7409E30919B31
0B300906035504061204855311300F06035504071304275646117065737431
273205060355040A131E4E65744C6F636B20496E632E20656E7374205472695369
74686F6731

Urien & All        Informational - Expires February 2005          41
EAP-Support in Smartcard              August 2004

Rx: 9000

Tx: A0800000F06B961646F005430523123010060A0992268993F22C6401191602
667231143012060A0992268993F22C6401191604656E737431153103060A0992
268993F22C64011916056261642761310F300D0603550403130663177696669
0723070312B302906355040B13224346F707972696768742028622920319393
37204D696726F736F66420436F727020E131E0C60355040B13154D696372
6F736F66420436F727021647696E6321120F060355040313184D696372
6F736F66420526F6F724017574686F726974790061305F3113011060A0992
268993F22C6401191603636F6D31193017060A0992
Rx: 9000

Tx: A080000046268993F22C64011916096D6963726F736F66743212D30206035504
0313244D6963726F736F667420526F6F72404365727469666963617465204175
74686F7269747900000000
Rx: 9F00

// Transfer Smartcard Response, eap.response#18

// 1st fragment 1594 bytes - 05D6 - Code=2 id=18
// Length=1494 Type=0D Flag=C0 Size=1825
Tx: A012000000
Rx: 021805D60D00000072116303106F10B0005E10005DE0005DB30205D7308204
BFA0302010202A61253DF0000000006300D06092A684886F70D01010505
0035023112301006A0992268993F22C640119160266723113413012060A099226
8993F22C6401191604656E737431153013060A0992268993F22C640119160562
6167261310F300D060355040313066317769669301E170D303331303630
39333635359A170D30343130353039333635395A4026312310301060A099226
8993F22C6401191602667231143012060A0992268993F22C6401191604656E73
The EAP-TLS smartcard model supports five functions:
- Public Key Encryption, with the server public key
- Private Key Encryption, with the client private key
- Public Key decryption, with the Certification Authority (CA) public key
- Reading of the client’s certificate
- Random Number Generator

In this mode the EAP-TLS smartcard interface doesn’t provide RSA functions. Furthermore all client’s parameters (RSA keys and certificate) are stored in the smartcard.
// RANDOM Number Generator
Tx: A060 0200 1C // 28 bytes
Rx: 08BDD285B43DF3BC9715FC9F85FC453E58F3A9E07FF397CD653922

// Set Server Public Key (FCT = Initialize + Public-Encrypt)
Tx: A0604800807008bcccc7013cbb15ec1281e65af4af9801ddc65cfc50cfd2ff63e
5d37792c7c3f6b766f67d0df9e8425143ba46ae85f7f7691930a39c459a22
f22b7566529795c32e7de6dfc9dced1169a346e8e825246214df022badf983b9
ccba81c44c15a1139701b6f99954c9b2ddfda1aee041e3bd6fd06cf585413c28a
e802b4670a8f30003010001

Urien & All
Informational - Expires February 2005

EAP-Support in Smartcard
August 2004

// Pre-Master Secret Encryption with the Server Public Key
// FCT = Do-Final + Public-Encrypt
Tx: A06008000300301c5a68f8b75123308e2ddbb27b63fe021e8724e7bc5c17078b3b
3f90ba00d128f80b07ad786b6de36e5f94fddfe4
Rx: 6180

Tx: 8fd83c571fe7d1e76a86405bbd95ba4b67a48f4b4d8084f4f944c1acdf1f6ac
f95f6ff11be3c8afffb4ff6da6c5477761a34c7789cb148da424116bcb1e94292a
bc8752b7fd2255574f654debd3edf89ef0f79b3ef43f7c73f158f9917a2461b2
c5d5e2a75fcbdb7f5275ad781127300e46ec61408ef2babec200f85363926301e

// Private Encrypt with Client Private Key
// FCT = Do-Final + Private-Encrypt
// (Client Certificate Verify)
Tx: A06040002249c0326e6d899fa802cc981b8e9b65f41234db8e2456e5f3dcd68
a34f25b4e72153f50e
Rx: 6180

Tx: A0C000008b
Rx: bdd2429d21dae14d9727d2f715bf30a65e61c7c6708d5c0b6035bbcc014bafbe24b
b98550af86e13b6d8371e5a922d20dd338b563b79c9af0e9110c77b468a65
1915575d348a7d29b89cc5a8d4b8aa715d53e340e6e7ad66e3e343f358b870c5
da5e61c45ee5e3f9454219f48a34cc9810a946f0c652675e3ca81aba229309b7

// Public Decrypt#1 with CA public key, first byte
// FCT = Do-Final + Index#1 + Private-Decrypt
// Checking of server certificate
Tx: A061 1B 00 01 13
Rx: 9000

// Public Decrypt#1 (with CA public key, 255 bytes)
Tx: A061A00FFAA233A6EDB4282A69E9D02D351F32FD0B97AF039C4BACD6B7ED5C1
55110EBACC3F0AD6853DEE845CC3D0E9DE8EC7514295F8541D6F409DFEB6
1A60C9AEF0BC09AD3C1A93BE546B2D9DBB8A9D90AAB5CEE35FF67512758
7301C95339B4ADE0F40C54754DAE7461966322B5772B460B7FA2F5985D49E
C522CFA7456DF2D78EDE981C4BF2ACB987BA9BDE3D16246533F0F0B0103C54
7DA547C1F03B1C2B5CDD0BD3D2ABFAFD06387235E8E49DEDC7E27E80A15B
1317A04EFC1ADBFE47547C96D57154A6EF4BFAFD40D55DF739517967703BC3A
9D34623BD28EC9186A1078130652552D5CFE1B6CDBA5197910A4C87CAD1F92F
A7EB7A0B
Rx: 6123

Tx: A0C0000023 // Certificate Hash
Rx: 3021300906052B0E03021A0500041429A563710F25832A6F692E44F9AFF36F
BE91A79000

// Read Client Certificate
Tx: A060000000 // Certificate first fragment
Rx: 308205D7308204BFA003020102020A61253DFF00000000006300D06092A8648
19 References


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


[14] RFC 2284 bis, draft-ietf-eap-rfc2284bis-08.txt

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: NA
BP 100, 13881 Gemenos Email: max.de-groot@gemplus.com
France

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

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

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described
in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the IETF’s procedures with respect to rights in IETF Documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an “AS IS” basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2004). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.