EAP Working Group
Internet Draft
Intended status: Informational

P. Urien
Telecom ParisTech
G. Pujolle
LIP6

January 9 2014

Expires: July 2014

EAP Support in Smartcard
draft-urien-eap-smartcard-26.txt

Abstract

This document describes the functional interface, based on the ISO7816 standard, to EAP methods, fully and securely executed in smart cards. This class of tamper resistant device may deliver client or server services; it can compute Root Keys from an Extended Master Session Key (EMSK).

Requirements Language

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 [RFC2119].

Status of this Memo

Copyright Notice

Table of Contents

1 Overview

All wireless LAN technologies derived from IEEE 802.11 or IEEE 802.16 specifications need strong security protocols for data privacy, integrity and network access control.

Standards such as [802.1X], [IEEE 802.16e], [IKEv2], require the Extensible Authentication Protocol (EAP) [RFC 3748] as the framework for authentication purposes, with a mutual authentication between a client (supplicant, subscriber's terminal, VPN user) and an authentication server (AS).

EAP methods MAY be implemented in smart cards.

This draft describes a standard interface to EAP methods embedded in ISO7816 smart cards. These devices are generally considered as one of the most secure computing platform. As an illustration the NIST issued a set of specifications [NIST-PIV], dealing with the Personal Identity Verification (PIV) integrated circuit card.

Annex one provides a reference example for a SIM based authentication [EAP-SIM]. Annex two gives a reference example for a MD5 based authentication. Annex three presents a reference example for a TLS based authentication [EAP-TLS]. Annex four describes the optional user's profile according to the ASN.1 [ASN.1] syntax. Annex five illustrates an MD5 authentication scenario working with an EAP smartcard. Annex six shows ISO 7816 data exchanges with EAP-TLS smartcards. Annex seven presents ISO 7816 data exchanges with EAP-AKA [EAP-AKA] smart cards.

2 Relationships with RFC 3748

2.1 EAP multiplexing model

According to [RFC 3748], EAP implementations conceptually consist of the four 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];
- IEEE 802.16e Wireless Metropolitan LANs [IEEE-802.16e];
- UDP (L2TP [L2TP] and IKEv2 [IKEv2])

2- EAP layer. The EAP layer receives and transmits EAP packets via the lower layer; it implements duplicate detection and retransmission, and delivers and receives EAP messages to and from EAP methods.

3- EAP peer and authenticator layers. Based on the Code field, the EAP layer de-multiplexes incoming EAP packets to the EAP peer and authenticator layers. Typically, an EAP implementation on a given host will support either peer or authenticator functionality.

4- EAP method. EAP methods implement the authentication algorithms and receive and transmit EAP messages. Since fragmentation support is not provided by EAP itself, this is the responsibility of EAP methods.

2.2 EAP smartcards

An EAP smart card implements one or several EAP methods, and works in cooperation with a Smartcard Interface Entity, that sends and receives EAP messages to/from this device. The simplest form of this interface is a software bridge that transparently forwards EAP messages to smart card.

```
          +-+-+-+-+-+-+                              +-+-+-+-+-+-+
          | EAP method|                              | EAP method|
          | Smartcard |                              | Smartcard |
          |   Client  |                              |   Server  |
          |  Type = X |                              |  Type = Y |
          +-+-+-+-+-+-+                              +-+-+-+-+-+-+
                !                                          !
          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+  +-+-+-+-+-+-+-+-+-+-+-+-+-+
          |  Smartcard  | EAP method|  | EAP method  | Smartcard |
          |  Interface  | Type = Y  |  |  Type = X   | Interface |
          |    Entity   |           |  |             |   Entity  |
          +-+-+-+-V-+-+-+-+-+-+-+-+-+  +-+-+-+-!-+-+-+-+-+-+-+-+-+
          |       !                 |  |       !                 |
          |  EAP  !  Peer Layer     |  |  EAP  !  Auth. Layer    |
          |       !                 |  |       !                 |
          +-+-+-+-!-+-+-+-+-+-+-+-+-+  +-+-+-+-!-+-+-+-+-+-+-+-+-+
          |       !                 |  |       !                 |
          |  EAP  ! Layer           |  |  EAP  !  Layer          |
          |       !                 |  |       !                 |
          +-+-+-+-!-+-+-+-+-+-+-+-+-+  +-+-+-+-!-+-+-+-+-+-+-+-+-+
          |       !                 |  |       !                 |
          | Lower !  Layer          |  | Lower !  Layer          |
          |       !                 |  |       !                 |
          +-+-+-+-!-+-+-+-+-+-+-+-+-+  +-+-+-+-!-+-+-+-+-+-+-+-+-+
                  !                            ! Authentication
                  !   Peer                     ! Server
                  +------------>---------------+
```

3 Overview of EAP smartcards in the IETF context.

```
   Peer-Layer Interface         EAP Smartcard                 Exported-
   (RFC 4137)                Additional Services             Parameters

   |                        |Content  |Security |Identity             |
   |                        |Mngt     |Mngt     |Mngt                 |
   |                        |         |         |                     |
   |                       +V+-+-+-+-+V+-+-+-+-+V+                    |
   |                       |    EAP Smartcard    |  SC.Get-Exp-Param  |
   |                       |        Method       |-> Channel Binding  |
   | SC.Reset,SC.Get-State |                     |-> Peer-ID          |
   |         MethodState ->|    Secure Method    |-> Server-ID        |
   |                       |      Processing     |-> Method-ID        |
   |        SC.Process-EAP |                     |                    |
   |          eapReqData ->|  Secure Storage of  | SC.Get-Session-Key |
   |                       |  Method Credentials |-> MSK              |
   |                       |                     |                    |
   |                       |     Keys caching    | SC.Get-AMSK        |
   |                       |         (EMSK)      |-> AMSK             |
                          +-+-+-+-+-+-+-+-+-+-+-+-+
```

The EAP smartcard offers four classes of services, the network interface, the content management, the security management and the identity management.

3.1 Network Interface

Network services comprise two kinds of functional interfaces, described in [RFC 4137] and in [EAP-KEY], that we call Peer-Layer and Exported-Parameters.

The interface between EAP methods and the Peer-Layer is introduced in [RFC 4137] and comprises two main procedures:

- methodState() initializes a method or gets its current state. This function is realized by two EAP-Smartcard procedures named SC.Reset-State and SC.Get-State.

- eapReqData() forwards EAP messages to methods, and optionally returns a response. The EAP smartcard uses the SC.Process-EAP command for that purpose.

Upon success, the method computes a set of values, whose use is more precisely defined in [EAP-KEY], and which are made available for other EAP layers:

- The Master Session Key (MSK) used as a shared secret, involved in cryptographic material generation. The SC.Get-Session-Key command returns this value.

- An additional key, the Extended Master Session Key (EMSK), never shared with a third party. This key is 'cached' by the Smartcard.

- Application Master Session Keys (AMSK) introduced by in [EAP-EXT] and obtained through a key distribution function (KDF) using EMSK and other values, as input parameters. This key is collected by the SC.Get-AMSK command.

- Method-ID used as a unique identifier of an EAP conversation. It's typically obtained by the concatenation of two random values generated by server and client entities. This value is obtained via the SC.Get-Exported-Parameter command.

- Server-ID corresponding to the identity, if any, of the server. For example it's the subject field of an X.509 certificate. This value is obtained via the SC.Get-Exported-Parameter command.

- Peer-ID used for the identity of the client, if any. It could be the subject field of an X.509 certificate. This value is obtained via the SC.Get-Exported-Parameter command.

- Channel Bindings used as elements of information, typically relative to the IEEE 802.1x access point (Called-Station-Id, Calling-Station-Id, NAS-Identifier, NAS IP-Address, etc.). They are optionally mirrored during an EAP session, from server to client. This value is obtained via the SC.Get-Exported-Parameter command.

3.2 Other services

An EAP-Smartcard has a physical interface with the EAP-Peer layer and produces output values as described in the previous section. However other management services are required for practical reasons:

- Content Management. It's the set of operations needed to download credentials required by a particular method (X.509 certificates, cryptographic keys ...).

- Security Management. This service manages mechanisms (PIN codes, biometric techniques ...) that restrict EAP-Smartcard use to authorized bearers.

- Identity Management. When several methods (or instances of methods) are available, this service allows selecting one of them.

3.3 Out Of Band (OOB) facilities

EAP sessions may tunnel protocols such as NAP (Network Access Protection) or NAC (Network Admission Control). It is likely that these transported messages will be encrypted and protected according to mechanisms managed by EAP methods.

OOB facilities are services that enable Smartcard Interface Entities to exchange OOB messages (such as NAP or NAC) tunneled by smart card embedded EAP methods.

4 User's Identity

The user's identity is a pointer to a tuple of values comprising:

- The EAP-ID (the parameter returned in the EAP-Response Identity message),
- The method type,
- Credentials (certificates, private keys, shared secrets...) associated to this particular type.

It may be of various types:

- A network SSID as described in the 802.11 standard [IEEE 802.11].
- A user's identifier (UserID) e.g. an ASCII string. A network access identifier, NAI [RFC 4282] MAY be used as UserID.

- A pseudonym, i.e. a friendly name.

- Etc...

5 EAP smartcard services

Mandatory services MUST be implemented in any smartcard that claims conformance with this draft.

Optional services are not required by basic authentication operations.

Secure services MAY be protected by a PIN code. Non secure services MUST be freely accessible.

5.1 Add-Identity

Status: Optional.
Security: Secure(ISSUER).
The smartcard is usually manufactured without any user's identity. The personalization software or the Identity Management software, assigns to the smartcard a user's identity that can be retrieved by other commands.

5.2 Delete-Identity

Status: Optional
Security: Secure(ISSUER)
The smartcard contains a list of one or several user's identity discovered by the Identity Management software. This command deletes one entry of this list.

5.3 Get-Preferred-Identity

Status: Optional
Security: Non Secure.
The smartcard contains at least one user's identity. The Identity Management software gets from the smartcard the initial and preferred user's identity. If the user has more than one identity, the Identity Management software uses the Get-Next-Identity to read all available identities.

5.4 Get-Current-Identity

Status: Mandatory
Security: Non Secure
The smartcard contains at least one user's identity related to the user's subscriptions. The Identity Management software gets from the smartcard its current user's identity.

5.5 Get-Next-Identity

Status: Mandatory
Security: Non Secure
The smartcard may contain one or more user's identities according to the user's subscriptions. The Identity Management software MAY prompt the user's identities and a subsequent selection allows the

smartcard to process the appropriate EAP authentication type. The

Get-Next-Identity command allows the Identity Management software to read all the available user's identities.

The Get-Next-Identity command MAY inform the Identity Management software when all user's identities have been read. Otherwise the Identity Management software detects the identity list end, when it collects again the first identity.

5.6 Set-Identity

Status: Mandatory

Security: Secure(BEARER)

Once the Identity selection is processed, the Identity Management software needs to set the smartcard EAP framework, according to the selected user's identity. The Set-Identity sets the smartcard EAP state machine to the NOT-AUTHENTICATED state.

5.7 Get-Profile-Data

Status: Optional

Security: Secure(BEARER)

The Identity Management software MAY request the subscriber's profile information. The Get-Profile-Data returns all related information available in the smartcard. Details of the subscriber's profile information are given in annex 4. The implementation of the information may be ruled by ASN.1 BER coding specification [ASN.1] or by an XML dialect [XML].

5.8 Process-EAP

Status: Mandatory

Security: Secure(BEARER)

The EAP process is described in the RFC 3748 specification 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 command 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 RFC 3748 packets. It retrieves the appropriate EAP authentication type and its associated identifier.

The smartcard maintains the EAP state machine and returns an EAP NAK packet if this state sequence is broken. In that case it reaches the NOT-AUTHENTICATED state.

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

The last step of the protocol retrieves the Session Key from the smartcard. The smartcard reaches the AUTHENTICATED state.

5.9 Process-EAP-OOB

Status: Optional.
Security: Secure (BEARER)

EAP method may tunnel Out Of Band messages used by protocols such as NAP or NAC.

According to [RFC 4282] an EAP packet includes a length field that indicates the whole packet size.

OOB data are appended to EAP packets, and their optional presence is implicitly notified by the use of the Process-EAP-OOB command.

5.10 Get-Session-Key

Status: Mandatory.
Security: Secure(BEARER)

At the end of a successful authentication the Smartcard Interface Entity needs to update the appropriate crypto suite (if any) using the master session key (MSK).

The Get-Session-Key returns MSK to the Smartcard Interface Entity.

In the 801.1X context, MSK should be interpreted as the unicast key.

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

5.11 Get-State.

Status: Optional.
Security: Secure(BEARER)

This command returns the current smartcard state:

1) IDENTITY-NOT-SET, no user's identity has been selected.
2) AUTHENTICATING, an authentication session is in progress.
3) AUTHENTICATED, last authentication session was successful.
4) NOT-AUTHENTICATED, no authentication in progress, or last authentication session failed.

5.12 Reset-State.

Status: Mandatory.
Security: Secure(BEARER)

If the current state is IDENTITY-NOT-SET, this command has no effect.

Otherwise this command forces the EAP smartcard in the AUTHENTICATING state.

5.13 Method Functions

Status: Optional.
Security: Secure(BEARER)

These facilities are dedicated to test issues and SHOULD BE forbidden in operational environments. The following services MAY be supported:

- X509 Certificate storage.
- Random generator.
- Private key encryption.
- Private key decryption.
- Public key encryption.
- Public key decryption.
- Symmetric key encryption.
- Symmetric key decryption.

5.14 Multiple EAP Identity selections

Status: Optional.
Security: Secure(BEARER)

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) Each selected EAP identity is associated with a short (one byte) identifier, returned by the Set-Identity command.

The Smartcard Interface Entity software MUST include this short identifier when necessary, in order to inform which of the selected EAP identities the command is targeted to.

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

5.15 Get-Exported-Parameters

Status: Optional.
Security: Secure(BEARER)

According to [EAP-KEY], EAP methods export a set of parameters that MAY be used by other EAP layers. In this draft, each attribute is

identified by an index, and is read thanks to the Get-Exported-Parameter(index) command.

Six indexes are defined, that are associated to the following attributes:

Index 1: Peer-ID.
The peer identity authenticated by the EAP method.

Index 2: Server-ID:
It's the optional server identity, authenticated by the EAP method.

Index 3: Method-ID.
EAP method specifications deriving keys MUST specify a temporally unique method identifier known as the Method-ID.

Index 4: Session-ID.
The Session-ID uniquely identifies an EAP session between an EAP peer (as identified by the Peer-ID) and server (as identified by the Server-ID).

Index 5: Key-Lifetime.
While EAP itself does not support key lifetime negotiation, it is possible to specify methods that do.

Index 6: Channel Bindings.
Channel Bindings include lower layer parameters that are verified for consistency between the EAP peer and server. In order to avoid introducing media dependencies, EAP methods that transport Channel Binding data MUST treat this data as opaque octets.

5.17 Get-AMSK

According to [RFC 4017] EMSK is an "additional keying material derived between the EAP client and server that are exported by the EAP method. The EMSK is at least 64 octets in length. The EMSK is not shared with the authenticator or any other third party. The EMSK is reserved for future uses that are not yet defined".

It has been suggested in [EAP-EXT] to derive Application-specific Master Session Keys (AMSKs) from EMSK. As an illustration AMSK MAY be obtained by a Key Derivation Function (KDF), such as

AMSK = KDF(EMSK, label, length)

As pointed in [HOKEY-EMSK] "Different uses for keys derived from the EMSK have been proposed. Some examples include hand off across access points in various mobile technologies, mobile IP authentication and higher layer application authentication". This document introduces Specific Root Keys (USRK), and defines a special

class of USRK, called a Domain Specific Root Key (DSRK); each DSRK

is a root key used to derive Domain Specific Usage Specific Root Keys (DSUSRK).

```
                    EMSK
                   /    \
              USRK   DSRK
                    /    \
             DSUSRK1 DSUSRK2
```

The USRK key derivation function is based on a pseudo random function (PRF) that has the following function prototype:

```
KDF = PRF(key, data), key=EMSK
```

and `DSUSRK = KDF(DSRK, key label, optional data, length)`

In [WiMAX-Forum-Stage2] the Mobile IP Root Key (MIP-RK) is generated at the EAP-Authentication Server which is collocated with the HAAA and at the EAP-Peer located in the MS.

```
MIP-RK = HMAC-SHA1(EMSK, "MIP APPLICATION ROOT KEY")
```

The Get-AMSK(index, data) is a generic command, used to compute AMSK key, (as defined in [HOKEY-EMSK], [WiMAX-Forum-Stage2]) identified by an index and optionally associated to data, needed to its calculation.

6 Client and Server facilities

EAP smartcard MAY offer two classes of services,
- Client smartcards process EAP requests and return EAP responses
- Server smartcards process EAP responses and return EAP requests

7 IEEE 802.16 services

The [IEEE 802.16] security is based on the PKM (Privacy Key Management) protocol which requires, on the user's side, an X509 certificate and a private RSA key.

[IEEE 802.16e] MAY support a version of PKM, referred as PKM-EAP, which at the end of authentication scenario, produces a MSK key, according to [RFC 3748]

An IEEE 802.16 service is a couple of credentials (X509Certificate, Private RSA Key), associated to a given identification label, and therefore working with a particular EAP method.

Two services are defined.

7.1 Get-Certificate

Status: Optional.
Security: Secure(BEARER)

This command reads the X509 certificate, associated with an identification label, which is either implicit or identified by an index.

7.2 Private-Key-Decryption

Status: Optional.
Security: Secure(BEARER)

This command decrypts a message encrypted with the client public key, according to [PKCS1].

8 Relationships with the Smartcard Interface Entity.

The Smartcard Interface Entity is a piece of software that establishes a logical bridge with smartcards. It MUST be able to detect a smartcard. If the device is not present, or if it silently discards an EAP message, then the Smartcard Interface Entity 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 ISO 7816-4 compatible smartcards. 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 hexadecimal representation.

Annexes of this document give implementation examples.

Note: The 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
'63' 'xx' indicates that a PIN code presentation is required, with xx attempts left.
'9F' 'xx' indicates that xx bytes (mod 256) are ready for reading.
- Operation result MUST be fetched by the ISO Get Response APDU (CLA = 'C0', P3= 'XX')
'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 (except for the last one) ends by the 9yxx Status Word, in which xx indicates the length of the next bloc and y MAY have any value between 1 and F. The GET (INS=C0) command (A0C00000xx) is used to read the next bloc.

- See annexes for examples.

9.3 PIN Management

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

Smartcard returns the '63' 'xx' 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 or 2A	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 (or four) 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's 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's 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's 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 identify an application.

9.5 Add-Identity

This command stores a new identity. The identity list is managed by the smartcard. The new identification label is appended as the last element of the list.

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

9.6 Delete-Identity

This command deletes an identity. 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 the Set-Identity command.

9.9 Get-Next-Identity

This command returns a user's identification label.

Command	Class	INS	P1	P2	Lc	Le
	A0	17	00	01	00	XX

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.

Command	Class	INS	P1	P2	Lc	Le
	A0	1A	00	AA	00	YY

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.

Command	Class	INS	P1	P2	Lc	Le
	A0	16	00	80	XX	00

9.12 Set-Multiple-Identity

```
+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
|        | A0  |  16 | 00 | 83 | XX | 00 |
+--------+-----+-----+----+----+----+----+
```

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 device 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

9.13.1 Standard format

The command is used for EAP packet management. The smartcard parses the EAP packet type and processes the EAP authentication according to the current state machine.

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

Lc indicates the ingoing EAP message length.
Le indicates the outgoing EAP message length, plus an optional OOB data size

The EAP request or response packets lengths are represented by the unknown value XX and YY. The Smartcard Interface Entity 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 the 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 discarded because invalid sequence) the smartcard shall inform the client software with an alert status word ('7000').

When the size of a returned EAP message is greater than the value indicated by the EAP length field, additional data should be interpreted as OOB messages.

Success and failure packets do not imply any response. A success Status Word ('9000') shall be produced by the smartcard, when 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 3748 [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 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 3748.

```
 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 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Response   |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 01   |                                               |
+-+-+-+-+-+-+-+-+                                               |
|                        User's Identity                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

9.13.2 ETSI format

```
+--------+-----+-----+----+----+----+----+
|Command |Class| INS | P1 | P2 | Lc | Le |
+--------+-----+-----+----+----+----+----+
|        | A0  | 88  | 00 | vv | XX | YY |
+--------+-----+-----+----+----+----+----+
```

The ETSI standard [TS 102 310] defines a framework for EAP support in SIM cards. EAP packets are pushed in smart cards thanks to the EAP Authenticate command.

For compatibility reasons, this command MAY be supported according to the following rules :

- The Class byte is set to 0xA0
- The INS byte is set to 0x88
- The P1 byte is set to a NULL value
- The P2 byte is not interpreted
- The P3 byte is the segment length

The [TS 102 310] standard works with implicit segmentation mechanisms. When an EAP request is greater than the maximum ISO 7816 size (255 bytes) it is fragmented in several segments whose size is less than 255. The first segment includes the packet length; therefore the transfer process is completed when the total length of exchanged data reaches this value.

Here is a brief example.

```
First Segment, A0 88 00 00 P3=FF [segment 1]
Other Segment, A0 88 00 00 P3=FF [segment k]
Last Segment,  A0 88 00 00 P3=xx [segment n]

EAP-Length = P3.1 + P3.k + P3.n
```

9.14 Process-EAP-OOB

This command has the same effects than Process-EAP, excepted that OOB data are concatenated to the incoming EAP message.

Command	Class	INS	P1	P2	Lc	Le
	A0	80	00	AA	XX	YY

Lc indicates the ingoing EAP message length plus the OOB data size. Le indicates the outgoing EAP message length plus an optional OOB data size

The EAP request or response packets lengths are represented by the unknown value XX and YY. The Smartcard Interface Entity software should set these elements in accordance with the EAP packet types.

9.15 Get-Session-Key

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

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

P1=00, EAP engine version.
P1=01, selected method version.

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.

9.17 Get-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
04 NOT-AUTHENTICATED

9.18 Reset-State

This command forces the EAP smartcard to the AUTHENTICATING state

Command	Class	INS	P1	P2	Lc	Le
	A0	19	10	AA	00	00

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:
- None

9.19 Get-Exported-Parameter

This command read an exported parameter, identified by its index

Command	Class	INS	P1	P2	Lc	Le
	A0	86	00	AA	01	yy

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 value: The value of the requested parameter.

9.20 Get-AMSK

This command reads an AMSK key, identified by its index. An optional label may be provided for this AMSK calculation.

Command	Class	INS	P1	P2	Lc	Le
	A0	88	00	AA	xx	yy

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.

The less significant bit of P1 is used as a "More" indicator, as previously defined in 10.2. Other bits of P1 (b7...b1) represent the left shifted value of an AMSK index (a value ranging between 0 and 127).

Lc gives the length (in bytes) of optional data.

Returned value: the value of the requested parameter. If no AMSK is available, the Le field is null.

9.21 Method Functions.

These facilities are available for test purposes only. They SHOULD NOT be available in operational environments.

Command	Class	INS	P1	P2	Lc	Le
Method-FCT	A0	60 or 82	zz	AA	xx	yy

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:

```
b7b6     00-Do.Final, 01-Initialize  10-More 11-Reserved
b5b4     Function index
b3b2b1   Function type
 0 X509      Certificate reading
 1 Random    Number Generator
 2 Private   key encryption
 3 Private   key decryption
 4 Public    key encryption
 5 Public    key decryption
 6 Symmetric key encryption
 7 Symmetric key decryption
b0 reserved (More bit)
```

9.22 IEEE 802.16 Services

Each EAP method MAY be associated to IEEE 802.16 services.

Command	Class	INS	P1	P2	Lc	Le
Method-FCT	A0	84	zz	AA	xx	yy

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:

```
b7b6     00-Do.Final, 01-Initialize, 10-More, 11-Reserved
b5b4     RFU (always 00)
b3b2b1   Function type
 0 X509 Certificate reading
 3 Private key decryption
```

b0 reserved (More bit)

9.23 Commands summary.

Command	Class	INS	P1	P2	Lc	Le
Process-EAP	A0	80-88	00	ii	xx	yy
Process-EAP-OOB	A0	80	00	ii	xx	yy
Method-FCT	A0	60-82	zz	ii	xx	yy
IEEE-802.16-Services	A0	84	zz	ii	xx	yy
Get-Exported-Parameter	A0	86	00	ii	01	yy
Get-AMSK	A0	88	zz	ii	xx	yy
Get-State	A0	19	00	ii	00	00
Reset-State	A0	19	10	ii	00	01
Get-Session-Key	A0	A6	00	ii	00	xx
Get-Profile-Data	A0	1A	00	ii	00	yy
Get-Current-Identity	A0	18	00	ii	00	yy
Get-Next-Identity	A0	17	00	01	00	yy
Get-Preferred-Identity	A0	17	00	02	00	yy
Set-Identity	A0	16	00	80	xx	00
Set-Multiple-Identity	A0	16	00	83	xx	00
Add-Identity	A0	17	00	81	xx	00
Delete-Identity	A0	17	00	82	xx	00
Get-Current-Version	A0	18	xx	yy	00	02
Verify-PIN	A0	20-2A	00	00	08	00
Change-PIN	A0	24	00	00	10	00
Enable-PIN	A0	26	00	00	08	00
Disable-PIN	A0	28	00	00	08	00

```
 |       Unblock-PIN      | A0  |  2C | 00 | 00 | 10 | 00 |
 +------------------------+-----+-----+----+----+----+----+
```

```
|       Select-AID       | A0  |  A4 | 04 | 00 | xx | 00 |
+------------------------+-----+-----+----+----+----+----+
|      Get-Response      | A0  |  C0 | 00 | 00 | 00 | xx |
+------------------------+-----+-----+----+----+----+----+
|          FETCH         | A0  |  12 | 00 | 00 | 00 | xx |
+------------------------+-----+-----+----+----+----+----+
```

10 Security Considerations

Smart cards are a highly effective means of enforcing security policies. They are typically carried by one party (the end user, such as an employee or customer) but are controlled by another party (the issuer, such as an enterprise or service provider). Applications running in the Smart Card are controlled by the issuer, and serve to protect the interests of the issuer.

10.1 Security Claims

Security claims expressed in this section are imported from [EAP-SC]

```
Integrity Protection:  no
Replay Protection:     no
Confidentiality:       yes (section 10.9.2)
Key Derivation:        yes (section 10.9.3)
Key Strength:          no
Dictionary Attacks:    yes (section 10.9.5)
Fast Reconnect:        no
Cryptographic Binding: yes (section 10.9.6)
Session Independence:  no
Fragmentation:         no
Channel Binding:       yes (section 10.9.7)
```

10.2 Smart Card Technology

The Smart Card consists of a microprocessor and non-volatile memory chipset enclosed in a physically tamper resistant module. This module is then embedded in a plastic card, or the module may be integrated into an alternative form factor, such as a USB device.

10.3 Tamper Resistant Storage and Execution

Smart cards provide protective measures against physical and logical attacks against the processor and non-volatile memory. This enables the secure storage of end user cryptographic keys and user credentials, and secures execution of security sensitive operations such as encryption and digital signatures.

The EAP-SC Authentication Method MUST store all secret cryptographic keys on the smart card in non-volatile memory. The EAP-SC Authentication Method MUST execute in the smart card all cryptographic functions that use stored secret cryptographic keys. The EAP-SC Authentication Method MUST NOT export any secret

cryptographic keys from the smart card.

10.4 Multi Factor Authentication

Smart cards generally require a Smart Card handler to authenticate to the Smart Card in order to access data or application functionality. This makes it possible to enforce multi factor user authentication by combining something the user has (the smart card) with something the user knows (such as PIN) or is (Biometric authentication).

The EAP Authentication Method MUST enforce the use of the user PIN or Biometric before user credentials may be accessed or used.

10.5 Random Number Generation

Smart Cards generally contain a hardware based true random number generator independent of external or internal clocks and immune to outside interferences. The quality of the hardware generator is further enhanced by logical processing to ensure excellent statistical properties; and these properties are checked regularly on-board.

The EAP Authentication Method MUST use the Smart Card Random Number Generator anywhere Random Numbers are required.

10.6 Cryptographic Capabilities

Smart cards provide certified, built-in implementation and optimized execution of common cryptographic algorithms such as AES, DES, RSA, and ECC...

The EAP Authentication Method MUST use the built-in Smart Card cryptographic capabilities for the execution of any cryptographic functionality.

10.7 Secure Provisioning

Smart cards provide a secure method of provisioning credentials, applications and trusted network information from the issuer or service provider to the end user, and managing this information after the card has been issued. Smart cards support automated personalization (including card initialization, loading of card data and printing) enabling issuance in very large numbers.

The EAP-SC Authentication method MUST implement support for pre-issuance personalization, as for example by supporting [GLOBAL PLATFORM] or similar functionality. The EAP-SC Authentication method SHOULD implement support for post-issuance card and application management.

10.8 Certification

The processes for designing and manufacturing smart cards are subject to rigorous security controls. This makes possible the certification of Smart Card functionality and applications by

standardization organizations.

The EAP-SC Authentication method MUST be implemented on a Smart Card platform that has been evaluated for security by a standards organization program such as [FIPS] or [COMMON CRITERIA].

10.9 Smart Cards and EAP Security Claims

EAP-SC enhances the security of Authentication Methods by enabling the enforcement of security policies on the End User platform. The overall security of EAP-SC is dependent on the security of the Authentication Method implemented on the Smart Card.

The following section discusses certain EAP Security Claims and how they are enhanced by Smart Card security features.

10.9.1 Mutual Authentication

Mutual authentication processes are generally based upon the use of random numbers. Smart Cards enhance the security of these processes by providing true random number generation.

10.9.2 Confidentiality

Smart Cards improve the robustness of EAP messages encryption, by providing tamper resistant storage for the encryption keys and secure execution of the encryption algorithms.

10.9.3 Key Derivation

Smart Cards improve the confidentiality of the key derivation process by providing tamper resistant storage for the master keys and secure execution of the key derivation algorithms.

10.9.4 Man-in-the-Middle Attacks

Smart Cards improve security against Trojan Horse attacks by providing a logically tamper resistant environment for the full implementation of EAP methods and secure execution of the encryption algorithms.

10.9.5 Dictionary Attacks

Smart Cards access is commonly protected via pin codes with a limited number of retries; permanent blocking of the device is enforced when the number of retries is exceeded. This mechanism provides enhanced protection against dictionary attacks aiming at discovering passwords.

10.9.6 Cryptographic Binding

Smart Cards provide tamper resistant storage for cryptographic keys and secure execution of the tunnel creation algorithms thus enhancing the cryptographic binding process.

10.9.7 Channel Binding

Smart Cards can be used as a secure out of band distribution method for channel parameters and therefore enhance the channel binding process.

10.9.8 Protection Against Rogue Networks

Smart Cards facilitate the provisioning and secure storage of information about trusted parties, such as the root certificates of trusted networks. This protects the end user against rogue networks and enables the enforcement of network roaming policies.

10.9.9 Authentication Method Security

The overall security of EAP-SC is dependent on the encapsulated EAP-SC Authentication Method. Weaknesses in the underlying method, such as weaknesses in integrity protection, replay protection or key strength, are detrimental to the overall security.

11 Intellectual Property Right Notice

To be specify according to the Author and Participants.

12 Annex 1, EAP-SIM packets details.

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 [EAP-SIM]. This section of the document gives the APDU coding.

12.1 Full Authentication

The following traces illustrate a full EAP-SIM authentication scenario, as described in annex A (tests vector) of EAP-SIM [EAP-SIM] specification

```
// select TEAPM
Tx: 00A40400 10 A0 00 00 00 30 00 02 FF FF FF FF 89 31 32 38 00
Rx: 90 00

// Verify User PIN
Tx: A0 20 00 00 04 30 30 30 30
Rx: 90 00

// Set-Identity ('sim') type=EAP-SIM
Tx: A0 16 00 80 03 73 69 6D
Rx: 90 00

// Identity request
Tx: A0 80 00 00 05 01A4 0005 01
Rx: 61 16

Tx: A0 C0 00 00 16
// Identity.response: anonymous@dot.com
Rx: 02 A4 00 16 01 61 6E 6F 6E 79 6D 6F 75 73 40 64
    6F 74 2E 63 6F 6D
    90 00

// SIM-START.request  AT-VERSION AT-PERMANENT
Tx: A0 80 00 00 14 01A6 0014 120a0000 0f02000200010000 0A010000
Rx: 61 40

Tx: A0C0 0000 40
// SIM-START.response AT-IDENTITY AT-SELECTED-VERSION AT-NOUNCE
Rx: 02 A6 00 40 0C 0A 00 00 0E 08 00 1B 31 32 34 34
    30 37 30 31 30 30 30 30 30 30 30 31 40 65 61 70
    73 69 6D 2E 66 6F 6F 00 07 05 00 00 01 02 03 04
    05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 10 01 00 01

// EAP-Request/SIM/Challenge - first fragment

Tx: A0 80 01 00 C8 01 02 01 18 12 0b 00 00 01 0d 00
    00 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e
    1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e
```

```
    2f 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e
```

```
    3f 81 05 00 00 9e 18 b0 c2 9a 65 22 63 c0 6e fb
    54 dd 00 a8 95 82 2d 00 00 55 f2 93 9b bd b1 b1
    9e a1 b4 7f c0 b3 e0 be 4c ab 2c f7 37 2d 98 e3
    02 3c 6b b9 24 15 72 3d 58 ba d6 6c e0 84 e1 01
    b6 0f 53 58 35 4b d4 21 82 78 ae a7 bf 2c ba ce
    33 10 6a ed dc 62 5b 0c 1d 5a a6 7a 41 73 9a e5
    b5 79 50 97 3f c7 ff 83 01 07 3c 6f 95 31 50 fc
    30 3e a1 52 d1 e1 0a 2d 1f 4f 52 26 da a1 ee 90
    05 47 22 52 bd b3 b7 1d 6f 0c 3a 34 90
Rx: 90 00

// EAP-Request/SIM/Challenge - second and last fragment

Tx: A0 80 00 00 50 31 6c 46 92 98 71 bd 45 cd fd bc
    a6 11 2f 07 f8 be 71 79 90 d2 5f 6d d7 f2 b7 b3
    20 bf 4d 5a 99 2e 88 03 31 d7 29 94 5a ec 75 ae
    5d 43 c8 ed a5 fe 62 33 fc ac 49 4e e6 7a 0d 50
    4d 0b 05 00 00 fe f3 24 ac 39 62 b5 9f 3b d7 82
    53 ae 4d cb 6A
Rx: 61 1C
Tx: 0C0 0000 1C

// EAP-Response/SIM/Challenge
Rx: 02 02 00 1C 12 0B 00 00 0B 05 00 00 F5 6D 64 33
    E6 8E D2 97 6A C1 19 37 FC 3D 11 54
    90 00

// EAP Success
Tx: A0 80 00 00 04 03 02 00 04
Rx: 90 00

// Reading MSK and EMSK keys.
Tx: A0 A6 00 00 80
Rx: 39 d4 5a ea f4 e3 06 01 98 3e 97 2b 6c fd 46 d1
    c3 63 77 33 65 69 0d 09 cd 44 97 6b 52 5f 47 d3
    a6 0a 98 5e 95 5c 53 b0 90 b2 e4 b7 37 19 19 6a
    40 25 42 96 8f d1 4a 88 8f 46 b9 a7 88 6e 44 88
    59 49 ea b0 ff f6 9d 52 31 5c 6c 63 4f d1 4a 7f
    0d 52 02 3d 56 f7 96 98 fa 65 96 ab ee d4 f9 3f
    bb 48 eb 53 4d 98 54 14 ce ed 0d 9a 8e d3 3c 38
    7c 9d fd ab 92 ff bd f2 40 fc ec f6 5a 2c 93 b9
    9000
```

12.2 Re-Authentication

The following traces illustrate a EAP-SIM Re-Authentication scenario, as described in annex A (tests vector) of EAP-SIM [EAP-SIM] specification

```
//Identity request
```

```
Tx: A0 80 00 00 14 01 A5 00 05 01
RX: 61 56
```

```
// PSEUDONYM

Tx: A0 C0 00 56
Rx: 02 00 00 56 01 59 32 34 66 4e 53 72 7a 38 42 50
    32 37 34 6a 4f 4a 61 46 31 37 57 66 78 49 38 59
    4f 37 51 58 30 30 70 4d 58 6b 39 58 4d 4d 56 4f
    77 37 62 72 6f 61 4e 68 54 63 7a 75 46 71 35 33
    61 45 70 4f 6b 6b 33 4c 30 64 6d 40 65 61 70 73
    69 6d 2e 66 6f
    90 00

// SIM-START.request  AT-VERSION AT-ANY-ID-REQ
Tx: A0 80 00 00 14 01A6 0014 120a0000 0f02000200010000 0D01 0000
Rx: 61 60
Tx: A0 C0 00 00 60
Rx: 02 A6 00 60 12 0A 00 00 0E 16 00 51 59 32 34 66
    4E 53 72 7A 38 42 50 32 37 34 6A 4F 4A 61 46 31
    37 57 66 78 49 38 59 4F 37 51 58 30 30 70 4D 58
    6B 39 58 4D 4D 56 4F 77 37 62 72 6F 61 4E 68 54
    63 7A 75 46 71 35 33 61 45 70 4F 6B 6B 33 4C 30
    64 6D 40 65 61 70 73 69 6D 2E 66 6F 6F 00 00 00
    90 00

// EAP-Request/SIM/Re-authentication
Tx: A0 80 00 00 A4 01 01 00 a4 12 0d 00 00 81 05 00
    00 d5 85 ac 77 86 b9 03 36 65 7c 77 b4 65 75 b9
    c4 82 1d 00 00 68 62 91 a9 d2 ab c5 8c aa 32 94
    b6 e8 5b 44 84 6c 44 e5 dc b2 de 8b 9e 80 d6 9d
    49 85 8A 5d b8 4c dc 1c 9b c9 5c 01 b9 6b 6e ca
    31 34 74 ae a6 d3 14 16 e1 9d aa 9d f7 0f 05 00
    88 41 ca 80 14 96 4d 3b 30 a4 9b cf 43 e4 d3 f1
    8e 86 29 5a 4a 2b 38 d9 6c 97 05 c2 bb b0 5c 4A
    ac e9 7d 5e af f5 64 04 6c 8b d3 0b c3 9b e5 e1
    7a ce 2b 10 a6 0b 05 00 00 48 3a 17 99 b8 3d 7c
    d3 d0 a1 e4 01 d9 ee 47 70
Rx: 61 44

Tx: A0 C0 00 00 44

// EAP-Response/SIM/Re-authentication
Rx: 02 01 00 44 12 0D 00 00 81 05 00 00 CD F7 FF A6
    5D E0 4C 02 6B 56 C8 6B 76 B1 02 EA 82 05 00 00
    B6 ED D3 82 79 E2 A1 42 3C 1A FC 5C 45 5C 7D 56
    0B 05 00 00 FA F7 6B 71 FB E2 D2 55 B9 6A 35 66
    C9 15 C6 17
    90 00

// EAP Success
```

```
Tx: A0 80 00 00 04 03 01 00 04
Rx: 90 00
```

```
// Get MSK
Tx: A0 A6 00 00 40
RX: 6263f614 973895e1 335f7e30 cff028ee
    2176f519 002c9abe 732fe0ef 00cf167c
    756d9e4c ed6d5ed6 40eb3fe3 8565ca07
    6e7fb8a8 17cfe8d9 adbce441 d47c4f5e
    90 00
```

13 Annex 2, EAP-MD5 packet details

The first EAP packet is the EAP Request Identity. This initial packet format complies with the RFC 3748. The smartcard returns an EAP response identity according to the NAI length.

```
+--------+-----+-----+----+----+----+----+
|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 [RFC 3748].

```
 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 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 [RFC 3748].

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

The second EAP Packet is the EAP/request/MD5/challenge as represented in [RFC 3748].

```
+--------+-----+-----+----+----+----+----+
|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 [RFC 3748].

```
 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 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 [RFC 3748].

```
 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 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 3748 [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 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Success    |  Identifier   |          Length = 04          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Further information can be retrieved from [RFC 3748].

14 Annex 3 - TLS support.

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

14.1 Unix Time issue.

As mentioned in [TLS] 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 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 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code=01   |  Identifier   |      Length  = 6              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 13   |     Flag=20   |          Unix Time            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Unix Time         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

14.2 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.

14.3 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 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
+-+-+-+-+-+-+-+-+
|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

14.4 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
                                (Fragment 2)

   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 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 be the operating system of a wireless terminal in order to get information about user credentials.

Various information MAY be also available.

15.1 ASN.1 Subscriber Profile Encoding

15.1.1 EapID

```
EapID ::= OCTET STRING
```

The EAP-ID associated to the current identity.

15.1.2 EapType

```
EapType ::= INTEGER
```

The EAP type associated to the current identity.

15.1.3 Version

```
Version ::= INTEGER
```

The protocol version associated to an EAP type.

15.1.4 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
},

RootCertificate [2] IMPLICIT SEQUENCE OF {
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 [PKCS#6].

15.1.5 UserProfile

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

15.1.6 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
 30 xx
  A0 0E
   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
```

16 Annex 5 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       63 03
// !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 3C 90 00
```

17 Annex 6, EAP-TLS ISO7816 APDUs Trace (T=0 Protocol)

17.1 EAP-TLS session parameters

17.1.1 CA Public Key (2048 bits)

```
modulus:
    00:a5:62:a0:41:52:9a:ec:8e:27:24:a1:0c:a2:45:
    68:e3:ed:bd:3d:64:9a:7c:c2:74:5a:e2:60:fa:ac:
    6d:0f:dd:4c:45:ce:9d:b9:74:4e:35:fd:74:cd:13:
    63:dd:dc:ce:19:25:b9:d7:06:31:13:d7:ea:1e:54:
    1a:07:36:eb:97:2f:88:19:58:c5:76:ec:f9:b3:71:
    66:fa:3a:4e:94:f9:04:98:ff:b0:7f:b0:dc:af:c3:
    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:
    af:37:ad:a2:54:83:ab:13:56:83:8e:6f:b4:3a:d3:
    63:95:00:ad:ec:57:5d:95:2d:01:f5:7b:ae:6c:b6:
    43:4b:da:2b:e1:ed:f4:ab:e1:75:27:0f:2e:06:5c:
    42:30:b4:5e:06:59:58:e4:4b:b6:0e:ba:71:d6:1c:
    a0:70:ac:b1:2c:b2:fe:6b:7d:d8:42:1d:45:9d:d5:
    4a:62:06:2e:e2:dc:88:5b:8b:72:45:ac:e1:24:ea:
    08:66:30:5f:8c:e6:52:12:37:70:04:b0:37:5c:09:
    1e:3b:d4:97:0c:9b:41:3f:86:08:d7:db:19:cb:07:
    a3:b9:cb:75:49:99:dc:20:cd:f0:db:52:19:4b:15:
    f1:6d
publicExponent: 65537 (0x10001)
```

17.1.2 Server Public Key (1024 bits)

```
modulus:
    00:bc:67:01:3c:b9:15:ec:12:81:e6:5a:4d:af:49:
    80:1d:db:6d:5c:f3:0c:fd:2f:f6:3f:5d:37:79:29:
    c7:39:1b:fd:76:6f:67:dd:0f:e9:e8:42:51:43:ba:
    46:ae:95:ff:76:91:9f:30:a3:9c:45:9a:22:f2:2b:
    75:66:52:97:95:c3:2f:ee:7d:cf:c9:dc:de:11:69:
    a3:46:ef:e8:25:24:62:14:df:02:2b:ad:f9:83:b9:
    3c:bb:a8:1c:44:c1:5a:11:39:70:1b:69:f9:95:4c:
    9b:d2:fd:fa:1a:e4:01:e3:bd:6f:d0:6c:f5:85:41:
    3c:28:ae:80:2b:46:70:a8:f3
publicExponent: 65537 (0x10001)
```

17.1.3 Client Private Key (1024 bits)

```
modulus: // N
    00:de:7d:0e:f5:1d:17:16:c0:6f:51:b0:4c:ef:2e:
    c6:ca:f4:d8:66:01:bc:7b:21:12:37:ce:dc:61:72:
    f3:c8:ff:83:5c:2f:f5:2b:f8:f0:0f:bd:89:86:6a:
    3f:c2:8b:3b:bd:c7:98:fd:4b:1d:67:8f:85:66:12:
    74:6f:64:74:d0:31:07:46:04:ba:b1:74:70:b1:fc:
    d9:42:44:f8:97:c2:74:b9:45:5c:84:15:33:ec:4a:
    cb:41:d2:6e:7c:6d:bd:bc:cd:3e:64:ff:8f:33:63:
```

```
    fe:06:55:69:96:c6:96:fa:17:db:f8:7f:eb:5b:fe:
    00:3e:d1:8e:42:83:62:be:c3
```

```
publicExponent: 65537 (0x10001)
privateExponent:
    00:9f:ad:4b:5d:d9:79:e7:a7:46:7d:6f:35:57:f7:
    cf:4e:7b:f9:0f:04:b1:fc:00:99:2d:9a:76:0a:2e:
    51:0e:71:6b:1a:6f:84:db:01:37:71:64:8b:5d:ff:
    c5:30:df:72:89:da:c5:4f:0c:68:d7:19:67:19:01:
    a7:b5:06:78:da:57:2f:2f:f6:c5:ce:75:b7:ca:9d:
    b2:f8:5a:62:27:40:b2:5c:42:f3:78:fd:42:f6:1a:
    56:44:a3:42:94:24:f6:37:53:fc:78:42:06:8a:1a:
    0b:43:cf:f8:92:60:8d:10:61:2c:ff:d3:79:ba:78:
    ed:f7:28:fb:61:dc:88:37:91
prime1: // P
    00:fc:30:c8:10:41:80:f7:f2:1a:0c:28:2b:58:a0:
    44:3e:01:13:91:66:4f:96:27:0e:c3:0a:4f:58:b5:
    73:9a:3c:7a:fa:b9:19:8f:2b:32:8b:c8:bf:6c:77:
    b3:4d:e5:71:80:e5:74:9a:76:a5:c0:41:14:81:76:
    e0:9b:46:bd:db
prime2: // Q
    00:e1:d9:6b:5e:41:2d:3e:b9:2a:a8:6b:6e:d0:fc:
    aa:b1:df:a7:4e:90:8d:11:54:7c:0d:ea:64:d5:f5:
    c1:d1:2b:02:77:b2:d2:6e:d8:93:56:ad:ee:ca:5a:
    c0:92:64:4b:b8:d8:f4:a2:8c:f0:18:17:64:51:0b:
    db:04:f3:3b:39
exponent1: // DP1
    00:db:27:a9:34:37:38:54:3f:d7:d2:e8:b5:82:77:
    03:d6:be:28:bb:1a:25:df:5e:61:bd:ac:9f:f7:7e:
    f7:ce:f8:f0:06:22:04:cc:1d:c5:f7:23:a4:f6:25:
    af:73:ea:08:10:f3:55:b9:45:92:14:d8:79:71:68:
    55:17:9b:0a:31
exponent2: // DQ1
    37:87:0e:27:d9:5c:77:6c:6d:39:85:58:74:97:7a:
    9c:4b:01:c6:86:31:b8:ce:0d:c6:1a:17:fa:a6:f6:
    a5:27:ae:ee:a1:0f:ad:e3:1f:ae:93:0a:ff:c3:7a:
    4f:43:cb:7e:42:11:3b:99:ed:39:ef:1e:61:f2:c9:
    41:99:4f:b9
coefficient: //PQ
    5f:88:21:11:1f:0d:f0:cd:56:47:4f:1f:64:81:0e:
    d1:02:eb:39:42:01:c7:e4:4b:b6:31:65:2a:fd:51:
    11:1f:cd:3a:68:d4:e8:3c:4e:47:c1:ce:76:6b:2b:
    52:bd:76:dd:71:81:76:0f:69:9a:94:c3:41:3a:2e:
    c9:47:3c:e5
```

17.2 Full EAP-TLS trace (mode 2)

```
// TLS-START + GMT-UNIX-TIME
Tx: A080000000A 011400060D20 3FAA2B6A
Rx: 6150

Tx: A0C0000050 // Read Client Hello
```

```
Rx: 021400500D8000000046160301004101000003D03013FAA2B6A08BDD285B43D1F
    3BC9715FC9F85FC453FE58F3A9E07FF397CD653922000016000400050 00A0009
```

```
    006400620003000600130012006301009000

// Forward Server_Hello frag#1 1396 octets, total size = 4710 octets
// eap.request#15

Tx: A0800100F0011505740DC00000126616030112610200004603013FAA2B9BCC3D
    6179E2D7E78460A2596342C5014289B753209CA02A31DEDB9142206124000089
    2B16D27FEBD10B93D1EFC224C322B69B994C1A8FB2B5BD4094861A0004000B00
    05A80005A50005A23082059E30820486A003020102020A613116E50000000000
    03300D06092A864886F70D010105050030523112301006 0A0992268993F22C64
    011916026672311430120 60A0992268993F22C6401191604656E737431153013
    060A0992268993F22C6401191605626164726 1310F300D060355040313066361
    77696669301E170D303331303032313532333 1345A
Rx: 9000

Tx: A0800100F0170D303531303031313532333 1345A3066310B3009060355040613
    024652311630140603550408130D696C652064652066726 16E6365310E300C06
    035504071305706172697 3310D300B060355040A1304656E7374310F300D0603
    55040B1306696E66726573310F300D060355040313066 16B6B61723130819F30
    0D06092A864886F70D010101050003818D0030818902818100BC67013CB915EC
    1281E65A4DAF49801DDB6D5CF30CFD2FF63F5D377929C7391BFD766F67DD0FE9
    E8425143BA46AE95FF76919F30A39C459A22F22B7566529795C32FEE7DCFC9DC
    DE1169A346EFE825246214DF022BADF983B93CBBA8
Rx: 9000

Tx: A0800100F01C44C15A1139701B69F9954C9BD2FDFA1AE401E3BD6FD06CF58541
    3C28AE802B4670A8F30203010001A38202E4308202E0300B0603551D0F040403
    0205A0304406092A864886F70D01090F04373035300E06082A864886F70D0302
    02020080300E06082A864886F70D030402020080300706052B0E030207300A06
    082A864886F70D030730130603551D25040C300A06082B06010505070301301D
    0603551D0E04160414234B9E6578CB280E3D968C5B6C4EA0911C1A7F73301F06
    03551D23041830168014E56DC55020881E3900398AF99EE0789DA4230F893081
    FB0603551D1F0481F33081F03081EDA081EAA081E7
Rx: 9000

Tx: A0800100F08681B16C6461703A2F2F2F434E3D6361776966692C434E3D616B6B
    6172312C434E3D4344502C434E3D5075626C69632532304B6579253230536572
    76696365732C434E3D53657276696365732C434E3D436F6E6669677572617469
    6F6E2C44433D62616472612C44433D656E73742C44433D66723F636572746966
    69636174655265766F636174696F6E4C6973743F626173653F6F626A65637443
    6C6173733D63524C4469737472696275746 96F6E506F696E748631687474703A
    2F2F616B6B6172312E62616472612E656E73742E66722F43657274456E726F6C
    6C2F6361776966692E63726C3082011306082B0601
Rx: 9000

Tx: A0800100F00505070101048201053082010130 81AA06082B0601050507300286
    819D6C6461703A2F2F2F434E3D6361776966692C434E3D4149412C434E3D5075
    626C69632532304B65792532305365727669636 5732C434E3D53657276696365
    732C434E3D436F6E66696775726174696F6E2C44433D62616472612C44433D65
```

```
6E73742C44433D66723F6341436572746966696361746 53F626173653F6F626A
656374436C6173733D636572746966696361746 96F6E417574686F7269747930
```

```
    5206082B06010505073002864668747470 3A2F2F616B6B6172312E6261647261
    2E656E73742E66722F43657274456E726F6C6C2F61
Rx: 9000

Tx: A0800000C46B6B6172312E62616472612E656E73742E66725F6361776966692E
    637274302106092B06010401823714020414 1E1200570065006200530065007 2
    007600650072300D06092A864886F70D010105050003820101009 46E33F7044A
    18F16E18337D8A22A230415DF07766ED94835E8A1FCBB7B16571D6EC6A9564AA
    C163383D17B223C29AB57825AE36156083249AA0A8EABED8C880D7E1EE58A301
    9D04D935EA3C6427052FDE1CCB60681691436C3580439F4C592ABA6489D43ABF
    EF9660EF60DA97FDA9
Rx: 6106

Tx: A0C0000006 // READ ACK#15
Rx: 021500060D009000

// Transfer Server Hello frag#2 1396 octets eap.request#16

Tx: A0800100F0011605740D40E8436722315A8D1479DCA19BFFC9F6B15A538D80E1
    A0C107F079DF79DB2674DD914481C8E1B388577645C100F44F4EC3A7E077CC4B
    3AC3577FD1CD0575E651FF1BCD6C716402DD83858563EC791593018CEB0BD9DB
    12F4B2E8D19FC185787E1717265BA3E11E76E343D2DA8AD83C77188E4E96C049
    B3F3B7BCB886BB574858FE331EE4407AA893212C171B1883A3B0EA580D000C63
    0201020C5E00C43081C1310B3009060355040613025553311730150603550 40A
    130E566572695369676E2C20496E632E313C303A060355040B1333436C617373
    2031205075626C6963205072696D61727920436572
Rx: 9000

Tx: A0800100F07469666936174696F6E20417574686F72697479202D204732313A
    3038060355040B1331286329203139393820566572695369676E2C20496E632E
    202D20466F7220617574686F72697A656420757365206F6E6C79311F301D0603
    55040B13165665726953696 76E205472757374204E6574776F726B00C43081C1
    310B3009060355040613025553311730150603550 40A130E566572695369676E
    2C20496E632E313C303A060355040B1333436C617373203420507562 6C696320
    5072696D617279204365727469666936174696F6E20417574686F7269747920
    2D204732313A3038060355040B1331286329203139
Rx: 9000

Tx: A0800100F0393820566572695369676E2C20496E632E202D20466F7220617574
    686F72697A656420757365206F6E6C79311F301D060355040B13165665726953
    69676E205472757374204E6574776F726B00D43081D1310B3009060355040613
    025A41311530130603550408130C5765737465726E204361706531123010060 3
    550407130943617065204 6F776E311A3018060355040A131154686177746520
    436F6E73756C74696E673128302606035504 0B131F436572746966963617469
    6F6E205365727669365732044697669736 96F6E3124302206035504031 31B54
    686177746520506572736F6E616C20467265656D61
Rx: 9000

Tx: A0800100F0696C204341312B302906092A864886F70D010901161C706572736F
```

```
    6E616C2D667265656D61696C407468617774652E636F6D00D23081CF310B3009
    060355040613025A4131153013060355040813 0C5765737465726E2043617065
```

```
    311230100603550407130943617065205 46F776E311A3018060355040A131154
    686177746520436F6E73756C74696E673128302606035504 0B131F4365727469
    6669636174696F6E205365727669636573204469766973696F6E312330210603
    550403131A546861777465205065727 36F6E616C205072656D69756D20434131
    2A302806092A864886F70D010901161B706572736F
Rx: 9000

Tx: A0800100F06E616C2D7072656D69756D407468617774652E636F6D0086308183
    310B3009060355040613025553312D302B060355040A13244669727374204461
    746120446967697461 6C204365727469666963617465732049 6E632E31453043
    0603550403133C466972737420446174612044696769746 16C20436572746966
    696361746573204 96E632E204365727469666963617469 6F6E20417574686F72
    69747900CE3081CB310B3009060355040613025A4131153013060355040813 0C
    5765737465726E204361706531123010060355040713094361706520546F776E
    311A3018060355040A131154686177746520436F6E
Rx: 9000

Tx: A0800000C473756C74696E673128302606035504 0B131F436572746966696361
    74696F6E205365727669636573204469766973696F6E3121301F060355040313
    185468617774652050657273 6F6E616C204261736963204341312830260609 2A
    864886F70D010901161970657273 6F6E616C2D626173696340746861774652E
    636F6D0061305F310B3009060355040613025553311730150603550 40A130E56
    6572695369676E2C20496E632E313730350603550 40B132E436C617373203320
    5075626C6963205072
Rx: 6106

Tx: A0C0000006 // Read ACK#16
Rx: 021600060D009000

// Transfer Server Hello frag#3 1396 octets eap.request#17

Tx: A0800100F0011705740D40696D617279204365727469666963617469 6F6E2041
    7574686F726974790061305F310B3009060355040613025553311730150603 55
    040A130E566572695369676E2C20496E632E313730350603550 40B132E436C61
    737320322050756 26C6963205072696D617279204365727469666963617469 6F
    6E20417574686F726974790061305F310B3009060355040613025553311730 15
    060355040A130E566572695369676E2C20496E632E313730350603550 40B132E
    436C617373203120507562 6C6963205072696D617279204365727469666963617
    74696F6E20417574686F7269747900C43081C1310B
Rx: 9000

Tx: A0800100F0300906035504061302555331173015060355040A130E5665726953
    69676E2C20496E632E313C303A060355040B1333436C617373203320507562 6C
    6963205072696D617279204365727469666963617469 6F6E20417574686F7269
    7479202D204732313A3038060355040B1331286329203139393820566572695 3
    69676E2C20496E632E202D20466F7220617574686F72697A656420757365206F
    6E6C79311F301D060355040B1316566572695369676E205472757374204E6574
    776F726B009C308199310B3009060355040613024855311130 0F060355040713
    0842756461706573743127302506035504 0A131E4E
```

Rx: 9000

```
Tx: A0800100F065744C6F636B2048616C6F7A617462697A746F6E73616769204B66
    742E311A3018060355040B131154616E7573697476616E796B6961646F6B3132
    303006035504031329 4E65744C6F636B20557A6C6574692028436C6173732042
    292054616E7573697476616E796B6961646F00473045310B3009060355040613
    02555331183016060355040A130F47544520436F72706F726174696F6E311C30
    1A06035504031313475445204379626572547275737420526F6F740077307531
    0B300906035504061302555331183016060355040A130F47544520436F72706F
    726174696F6E31273025060355040B131E47544520
Rx: 9000

Tx: A0800100F0437962657254727573742053 6F6C7574696F6E732C20496E632E31
    233021060355040313 1A475445204379626572547275737420476C6F62616C20
    526F6F7400C63081C3310B300906035504061302555331143012060355040A13
    0B456E74727573742E6E6574313B3039060355040B13327777772E656E747275
    73742E6E65742F43505320696E636F72702E206279207265662E20286C696D69
    7473206C6961622E29312530230603550 40B131C286329203139393920456E74
    727573742E6E6574204C696D69746564313A303806035504031331456E747275
    73742E6E65742053656375726520536572 76657220
Rx: 9000

Tx: A0800100F0436572746966696361746 96F6E20417574686F7269747900B23081
    AF310B300906035504061302485531103 00E06035504081307487 56E67617279
    3111300F06035504071308427564617065737431273025060355040A131E4E65
    744C6F636B2048616C6F7A617462697A746F6E73616769204B66742E311A3018
    060355040B131154616E7573697476616E796B6961646F6B3136303406035504
    03132D4E65744C6F636B204B6F7A6A6567797A6F692028436C61737320412920
    54616E7573697476616E796B6961646F00C43081C1310B300906035504061302
    555331173015060355040A130E566572695369676E
Rx: 9000

Tx: A0800000C42C20496E632E313C303A060355040B1333436C6173732032205075
    626C6963205072696D6172792043657274696669636174696F6E20417574686F
    72697479202D204732313A3038060355040B1331286329203139393820566572
    695369676E2C20496E632E202D20466F7220617574686F72697A656420757365
    206F6E6C79311F301D060355040B131656657269536967 6E205472757374204E
    6574776F726B0070306E310B300906035504061302555331183016060355040A
    130F47544520436F72
Rx: 6106

Tx: A0C0000006 // Transfer ACK#17
RX: 021700060D009000

// Read Server Hello frag#4 550 octets  eap.request#18
Tx: A0800100F0011802260D00706F726174696F6E3127302506035504 0B131E4754
    4520437962657254727573742053 6F6C7574696F6E732C20496E632E311C301A
    06035504031313475445204379626572547275737420526F6F74009E30819B31
    0B300906035504061302485531113 00F06035504071308427564617065737431
    273025060355040A131E4E65744C6F636B2048616C6F7A617462697A746F6E73
    616769204B66742E311A3018060355040B131154616E7573697476616E796B69
```

```
61646F6B31343032060355040313 2B4E65744C6F636B20457870726573737A20
28436C617373204329205461 6E7573697476616E79
```

```
Rx: 9000

Tx: A0800100F06B6961646F0054305231123010060A0992268993F22C6401191602
    667231143012060A0992268993F22C6401191604656E737431153013060A0992
    268993F22C640119160562616472613 10F300D06035504031306636177696669
    00723070312B3029060355040B1322436F707972696768742028632920313939
    37204D6963726F736F667420436F72702E311E301C060355040B13154D696372
    6F736F667420436F72706F726174696F6E3121301F060355040313184D696372
    6F736F667420526F6F7420417574686F726974790061305F31133011060A0992
    268993F22C6401191603636F6D31193017060A0992
Rx: 9000

Tx: A080000046268993F22C64011916096D6963726F736F6674312D302B06035504
    0313244D6963726F736F667420526F6F74204365727469666963617465204175
    74686F726974790E000000
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: 021805D60DC00000072116030106F10B0005E10005DE0005DB308205D7308204
    BFA003020102020A61253DFF000000000006300D06092A864886F70D01010505
    0030523112301006 0A0992268993F22C64011916026672311430120 60A099226
    8993F22C6401191604656E737431153013060A0992268993F22C640119160562
    61647261310F300D06035504031306636177696669301E170D30333131303630
    39333635395A170D303431313035303933363539 5A306231123010060A099226
    8993F22C64011916026672311430120 60A0992268993F22C6401191604656E73
    7431153013060A0992268993F22C640119160562616472613 10E300C06035504
    9F00

Tx: A012000000
Rx: 031305557365727331 0F300D06035504031306686 16A6A656830819F300D0609
    2A864886F70D010101050003818D0030818902818100DE7D0EF51D1716C06F51
    B04CEF2EC6CAF4D86601BC7B211237CEDC6172F3C8FF835C2FF52BF8F00FBD89
    866A3FC28B3BBDC798FD4B1D678F856612746F6474D031074604BAB17470B1FC
    D94244F897C274B9455C841533EC4ACB41D26E7C6DBDBCCD3E64FF8F3363FE06
    556996C696FA17DBF87FEB5BFE003ED18E428362BEC30203010001A382032130
    82031D300B0603551D0F0404030205A0304406092A864886F70D01090F043730
    35300E06082A864886F70D030202020080300E06082A864886F70D0304020200
    9F00

Tx: A012000000
Rx: 80300706052B0E030207300A06082A864886F70D0307301D0603551D0E041604
    14526E170649667E12FD1EC69D4CC8A02640B75928301706092B060104018237
    1402040A1E08005500730065007230 1F0603551D23041830168014E56DC55020
    881E3900398AF99EE0789DA4230F893081FB0603551D1F0481F33081F03081ED
```

```
A081EAA081E78681B16C6461703A2F2F2F434E3D6361776966692C434E3D616B
6B6172312C434E3D4344502C434E3D5075626C69632532304B65792532305365
```

```
    7276696365732C434E3D53657276696365732C434E3D436F6E66696775726174
    696F6E2C44433D62616472612C44433D656E73742C44433D66723F6365727469
    9F00

Tx: A012000000
Rx: 6669636174655265766F636174696F6E4C6973743F626173653F6F626A656374
    436C6173733D63524C4469737472696275746 96F6E506F696E74863168747470
    3A2F2F616B6B6172312E62616472612E656E73742E66722F43657274456E726F
    6C6C2F6361776966692E63726C3082011306082B06010505070101048201053 0
    8201013081AA06082B06010505073002868 19D6C6461703A2F2F2F434E3D6361
    776966692C434E3D4149412C434E3D5075626C69632532304B65792532305365
    7276696365732C434E3D53657276696365732C434E3D436F6E66696775726174
    696F6E2C44433D62616472612C44433D656E73742C44433D66723F6341436572
    9F00

Tx: A012000000
Rx: 7469666936174653F626173653F6F626A656374436C6173733D636572746966
    69636174696F6E417574686F7269747930520608 2B06010505073002864668 74
    74703A2F2F616B6B6172312E62616472612E656E73742E66722F43657274456E
    726F6C6C2F616B6B6172312E62616472612E656E73742E66725F636177696669
    2E63727430290603551D2504223020060A2B0601040182370A030406082B0601
    05050703040608 2B06010505070302302F0603551D1104283026A024060A2B06
    0104018237140203A0160C1468616A6A65684062616472612E656E73742E6672
    300D06092A864886F70D01010505000382010100 13A233AA6EDB4282A69EF9D0
    9FD6

Tx: A0120000D6
Rx: 23D51F32FD0B97AF03C4BACD6B7ED5C155110EBACC3F0FAD6D853DEE845CC33D
    0E9D8ECC7514295F854D16F6409DFEB61A60C9A1EF0BC09AD3C1A93BEE546B2D
    F9DBAB8AD9A90AAB5CEE35FF6751275873D1C5093339B4ADEA0F40C54754DAE7
    461966322B5772B460B7FA2F5985D496C52CAF7456DF2D78E4DE9B1C48F2ACB9
    87BA9BDE3D1624645330F0FBF0103C547DA547C1F03B1C2BB5CDD06D38D2ABFA
    FD06387235E8E49DEDCB7E2B7E80A15B1317A04ECF1ADBF475AC82D67514A6EF
    5EBFFAD40D5D5F7395179677703BFC3A9D34623BD28E9000

// Read ACK#19
Tx: A0800000060119000 60D00
Rx: 9F00

// Transfer 2nd fragment, 347 bytes, Code=2 id=19 Length=347
// Type=0D Flag=00
//
Tx: A012000000
Rx: 0219015B0D00C9186A1078130652552D5CFEF1B6CDBA5197910A4C87CAD1F92F
    A7EB7A0B1000008200808FD83C571FE7D71E76A86405BDBC95BA4BD67A48F4BD
    8084F4F944C1ACDF1FACF85FFC111BE3CE8AFFB48F6DA6C5477761A34C7889CB
    148DA42141BBC1E942BAC8752B7FD255574F654DBED3DEF89EE0F79BEEBF43DC
    737F158F99C17A2461B2C5D5E2A75FCBBD7F5275AD781127300E46EC61408EF2
    BABC200F85363926301E0F0000820080BDD2429D21DAE14D9727D2F715BF30A6
```

```
    5E61C7608D5C0B6035BCCC014BAFE24BB98550AF86E13B6D8D371E5A922D20DD
    338B563B7E9C9AF0EF9110C77B468A651915575D348A7D29B89CC5A8D4B8AA71
```

```
    9F5B

Tx: A01200005B
Rx: 5D53E340E6E7AD6B6E3438F358B870C5DA5E61C45EE5E3F9454219F48A34CC98
    10A946F0C652675E3CA81ABA229309B7140301000101160301002 0C97EBCFF0C
    20271CAE21FAA80898278660D393CB4C640390CDEB14592A0392F79000

// Transfer Server last message, eap request #1A
Tx: A080000035011A00350D800000002B1403010001011603010020 9255D2089E41
    30B5984AF43B604A108AA11376F368E71BCF81EEFEBC00289C1C
Rx: 6106

// Read ACK#1A
Tx: A0C0000006
Rx: 021A00060D009000

// Read MSK
Tx: A0A6000040
Rx: 8F0A6773E9C0264015861CE712C9A692844A28B6D5641E4D90D38994A94A2C6D
    B7CD0C7DCBD83D45B2DB1D6598FE696A10176E21B62D8A33AD2970A560CE5E84
    9000
//
```

17.3 EAP-TLS mode1 ISO7816 trace (T=0 protocol)

The EAP-TLS smartcard mode1, 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.

```
// Set-Identity (abc TLS) type=TLS
Tx: A016800003616263
Rx: 9000

// RANDOM Number Generator
Tx: A060 0200 1C  // 28 bytes
Rx: 08BDD285B43D1F3BC9715FC9F85FC453FE58F3A9E07FF397CD653922

// Set Server Public KEY (FCT = Initialize + Public-Encrypt)
Tx: A0604800870080bc67013cb915ec1281e65a4daf49801ddb6d5cf30cfd2ff63f
    5d377929c7391bfd766f67dd0fe9e8425143ba46ae95ff76919f30a39c459a22
    f22b7566529795c32fee7dcfc9dcde1169a346efe825246214df022badf983b9
    cbba81c44c15a1139701b69f9954c9bd2fdfa1ae401e3bd6fd06cf585413c28a
    e802b4670a8f30003010001
```

```
// Pre-Master Secret Encryption with the Server Public Key
// FCT = Do-Final + Public-Encrypt
Tx: A0600800300301c5a68fb75123308e2ddbb27b63fe021e8724e7bc5c17078b3b
    3f90ba00d128f80b07ad786b6de36e5f94ffdfeb49
RX: 6180
TX: 8fd83c571fe7d71e76a86405bdbc95ba4bd67a48f4bd8084f4f944c1acdf1fac
    f85ffc111be3ce8affb48f6da6c5477761a34c7889cb148da42141bbc1e942ba
    c8752b7fd255574f654dbed3def89ee0f79beebf43dc737f158f99c17a2461b2
    c5d5e2a75fcbbd7f5275ad781127300e46ec61408ef2babc200f85363926301e

// Private Encrypt with Client Private Key
// FCT = Do-Final + Private-Encrypt
// (Client Certificate Verify)
Tx: A0604002249c0326e6d899fa802cc981b86e9b65f41234db8e2456e5f3dccd68
    a34f25b4e72153f50e
Rx: 6180
Tx: A0C0000080
Rx: bdd2429d21dae14d9727d2f715bf30a65e61c7608d5c0b6035bccc014bafe24b
    b98550af86e13b6d8d371e5a922d20dd338b563b7e9c9af0ef9110c77b468a65
    1915575d348a7d29b89cc5a8d4b8aa715d53e340e6e7ad6b6e3438f358b870c5
    da5e61c45ee5e3f9454219f48a34cc9810a946f0c652675e3ca81aba229309b7

// Public Decrypt#1 with CA public key, first byte
// FCT = Do-Final + Index#1 + Public-Decrypt
// Checking of server certificate
Tx: A061 1B 00 01 13
Rx: 9000
// Public Decrypt#1 (with CA public key, 255 bytes)
Tx: A0601A00FFA233AA6EDB4282A69EF9D023D51F32FD0B97AF03C4BACD6B7ED5C1
    55110EBACC3F0FAD6D853DEE845CC33D0E9D8ECC7514295F854D16F6409DFEB6
    1A60C9A1EF0BC09AD3C1A93BEE546B2DF9DBAB8AD9A90AAB5CEE35FF67512758
    73D1C5093339B4ADEA0F40C54754DAE7461966322B5772B460B7FA2F5985D496
    C52CAF7456DF2D78E4DE9B1C48F2ACB987BA9BDE3D1624645330F0FBF0103C54
    7DA547C1F03B1C2BB5CDD06D38D2ABFAFD06387235E8E49DEDCB7E2B7E80A15B
    1317A04ECF1ADBF475AC82D67514A6EF5EBFFAD40D5D5F7395179677703BFC3A
    9D34623BD28EC9186A1078130652552D5CFEF1B6CDBA5197910A4C87CAD1F92F
    A7EB7A0B
Rx: 6123
Tx: A0C0000023 // Certificate Hash
Rx: 3021300906052B0E03021A0500041429A563710F25832AFB692E44F4B9AFF36F
    BE91A79000

// Read Client Certificate
Tx: A060000000 // Certificate 1st fragment
Rx: 308205D7308204BFA003020102020A61253DFF000000000006300D06092A8648
    86F70D01010505003052311230100 60A0992268993F22C640119160266723114
    3012060A0992268993F22C6401191604656E737431153013060A0992268993F2
    2C640119160562616472613 10F300D06035504031306636177696669301E170D
    303331313036303933363539 5A170D303431313035303933363539 5A30623112
```

3010060A0992268993F22C64011916026672311430120 60A0992268993F22C64
01191604656E737431153013060A0992268993F22C64011916056261647261 31

```
    0E300C060355040313055573657273310F300D060355040313066861 6A6568
    9F00
Tx: A012000000 // Certificate 2nd fragment
Rx: 30819F300D06092A864886F70D010101050003818D0030818902818100DE7D0E
    F51D1716C06F51B04CEF2EC6CAF4D86601BC7B211237CEDC6172F3C8FF835C2F
    F52BF8F00FBD89866A3FC28B3BBDC798FD4B1D678F856612746F6474D0310746
    04BAB17470B1FCD94244F897C274B9455C841533EC4ACB41D26E7C6DBDBCCD3E
    64FF8F3363FE06556996C696FA17DBF87FEB5BFE003ED18E428362BEC3020301
    0001A38203213082031D300B0603551D0F0404030205A0304406092A864886F7
    0D01090F04373035300E06082A864886F70D030202020080300E06082A864886
    F70D0304020200803007060 52B0E030207300A06082A864886F70D0307301D06
    9F00
Tx: A012000000 // Certificate 3rd fragment
Rx: 03551D0E04160414526E170649667E12FD1EC69D4CC8A02640B7592830170609
    2B06010401823714020 40A1E0800550073006500723 01F0603551D2304183016
    8014E56DC55020881E3900398AF99EE0789DA4230F893081FB0603551D1F0481
    F33081F03081EDA081EAA081E78681B16C6461703A2F2F2F434E3D6361776966
    692C434E3D616B6B6172312C434E3D4344502C434E3D5075626C69632532304B
    657925323053657276696365732C434E3D53657276696365732C434E3D436F6E
    66696775726174696F6E2C44433D62616472612C44433D656E73742C44433D66
    723F636572746966696361746552657 66F636174696F6E4C6973743F62617365
    9F00
Tx: A012000000 // Certificate 4th fragment
Rx: 3F6F626A656374436C6173733D63524C4469737472696275746 96F6E506F696E
    748631687474703A2F2F616B6B6172312E62616472612E656E73742E66722F43
    657274456E726F6C6C2F6361776966692E63726C3082011306082B0601050507
    0101048201053082010 13081AA06082B06010505073002868 19D6C6461703A2F
    2F2F434E3D6361776966692C434E3D4149412C434E3D5075626C69632532304B
    657925323053657276696365732C434E3D53657276696365732C434E3D436F6E
    66696775726174696F6E2C44433D62616472612C44433D656E73742C44433D66
    723F6341436572746966696361746 53F626173653F6F626A656374436C617373
    9F00
Tx: A012000000 // Certificate 5th fragment
Rx: 3D636572746966696361746 96F6E417574686F7269747930520608 2B06010505
    0730028646687474703A2F2F616B6B6172312E62616472612E656E73742E6672
    2F43657274456E726F6C6C2F616B6B6172312E62616472612E656E73742E6672
    5F6361776966692E63727430290603551D2504223020060A2B0601040182370A
    030406082B06010505070304 06082B06010505070302302F0603551D11042830
    26A024060A2B06010401823714020 3A0160C1468616A6A65684062616472612E
    656E73742E6672300D06092A864886F70D010105050003820101001 3A233AA6E
    DB4282A69EF9D023D51F32FD0B97AF03C4BACD6B7ED5C155110EBACC3F0FAD6D
    9FDB
Tx: A0120000DB // Certificate 6th and last fragment
Rx: 853DEE845CC33D0E9D8ECC7514295F854D16F6409DFEB61A60C9A1EF0BC09AD3
    C1A93BEE546B2DF9DBAB8AD9A90AAB5CEE35FF6751275873D1C5093339B4ADEA
    0F40C54754DAE7461966322B5772B460B7FA2F5985D496C52CAF7456DF2D78E4
    DE9B1C48F2ACB987BA9BDE3D1624645330F0FBF0103C547DA547C1F03B1C2BB5
    CDD06D38D2ABFAFD06387235E8E49DEDCB7E2B7E80A15B1317A04ECF1ADBF475
    AC82D67514A6EF5EBFFAD40D5D5F7395179677703BFC3A9D34623BD28EC9186A
```

1078130652552D5CFEF1B6CDBA5197910A4C87CAD1F92FA7EB7A0B9000

18 Annex 7, EAP-AKA ISO7816 APDUs Trace (T=0 Protocol)

This annex gives test vectors for the EAP-AKA protocol, introduced by [EAP-AKA]

```
// Select EAP_APPLICATION
Tx: 00A40400 07 11 22 33 44 55 66 01
Rx: 9000

// Verify User PIN ('0000')
Tx: A020000004 30303030
Rx: 9000

// Set-Identity ('zzz') type=AKA
Tx: A0 16 00 80 03 7A 7A 7A
// 90 00

// EAP-Identity request
Tx: A0 80 00 00 05 01 A4 00 05 01
// Identity.response: anonymous@dot.com
Rx: 02 A4 00 16 01 61 6E 6F 6E 79 6D 6F 75 73 40 64 6F 74 2E 63 6F
    6D 90 00

// EAP-AKA GET AT-PERMANENT-ID-REQ: aka@dot.com
Tx: A0 80 00 00 0A 01A6 000A 1705 0A01 0000
Rx: 02 A6 00 16 17 05 0E 04 00 0B 61 6B 61 40 64 6F 74 2E 63 6F 6D
    00 90 00
//=================
// Milenage Values=
//=================
// These values are imported from
// 3GPP TS 35.207 V5.0.0 (2002-06),
// sections 4.3-Test set 1 and  6.3-Test set 1

// K:    465b5ce8 b199b49f aa5f0a2e e238a6bc
// OP:   cdc202d5 123e20f6 2b6d676a c72cb318
// SQN:  ff9bb4d0b607
// AMF:  b9b9
// RAND: 23553cbe 9637a89d 218ae64d ae47bf35

// f1|f1*: 4A 9F FA C3 54 DF AF B3 01 CF AF 9E C4 E8 71 E9
// f2/sres A5 42 11 D5 E3 BA 50 BF
// f3/ck   B4 0B A9 A3 C5 8B 2A 05 BB F0 D9 87 B2 1B F8 CB
// f4/ik   F7 69 BC D7 51 04 46 04 12 76 72 71 1C 6D 34 41
// f5/ak   AA 68 9C 64 83 70
// f5*/ak2 45 1E 8B EC A4 3B
```

```
//=============================
// Values for XKEY & PRF(XKEY)=
//=============================
// ID: 61 6B 61 40 64 6F 74 2E 63 6F 6D = aka@dot.com
// IK: F7 69 BC D7 51 04 46 04 12 76 72 71 1C 6D 34 41
// CK: B4 0B A9 A3 C5 8B 2A 05 BB F0 D9 87 B2 1B F8 CB
// XKEY = MK = sha1(ID|IK|CK) =
// C4 83 4F 21 BE AD F0 9E 7A 3B E8 17 97 5A BA 99 DD B4 0C 9A

// PRF(XKEY)
// K_Encr: 28 FF 32 38 42 05 6B 55 4B 85 A5 11 16 34 5A A4
// K_Auth: B3 08 06 82 48 8E 68 6F AC 3E 1C F8 24 8E 73 63
// MSK:    BE 12 98 C0 B5 33 8C 91 D6 E1 1B 33 AE 7D 46 2D
//         E2 99 64 64 0C F5 05 FF 26 AE D5 98 82 2D 41 F9
//         20 AF 49 FD CB 77 00 8C 2A AC DB A3 A1 AE 79 75
//         20 8C 25 E5 40 17 5D 22 D5 48 0C DE 88 D7 90 33
// EMSK:   CD 10 C9 14 BB 54 DC 97 AE E8 96 06 67 F8 C8 59
//         12 44 DF E7 BD 4A C1 B1 6E 63 1B 4D FA 5D F6 97
//         4A 4C 51 F5 D8 19 FE 68 E7 37 0F 9E 47 43 9B 43
//         FD 6E 83 CC 35 7A 01 E7 16 57 F3 BE 6D 26 4A 2B


//========================================
// Test #1 : FULL AUTHENTICATION, GOOD SQN
//========================================
//AT-RAND AT-AUTN AT-MAC
Tx: A0 80 00 0044 01A5 0044 17010000 01050000 23553CBE
    9637A89D 218AE64D AE47BF35 02050000 55F328B43577 B9B9 4A9FFAC3
    54DFAFB3 0B050000 C7003536 662D5201 B011F20F E5DD8CE4

// AT-RES AT-MAC
Rx: 02 A5 0028 17010000 03030040 A54211D5 E3BA50BF
    0B050000 45703D12 9567DCA9 2C9101C4 9392F267 9000

// Get MSK
Tx: A0 A6 00 00 40
Rx: 20 AF 49 FD CB 77 00 8C 2A AC DB A3 A1 AE 79 75 20 8C 25 E5 40
    17 5D 22 D5 48 0C DE 88 D7 90 33 BE 12 98 C0 B5 33 8C 91 D6 E1
    1B 33 AE 7D 46 2D E2 99 64 64 0C F5 05 FF 26 AE D5 98 82 2D 41
    F9 90 00
```

```
//=========================================
// Test #2 : FULL AUTHENTICATION, WRONG SQN
//=========================================
//GPP TS 33.102, Release 9 V9.2.0 (2010-03), page 23:
//The AMF used to calculate MAC-S assumes a dummy value of all zeros
//so that it does not need to be transmitted in the clear in the
//re-synch message.

Tx: A0 80 00 0044 01A5 0044 17010000 01050000 23553CBE
    9637A89D 218AE64D AE47BF35 02050000 55F328B43577 B9B9 4A9FFAC3
    54DFAFB3 0B050000 C7003536 662D5201 B011F20F E5DD8CE4

// According to 3GPP TS 33.102 V6.4.0 (2005-09)
// AT_AUTS = AK2+SQNms | MAC-S
// MAC-S   = f1*(AMF=0000,RAND,SQNms)
// AK2     = f5*(RAND)

// AK2   =  45 1E 8B EC A4 3B
// SQNms =  ff 9b b4 d0 b6 08
// MAC-S  = 00 10 C1 DA 38 A7 5A 31

Rx: 02A50018 1704 0000 0404 BA853F3C1233 0010C1DA38A75A31 9000

//=========================================
// Test #3 : FULL AUTHENTICATION, WRONG MAC=
//=========================================
Tx: A0 80 00 0044 01A5 0044 17010000 01050000 23553CBE
    9637A89D 218AE64D AE47BF36 02050000 55F328B43577 B9B9 4A9FFAC3
    54DFAFB3 0B050000 C7003536 662D5201 B011F20F E5DD8CE4

// AKA-Authentication-Reject
Rx: 02 A5 0008 17020000 9000

//==========================================
// Test #4 : Full Authentication + PSEUDO-ID=
//==========================================

// AT-RAND AT-AUTN AT-ENCR AT-MAC
// AT-ENCR: 82090000 8205000D 31323334 31323334 31323334 31000000
//          06030000 00000000 00000000

Tx: A0 80 00 00 7C 01A5 007C 17010000 01050000 23553CBE 9637A89D
    218AE64D AE47BF35 02050000 55F328B43577 B9B9
    4A9FFAC3 54DFAFB3 81050000 12345678 12345678 12345678 12345678
    82090000 819DCAF9 E851072D 660A36FB 79D96C09 6AC36F2E 58D6E32D
    3FC84869 9DA076D4 0B050000 B05E0FFC 0A99A434 2A2BFAD8 1900F1B3

// AT-RES AT-MAC
Rx: 02 A5 00 28 17010000 03030040 A54211D5 E3BA50BF 0B050000
    45703D12 9567DCA9 2C9101C4 9392F267 9000
```

```
// AT-FULLAUTH-ID-REQ = "12341234123412341@dot.com"
Tx: A0 80 00 00 0C 01A6 000C 17050000 11010000
Rx: 02 A6 00 24 17050000 0E070015 31323334 31323334 31323334
    3140646F 742E636F 6D000000 9000

// AT-PERMANENT-ID-REQ = "aka@dot.com"
Tx: A0 80 00 00 0C 01A6 000C 17050000 0A010000

Rx: 02 A6 00 18 17050000 0E04000B 616B6140 646F742E 636F6D00 9000

//=============================================
// Test #5 : Full Authentication + ReAUTH-ID=
//=============================================
// AT-RAND AT-AUTN AT-ENCR AT-MAC
// AT-ENCR: 82090000 8505000D 31323334 31323334 31323334 31000000
//          06030000 00000000 00000000

Tx: A0 80 00 00 7C 01A5 007C 17010000 01050000 23553CBE 9637A89D
    218AE64D AE47BF35 02050000 55F328B43577 B9B9
    4A9FFAC354DFAFB3 81050000 12345678 12345678 12345678 12345678
    82090000 49E8E4BE 42452611 89AFE3A1 E913953F 4A966DBE 53D621A8
    CC771072 DA7B1964
    0B050000 4081C920 AB6A42EB A06DD4B6 A598D741

// AT-RES AT-MAC
Rx: 02 A5 00 28 17010000 03030040 A5 42 11 D5 E3 BA 50 BF
    0B050000 45703D12 9567DCA9 2C9101C4 9392F267 9000

// GET AT-ANY-ID-REQ: "1234123412341"
Tx: A0 80 00 00 0C 01A6 000C 17050000 0D010000
Rx: 02 A6 00 1C 17050000 0E05000D 31323334 31323334 31323334
    31000000 9000

//==============================
// Test #6: ReAUTH, GoodCounter=
//==============================
```

According to [EAP-AKA] for EAP-Response/AKA-Reauthentication, the MAC code is calculated over the following data: EAP packet | NONCE-S

```
// XKEY'    = SHA1(Identity | counter | NONCE-S | MK)
// Identity = 31323334 31323334 31323334 31
// Counter=   0001
// NONCE-S=   12345678 12345678 12345678 12345678
// MK =       C4834F21BEADF09E7A3BE817975ABA99DDB40C9A
// XKEY'=     7A790D9602767568BF4D9AD23C0E28F44A0A64B3
// PRF(XKEY') =
// C2BCFE5D383ED9C30F55B83619BF1C7A09A26320AF7F323AD9A0E58CCBE1FA7E
// 6894EE064D7E38C6EBEFFE95DBEC150759B08C18B6AF02EF9D7E52A52B670E13
```

```
// AT-IV AT-ENCR AT-MAC
// ENCR: 82090000 13010001 15050000 12345678 12345678 12345678
//       12345678 85020004 31323334

Tx: A080000054 01A50054 170D0000
    81050000 12345678 12345678 12345678 12345678
    82090000 99571855 2FAF6D99 57A67521 3A37A839 A76923F4 6A0040CF
    839AC1E4 DC8B3CDF
    0B050000 842D9F634BAA8D4BC10EF5A78C90F705


// AT-ENCR  AT-COUNTER AT-PADDING
// AT_ENCR: 82040000 13010001 06030000 00000000 00000000

Rx: 02A50044 170D0000 81050000 A5A5A5A5A5A5A5A5A5A5A5A5A5A5A5A5
    82050000 C528C7A2154DF5BA6744A249557CC823
    0B050000 99B31F633BB5BD9399DBD2B321376258
    9000


// Get MSK
Tx: A0 A6 00 00 40
Rx: 6894EE064D7E38C6EBEFFE95DBEC150759B08C18B6AF02EF9D7E52A52B670E13
    C2BCFE5D383ED9C30F55B83619BF1C7A09A26320AF7F323AD9A0E58CCBE1FA7E
    9000

//==============================
// Test #7: ReAUTH,WrongCounter=
//==============================

// AT-IV AT-ENCR AT-MAC
// AT-ENCR: 82090000 13010001 15050000 12345678 12345678 12345678
//          12345678 85020004 31323334

Tx: A080000054 01A50054 170D0000
    81050000 12345678 12345678 12345678 12345678
    82090000 99571855 2FAF6D99 57A67521 3A37A839 A76923F4 6A0040CF
    839AC1E4 DC8B3CDF
    0B050000 842D9F634BAA8D4BC10EF5A78C90F705

// AT-ENCR AT-COUNTER-TOO-SMALL AT-COUNTER AT-PADDING
// AT-ENCR: 82050000 14010000 13010001 06020000 00000000

Rx: 02A50044 170D0000 81050000 A5A5A5A5A5A5A5A5A5A5A5A5A5A5A5A5
    82050000 605F0CA17BBDE27ABFF0A82D20E2B945
    0B050000 C79D53756722DEDC753FD0D85C1A90FD
    9000
```

19 IANA Considerations

This draft does not require any action from IANA.

20 References

20.1 Normative References

[RFC 3748] B. Aboba, L. Blunk, J. Vollbrecht,C. Sun, H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004

[L2P] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, B. Palter "Layer Two Tunneling Protocol", RFC 2661, August 1999

[TLS] T.Dierks, E. Rescorla, RFC 5246, "The TLS Protocol Version 1.2", August 2008.

[GSM 11.11] GSM Technical Specification GSM 11.11. Digital cellular telecommunications system (Phase 2+); Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME)

[IEEE 802.11] Institute of Electrical and Electronics Engineers, "Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Standard 802.11, 1999

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

[IEEE 802.11i] Institute of Electrical and Electronics Engineers, "Approved Draft Supplement to Standard for Telecommunications and Information Exchange Between Systems-LAN/MAN Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Specification for Enhanced Security", IEEE 802.11i-2004, 2004.

[RFC 4282] B. Aboba, J. Arkko, P. Eronen, "The Network Access Identifier" RFC 4282, December 2005

[ASN.1] ASN.1 standard 2002 edition ISO/IEC 8825.1. http://asn1.elibel.tm.fr/en/standards/index.htm

[XML] Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000

[RFC 5216] B. Aboba, D. Simon, R. Hurst, "EAP TLS Authentication Protocol" RFC 5216, March 2008.

[PKCS1] "PKCS #1: RSA Encryption Standard", RSA Laboratories,

[PKCS6] PKCS #6: "Extended-Certificate Syntax Standard, An RSA Laboratories Technical Note", RSA Laboratories.

[RFC 3748] B. Aboba, L. Blunk, J. Vollbrecht, J. C. Sun, H. Levkowetz, "Extensible Authentication Protocol (EAP)" RFC 3748, June 2004

[RFC 4017] D. Stanley, J. Walker, B. Aboba, "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs", March 2005.

[RFC 4137] J. Vollbrecht, P. Eronen, N. Petroni, Y. Ohba, "State Machines for Extensible Authentication Protocol (EAP)Peer and Authenticator", August 2005

[EAP-SIM] H. Haverinen, J. Salowey, "Extensible Authentication Protocol Method for GSM Subscriber Identity Modules (EAP-SIM)", EAP SIM Authentication", RFC 4186, January 2006.

[EAP-AKA] J. Arkko, H. Haverinen, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)" RFC 4187, January 2006

[IKEv2] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, "Internet Key Exchange (IKEv2) Protocol", RFC 5996, September 2010

[IEEE 802.16-2004] IEEE Standard for Local and metropolitan area networks. Part 16: Air Interface for Fixed Broadband Wireless Access Systems - 2004

[IEEE 802.16e] IEEE Standard for Local and metropolitan area networks. - Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems - Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1, February 2006

[TS 102 310] ETSI TS 102 310 V6.2.0 (2005-09) Technical Specification Smart Cards; Extensible Authentication Protocol support in the UICC(Release 6)

[HOKEY-EMSK] J. Salowey, L. Dondeti, V. Narayanan, M. Nakhjiri, "Specification for the Derivation of Root Keys from an Extended Master Session Key (EMSK)", RFC 5295, August 2008

[EAP-KEY] Bernard Aboba, Dan Simon, P. Eronen, H. Levkowetz, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, August 2008

20.2 Informative References

[NIST-PIV]: Special Publication 800-73-1 Interfaces for Personal Identity Verification, March 2006

[EAP-SC] P.Urien, W.Habraken, D.Flattin , H.Ganem , "draft-urien-eap-smartcard-type-02.txt", October 2005

[WiMAX-Forum-Stage2] "WiMAX End-to-End Network Systems Architecture (Stage 2: Architecture Tenets, Reference Model and Reference Points)" draft, august 2006

[EAP-EXT] Bernard Aboba, "Extensible Authentication Protocol (EAP) Key Management Extensions", draft-aboba-eap-keying-extens-00.txt, April 2005

[PEAP] Ashwin Palekar, Dan Simon, Joe Salowey, Hao Zhou, Glen Zorn,S. Josefsson, "Protected EAP Protocol (PEAP) Version 2" draft-josefsson-pppext-eap-tls-eap-10.txt, work-in-progress, October 2004.

21 Authors' Addresses

Pascal Urien
Telecom ParisTech
23 avenue d' Italie
75013 Paris Phone: NA
France Email: Pascal.Urien@telecom-paristech.fr

Guy Pujolle
LIP6 - University Paris 6
4 place jussieu Phone: NA
Paris 75005 France Email: Guy.Pujolle@lip6.fr