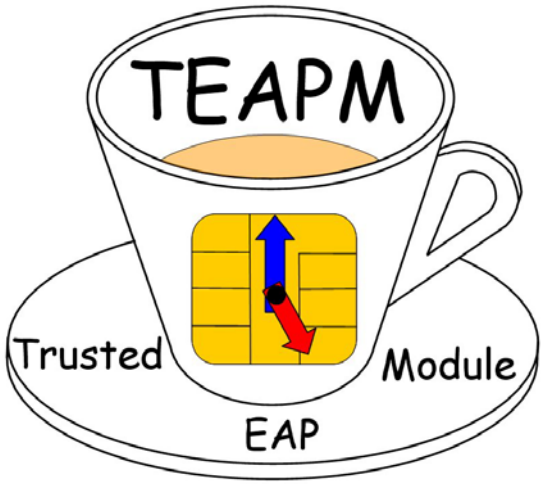


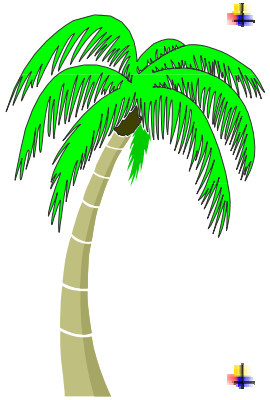
TEAPM
Trusted EAP Module

Pascal Urien, Guy Pujolle

Pascal.Urien@enst.fr

Guy.Pujolle@lip6.fr

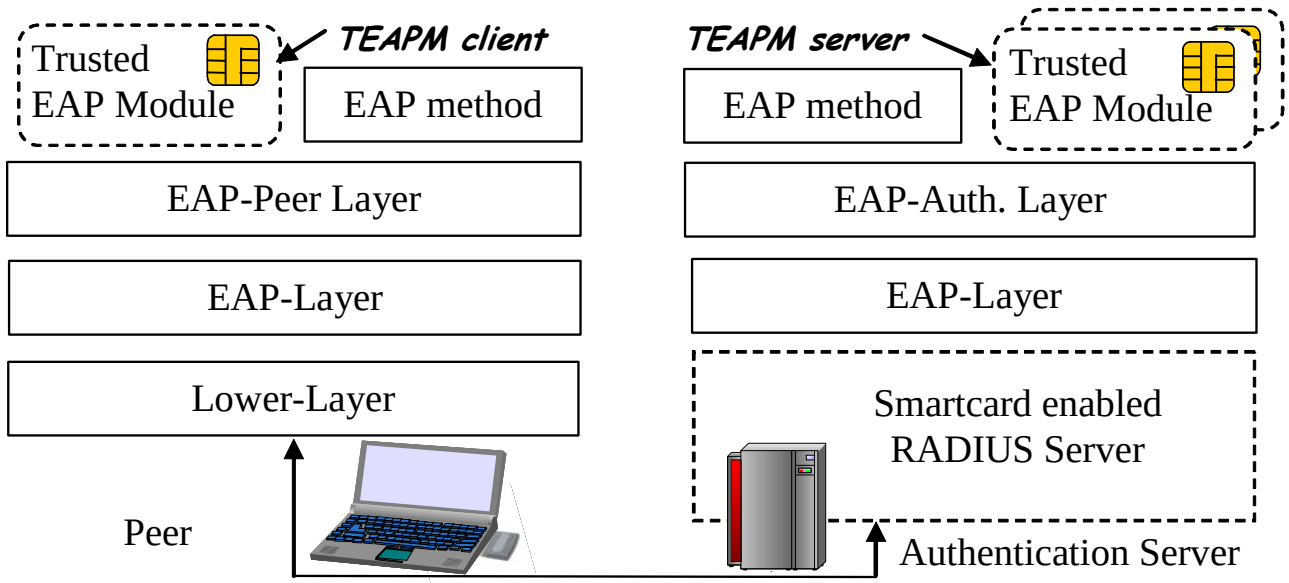
AGENDA



1. Introduction

1.1 What is a TEAPM ?

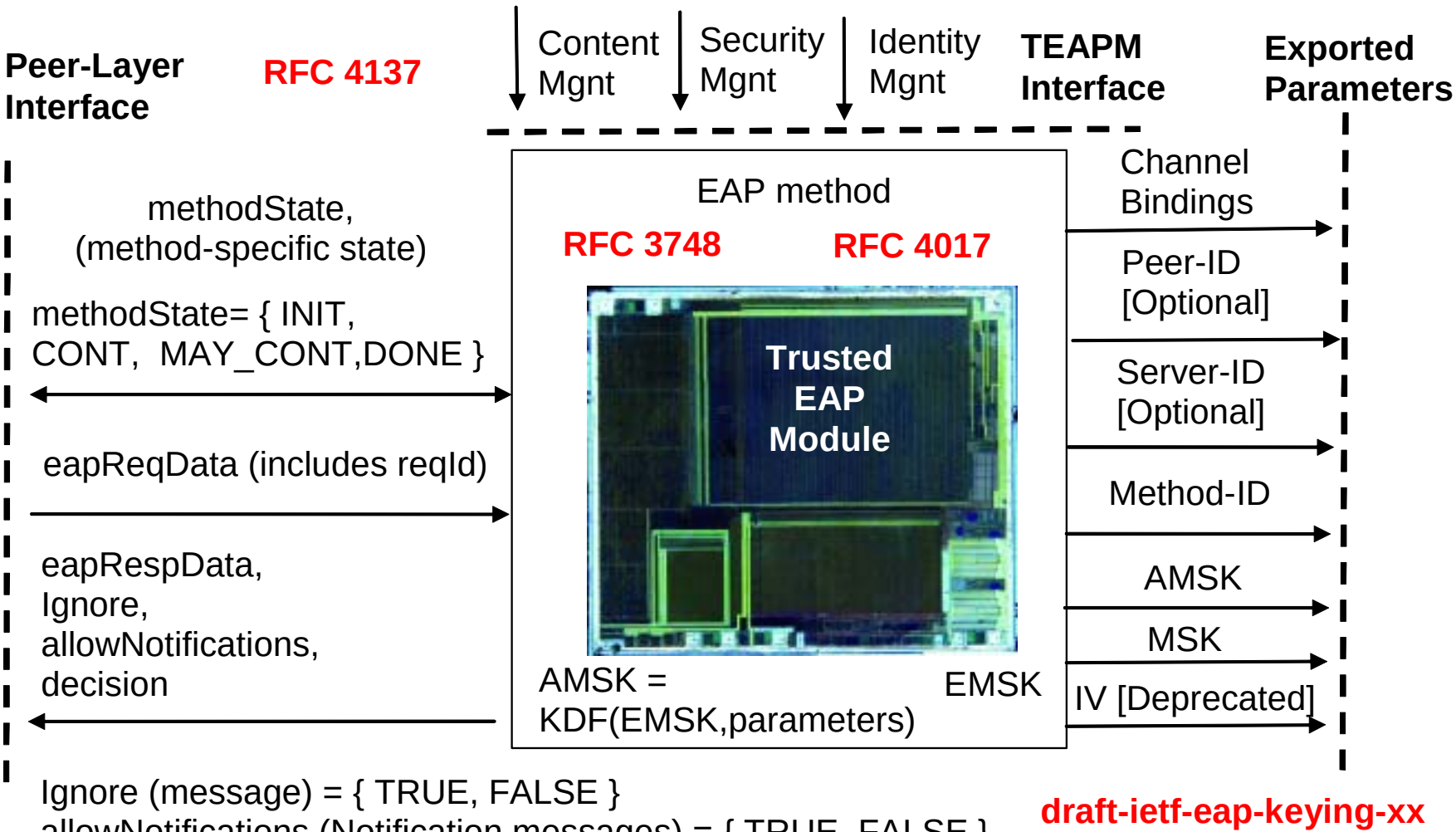
- **TEAPMs are smartcards that run EAP client and/or server applications.**
- **A public javacard implementation, based on the *OpenEapSmartcard* platform is available on the WEB. Multiple client and server entities may simultaneously work in a 64 KB device.**
- **TEAPMs benefits**
  - Security modules dedicated to IP devices.
  - Independent of any operating system (Windows, LINUX,…).
  - Highly secure authentication servers.
  - Privacy and tracability .
  - Remote administration.



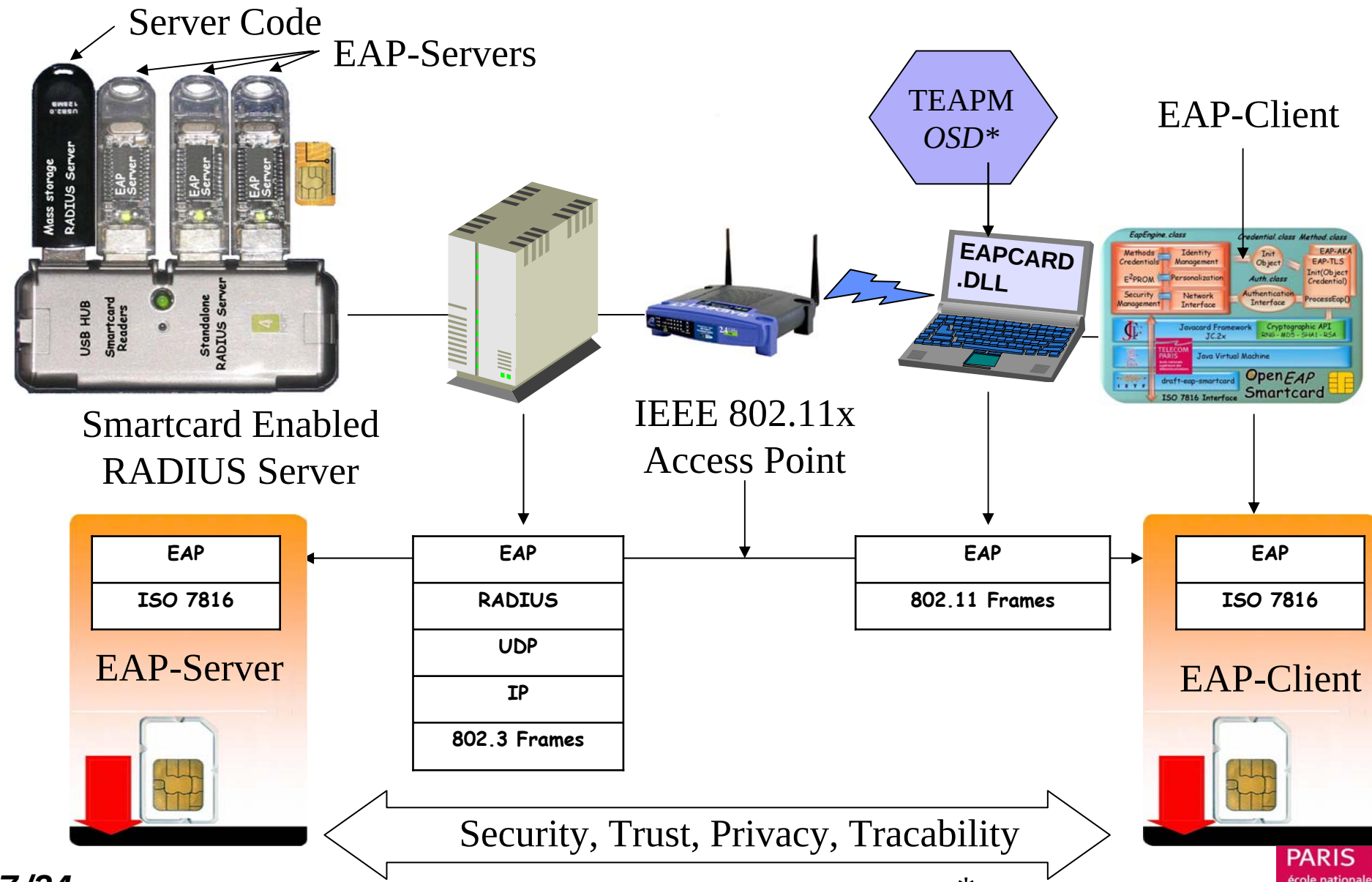
1.2 What is EAP ?

- **EAP is a new *Esperanto* for IP networks**
- **RFC 2284, "PPP Extensible Authentication Protocol (EAP)",1998.**
  - RFC 2661, "Layer Two Tunneling Protocol (L2TP) ", 1999.
  - RFC 2637, " Point-to-Point Tunneling Protocol (PPTP) ", 1999.
- **IEEE 802.1x, 2001**
- **RFC 3559, "RADIUS Support For Extensible Authentication Protocol", 2003**
- **RFC 3748, Extensible Authentication Protocol, 2004**
  - RFC 2716, "PPP EAP TLS Authentication Protocol“, 1999.
  - RFC 4186, "Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM) ", 2006
  - RFC 4187, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)", 2006
- **RFC 4072, "Diameter Extensible Authentication Protocol Application ", 2005**
- **RFC 4306, "Internet Key Exchange (IKEv2) Protocol", 2005**
- **IEEE 802.16e (WiMAX mobile), PKM-EAP, 2005**

1.3 The TEAPM in the IETF context

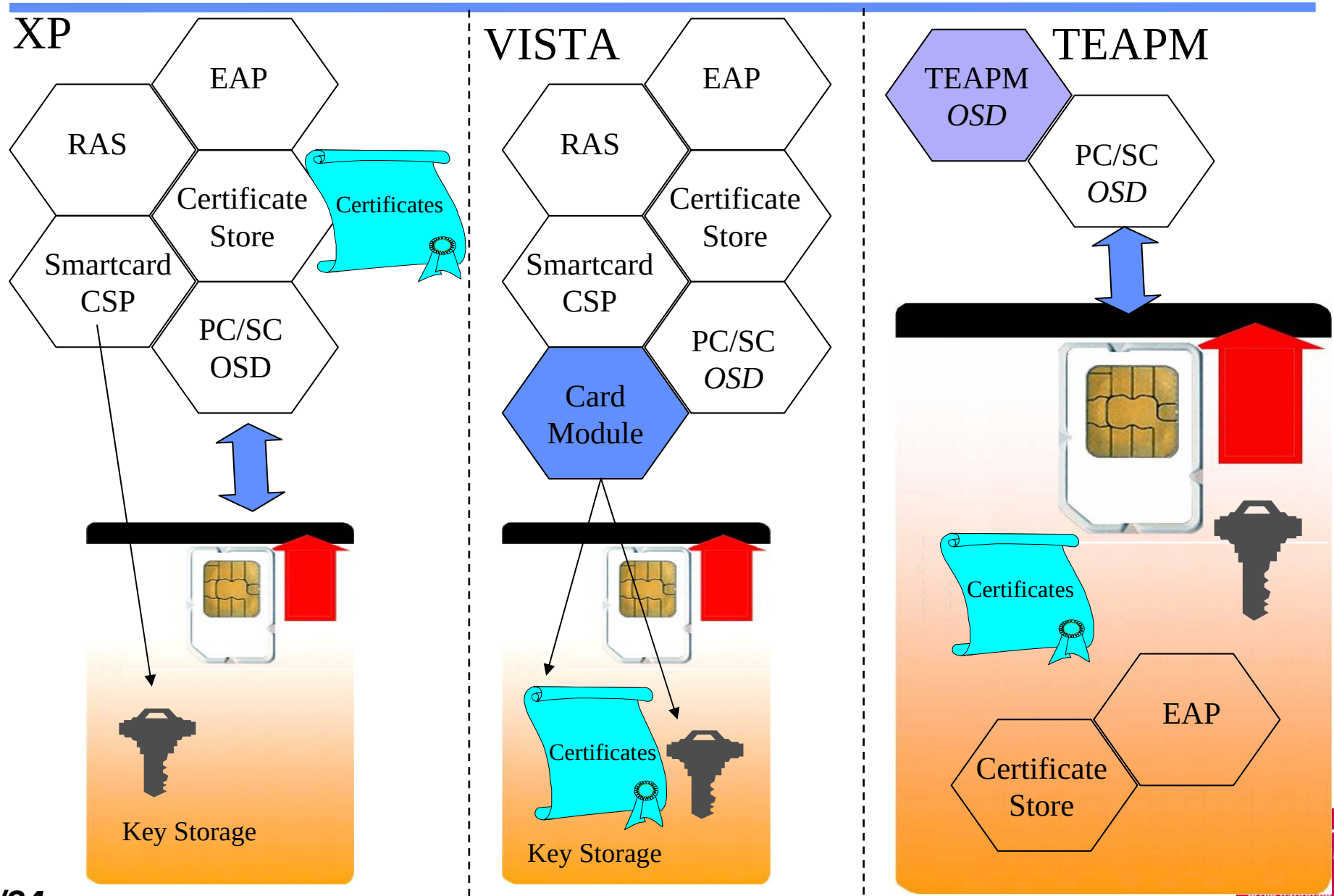


1.4 The TEAPM in a Wi-Fi context



*OSD: Operating System Dependant

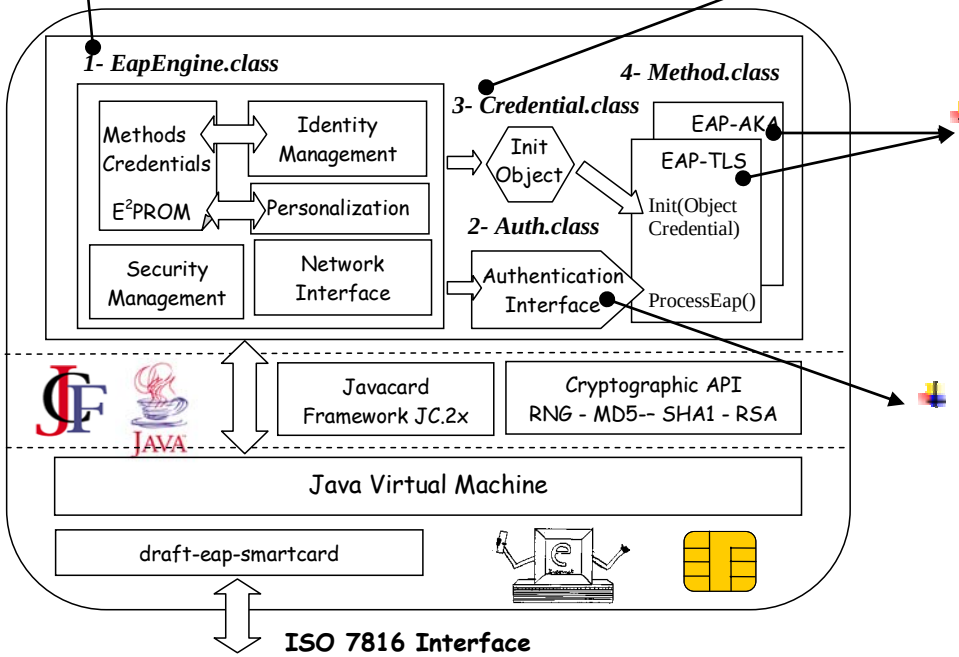
1.5 TEAPM versus Windows



2. OpenEapSmartcard
& Performances Issues

2.1 An Open Platform for EAP support in smartcard

- 1- The EapEngine manages several methods and/or multiple instances of the same one. It implements the EAP core, and acts as a router that sends and receives packets to/from authentication methods. At the end of authentication process, each method computes a master cryptographic key (*AAA Key*) which is read by the terminal operating system.



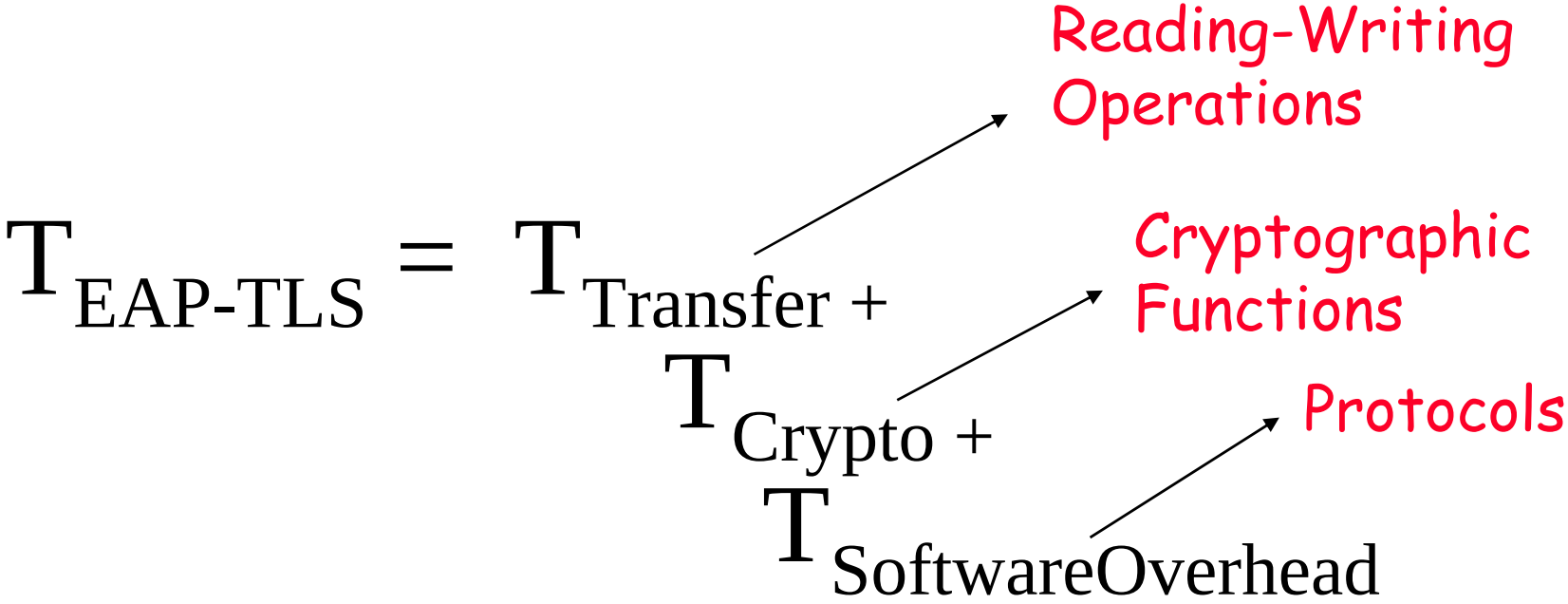
- 3- The Credential objects, are used by to methods, and encapsulate all information required for processing a given authentication scenario.

- 4- The Methods are associated to various authentication scenari. Once initialized, the selected method analyses each incoming EAP request and delivers corresponding response.

- 2- The Authentication interface defines all mandatory services in EAP methods, in order to collaborate with the EapEngine. The two main functions are *Init*() and *Process-Eap*().
  - First initializes method and returns an Authentication interface;
  - Second processes incoming EAP packets. Methods may provide additional facilities dedicated to performances evaluations.

2.2 Performances Issues

- **3xT analysis**
  - Data Transfer
  - Cryptographic Operations
  - Software Overhead

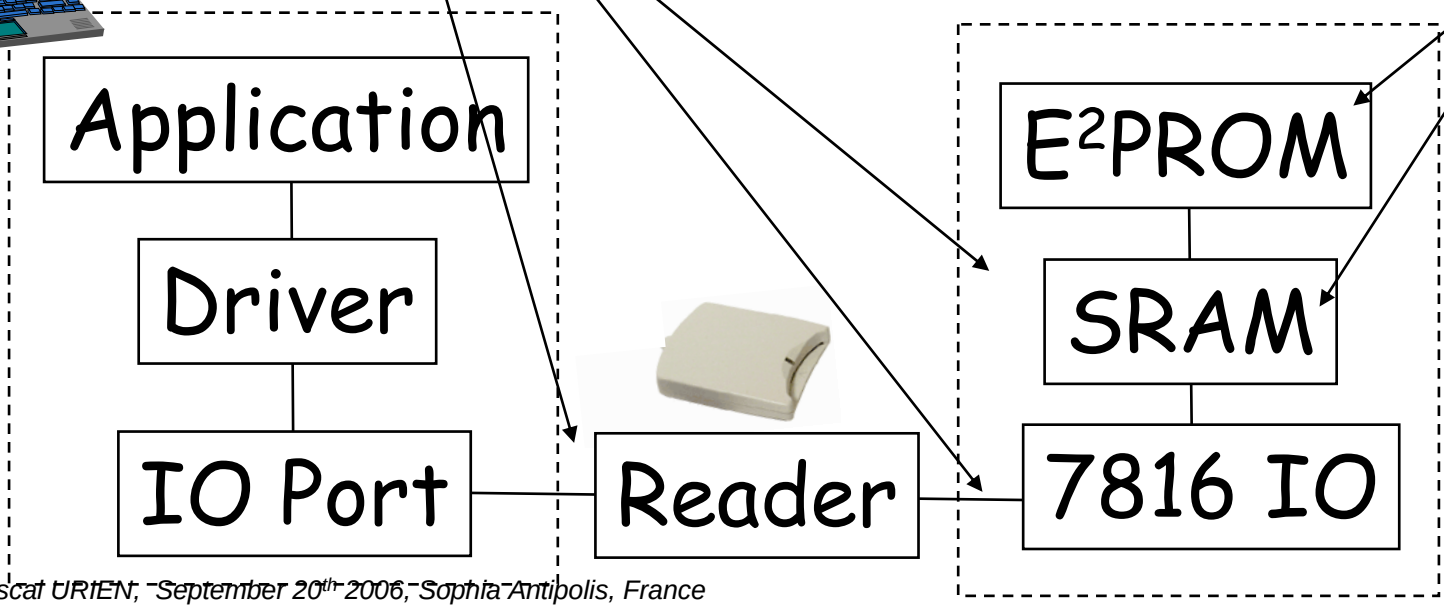


2.2.1 $T_{Transfer}$

- **In protocols dealing with X.509 certificates like EAP-TLS, several kilobytes (typically 3600 bytes) of data are sent/received to/from the smartcard. Due to the lack of RAM memory, these information are written or read in the non-volatile memory (E^2PROM, flash memory,…)**

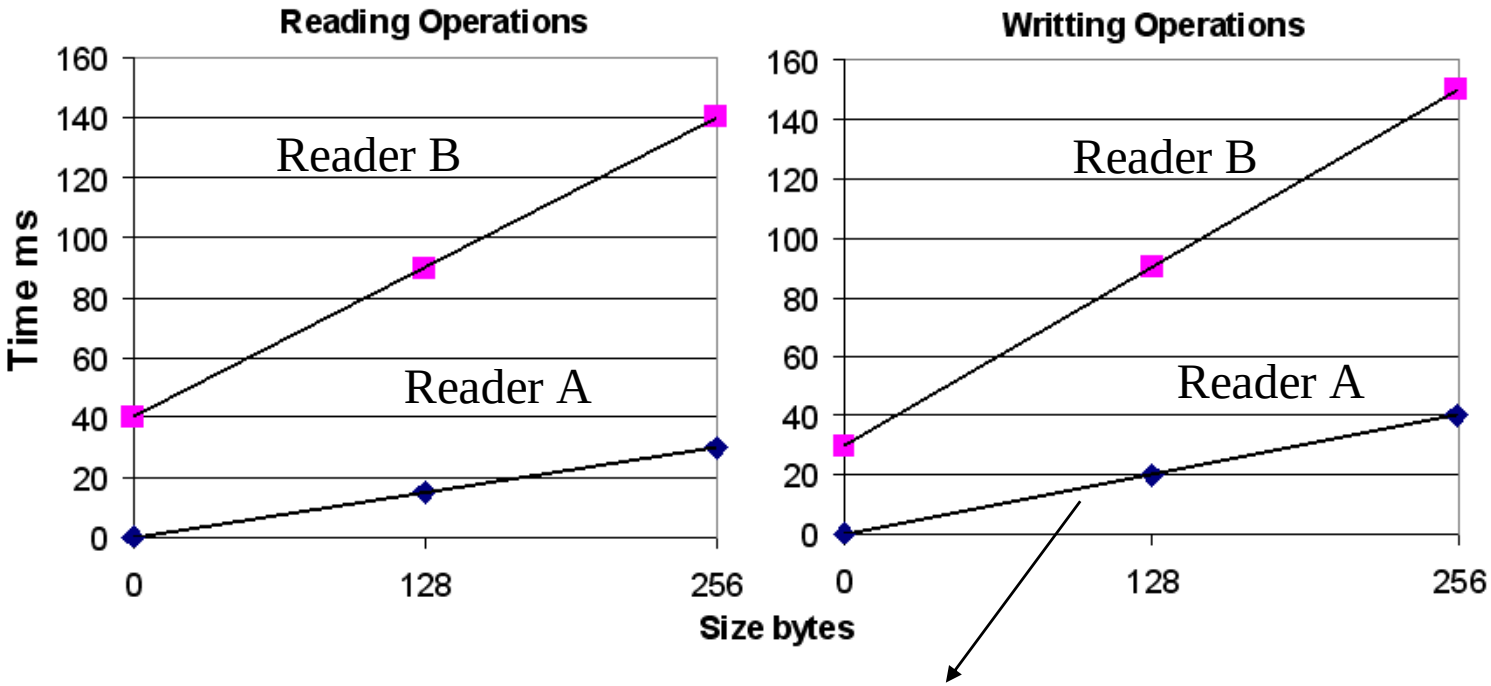
$$T_{Transfer} = T_{TransferReader} + T_{TransferSmartcard} + T_{SoftwareOverhead} + T_{MemoriesAccesses}$$

Docking station



TEAPM

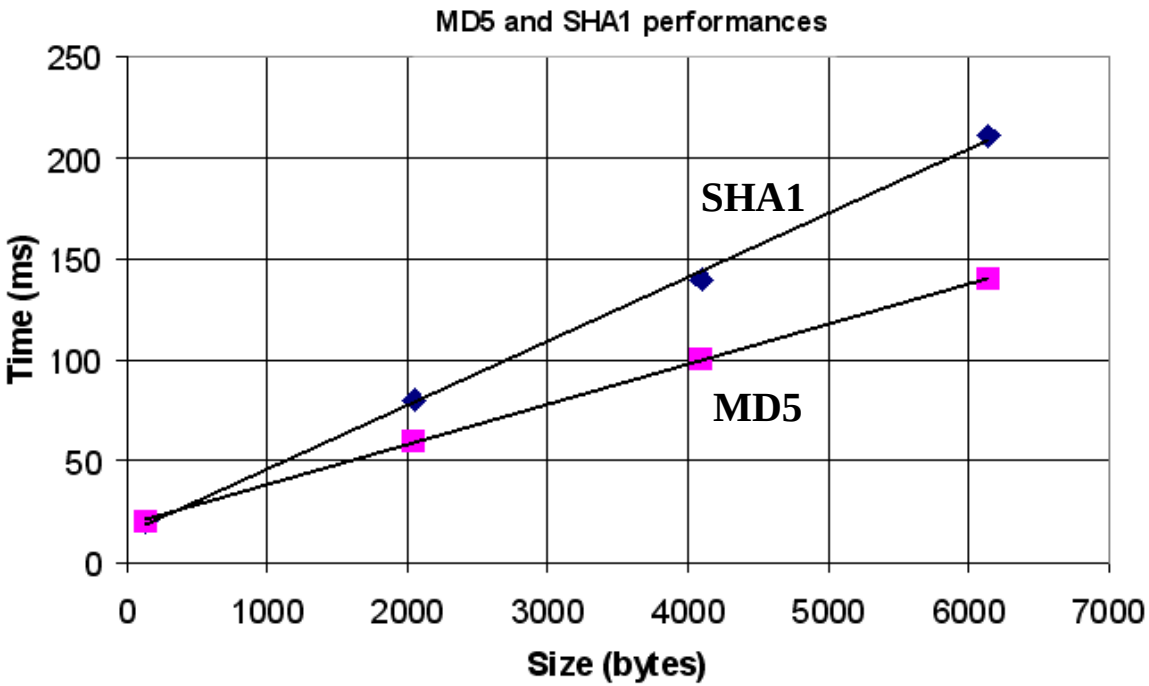
2.2.2 Example of Reading-Writing Operations



0,15 ms/byte (50 Kbits/s)

$T_{Transfer}$ = 2600 x 0,15 = 390 ms

2.2.3 Cryptographic Operations



$T_{Digest} = \frac{1}{2}(T_{MD5} + T_{SHA1})$
$= 11{,}8$ ms/bloc)

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

$T_{RSA} = T_{PubKD} + T_{PubKE} + T_{PrivKD} = 890$ ms

$$T_{Crypto} = T_{RSA} + 532 \times T_{Digest} = 1850 \text{ ms}$$

2.2.4 Summary of performances

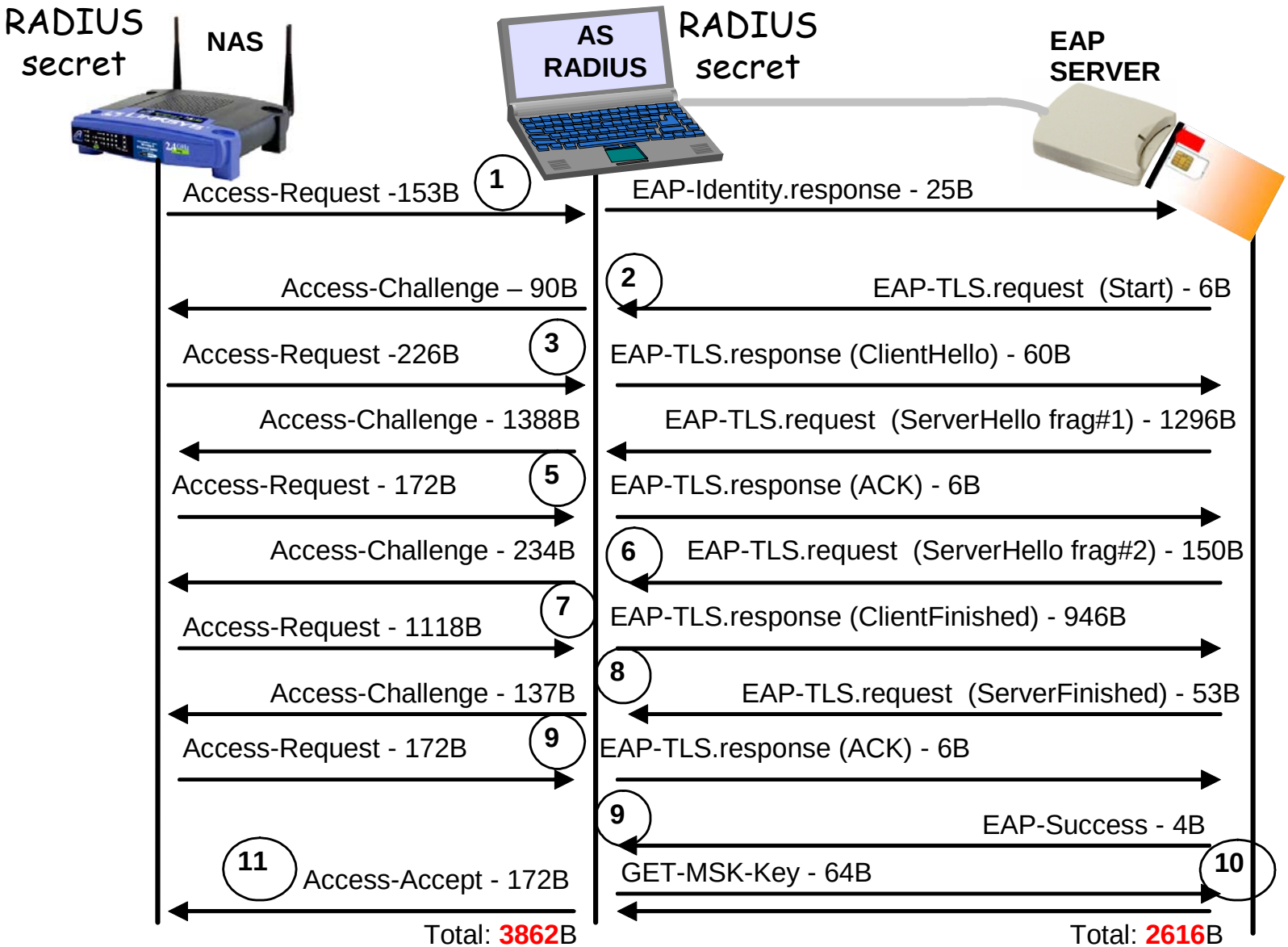
- **$T_{EAP-TLS}$ = 5300 ms**
  - $T_{Other} = T_{EAP-TLS} - T_{Transfer} - T_{Crypto}$
    $= 5300 - 400 - 1850$
    $= 3050ms$
- **As a conclusion TEAPMs spend**
  - 0,4s (08%) in data exchange with the docking station.
  - 1,9s (35%) in cryptographic APIs,
  - 3,0s (57%) in other operations realized by Java software.

3. Smartcard enabled RADIUS server

3.1 Classical RADIUS Server

- **RFC 2865, "Remote Authentication Dial In User Service (RADIUS)", 2000**
  - Two entities
    - The *Network Access Server* (NAS) .
    - The *Authentication Server* (AS).
  - In a telephony context the NAS is running in a *Point Of Presence* (POP) , while in Wi-Fi applications it runs in *Access Points* (AP), and blocks all frames that are sent/received by unauthenticated users.
- **RADIUS works over an UDP/IP stack, and therefore RADIUS messages are routable through the Internet.**
- **Mainly four types of messages**
  - Access-Request, Access-Challenge, Access-Reject, Access-Success
- **RADIUS in IEEE 802.11x context**
  - Clients (called supplicants) are authenticated before allocations of their IP addresses.
  - Authentication messages (EAP) are exchanged between user and NAS over PPP or LAN frames. These messages are encapsulated in RADIUS packets exchanged between NAS and AS entities.
- **RADIUS security is based on a shared secret (the *RADIUS secret*) shared between the NAS and the AS**
  - Cryptographic procedures use MD5 and HMAC-MD5

3.2 Overview of RADIUS Sessions



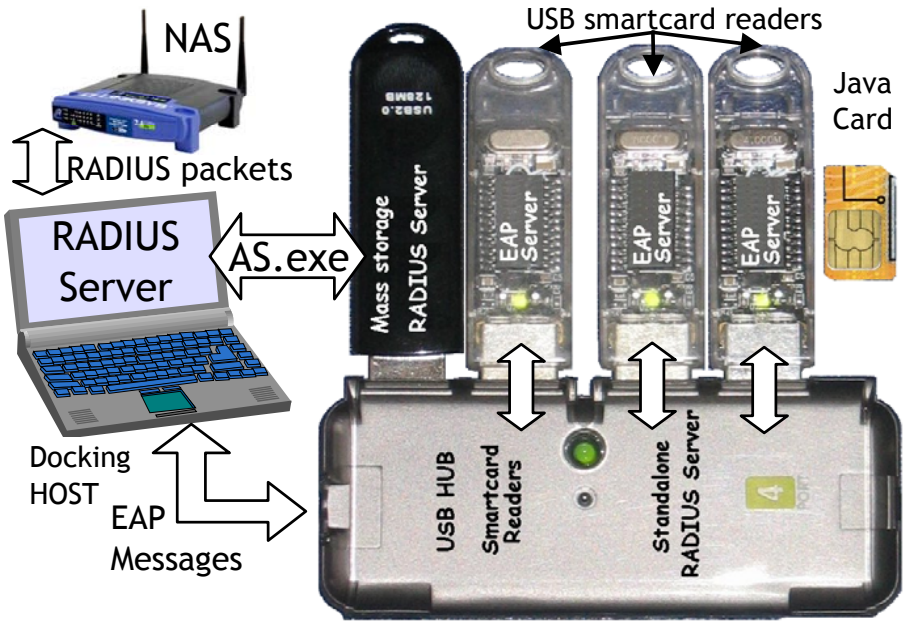
3.3 Benefits of smartcard enabled RADIUS server

- **We believe that EAP server smartcards enhance the RADIUS security, specially in EAP-TLS case for the following reasons,**
  - The server private key is securely stored and used by the smartcard.
  - The client's certificate is autonomously checked by the EAP server.
  - If the EAP client also runs in a smartcard, the EAP session is then fully processed by a couple of tamper resistant devices, working as *Secure Access Module* **(SAM),** a classical paradigm deployed in highly trusted architectures.

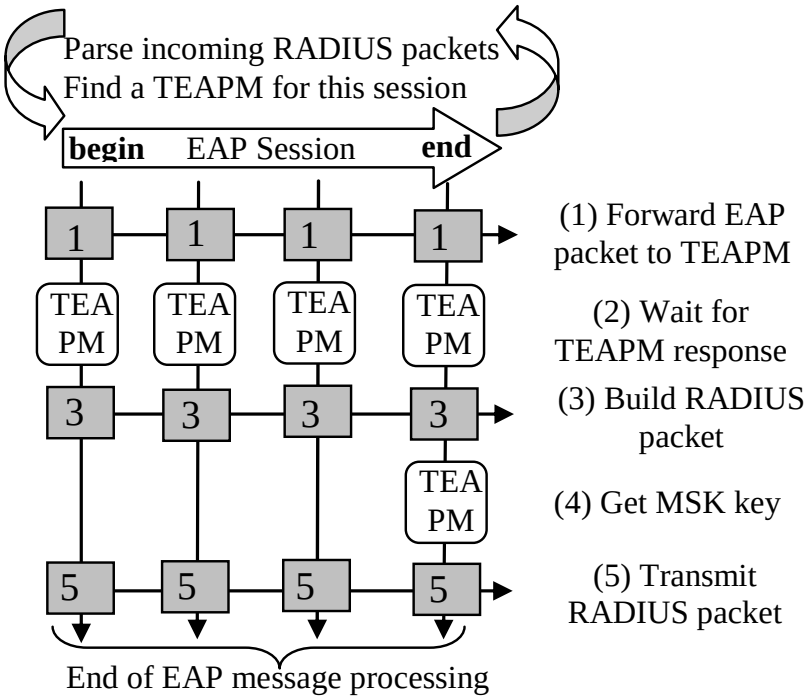
3.4 Smartcard Enabled RADIUS server

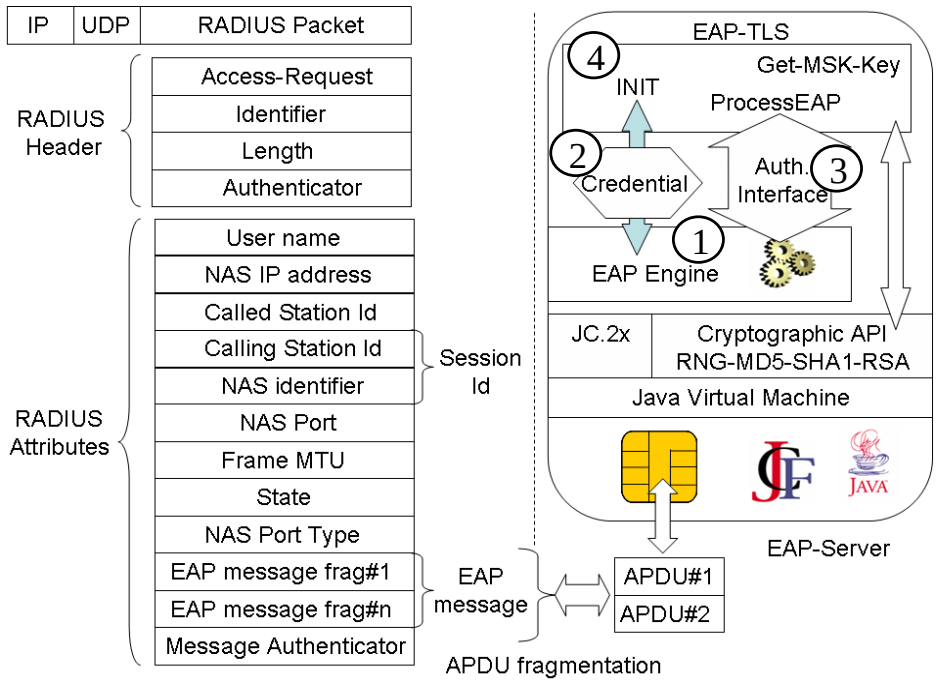
- **Two components**
- **A RADIUS authentication server, running in a docking host.**
  - It offers the Ethernet connectivity and IP services. It receives and sends RADIUS packets over UDP sockets.
  - It builds or parses RADIUS messages, handles the RADIUS secret, checks or generates authentication attributes. EAP messages, transported by RADIUS payloads are forwarded to smartcards, running EAP-Servers.
- **EAP servers.**
  - Each smartcard runs an EAP-server, and fully handles an EAP-TLS authentication procedure.
  - Each component stores an unique X509 certificate and its associated RSA private key.
  - It computes EAP responses and produces EAP requests.
  - At the end of a successful authentication session, a MSK is calculated and delivered to the RADIUS entity.

- **EAP sessions**
  - An EAP session is a set of messages associated to an unique Session-Id value, which is obtained by the concatenation of two values, the NAS-Identifier (RADIUS attribute n°32) and the Calling-Station-Id (the client's MAC address, corresponding to RADIUS attribute n°31) as follows:
  - Session-Id = NAS-Identifier | Calling-Station-Id



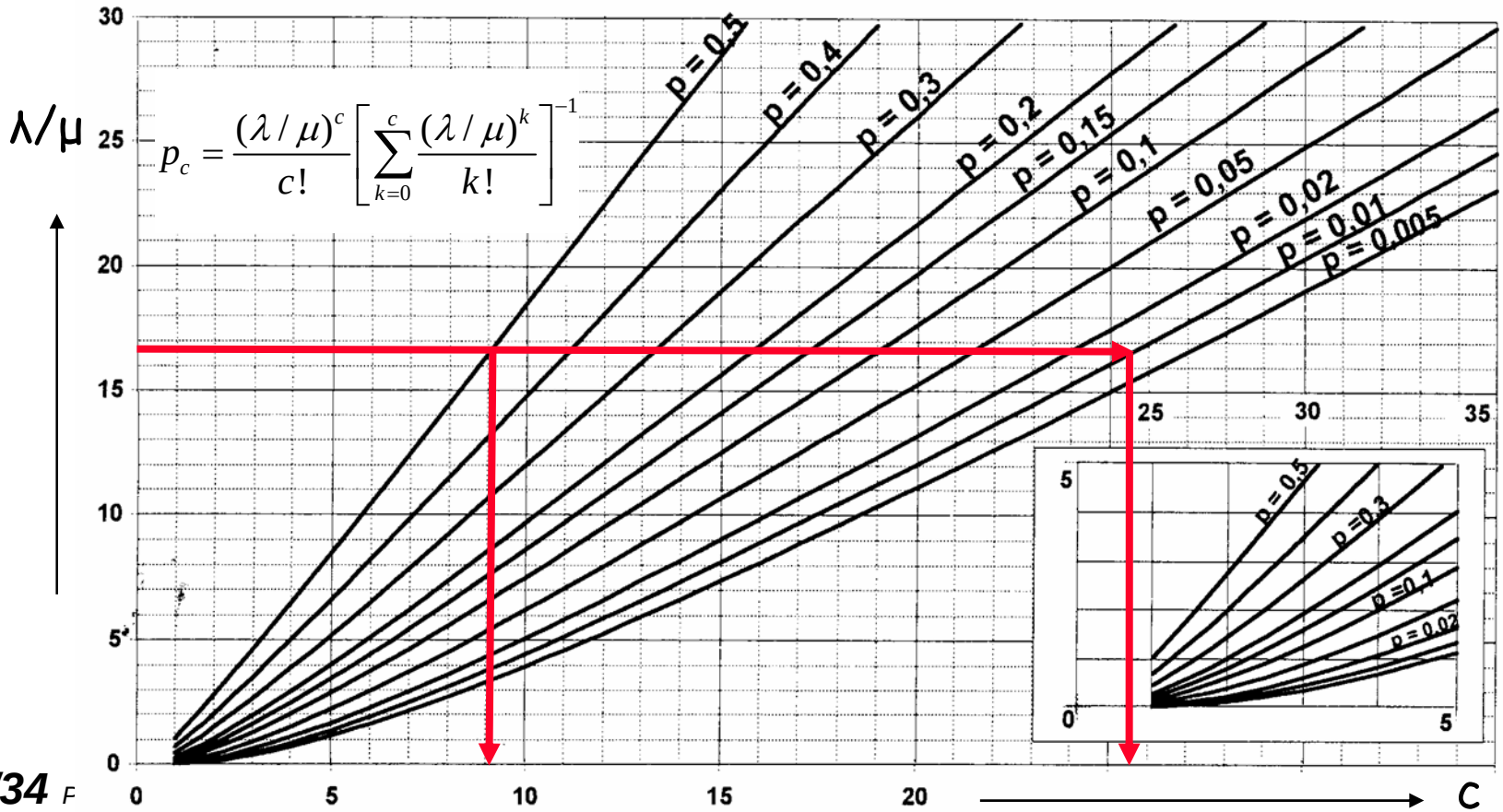
3.5 Implementation Details





3.6 Scalability, privation versus the *Erlang B* law

Pc is the probability of blocking (e.g. a RADIUS packet is silently discarded),
c is the number of EAP servers,
λ is the rate of authentication sessions, and
1/μ the mean time of an authentication session (10s = 5s + 5s)
Let's assume a network with 1000 users, authenticated every 10mn, then λ = 6x1000/3600=1,7 and so λ/μ = 60,000/3600 = 16,7. The probability of blocking (pc) is about 50% with 9 smartcards (c = 9) and only 1% with 21 smartcards (c = 21).

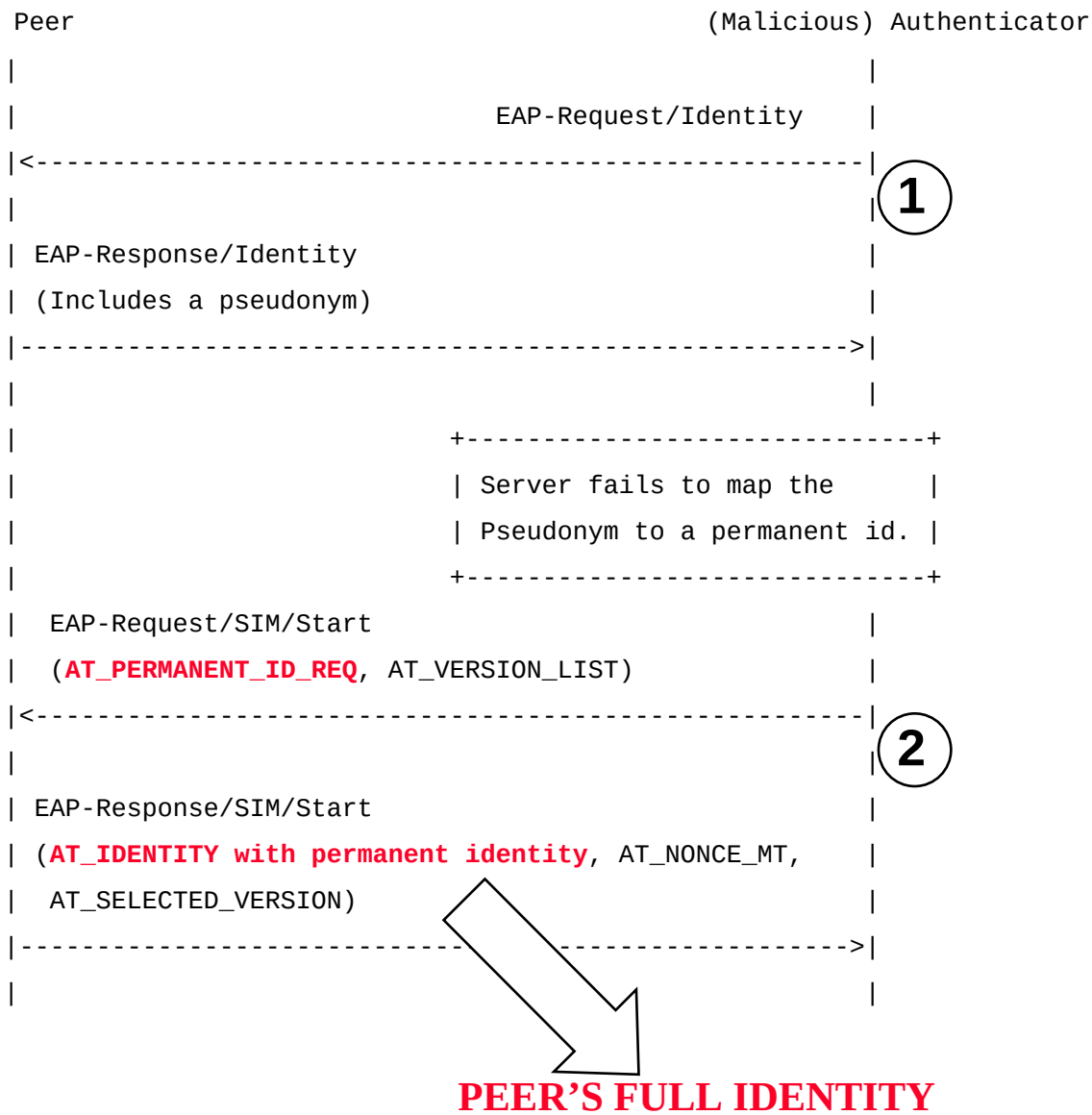


4. Privacy Issues in emerging WLANs

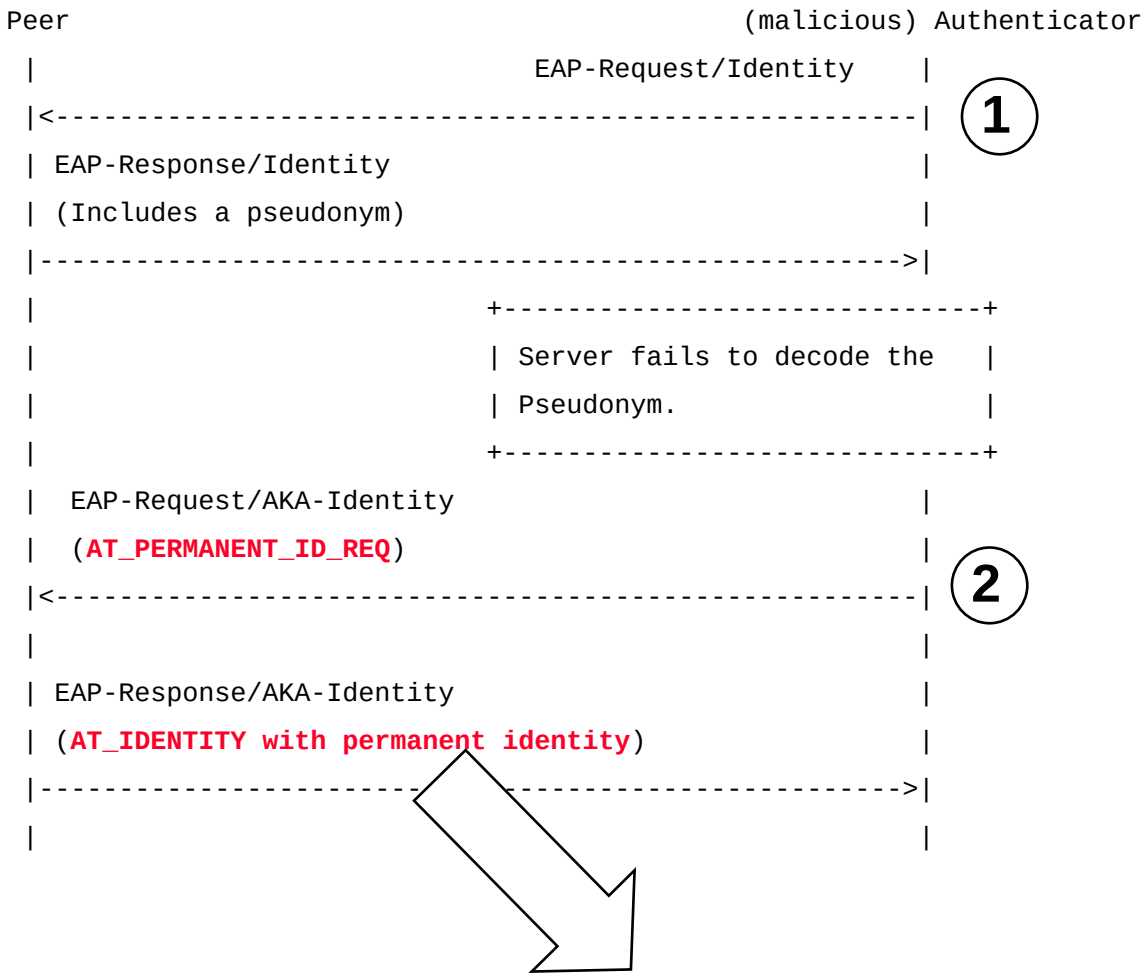
4.1 EAP-METHODS as RFIDs: Identity Leakage

- **The hacker aims at collecting the peer's identity,** *over the air*
  - Passive attack, simple eavesdropping
  - Active attack, EAP packets generation from a malicious Access Point.
- **Number of EAP packets needed for active attacks**
  - EAP-SIM, RFC 4186, 2x requests, without previous knowledge
  - EAP-AKA, RFC 4187, 2x requests, without previous knowledge
  - EAP-TLS, RFC 2716, 3x requests. The knowledge of a *valid* authenticator's certificate is required.

4.2 RFC 4186, EAP-SIM Identity Attack

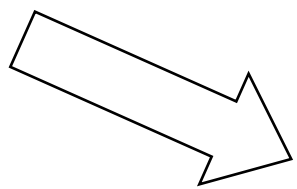


4.3 EAP-AKA, RFC 4187, Identity Attack



4.4 EAP-TLS, RFC 2716, Identity Attack

```
Authenticating Peer    (Malicious) Authenticator
===================    =========================
                           <- PPP EAP-Request/Identity      (1)
  PPP EAP-Response/
  Identity (MyID) ->
                           <- PPP EAP-Request/              (2)
                           EAP-Type=EAP-TLS
                           (TLS Start)
  PPP EAP-Response/
  EAP-Type=EAP-TLS
  (TLS client_hello)->
                           <- PPP EAP-Request/              (3)
                           EAP-Type=EAP-TLS
                           (TLS server_hello,
                            TLS certificate,
                      [TLS server_key_exchange,]
                      [TLS certificate_request,]
                          TLS server_hello_done)
  PPP EAP-Response/
  EAP-Type=EAP-TLS
  (TLS certificate,
   TLS client_key_exchange,
  [TLS certificate_verify,]
   TLS change_cipher_spec,
   TLS finished) ->
```



PEER'S FULL IDENTITY

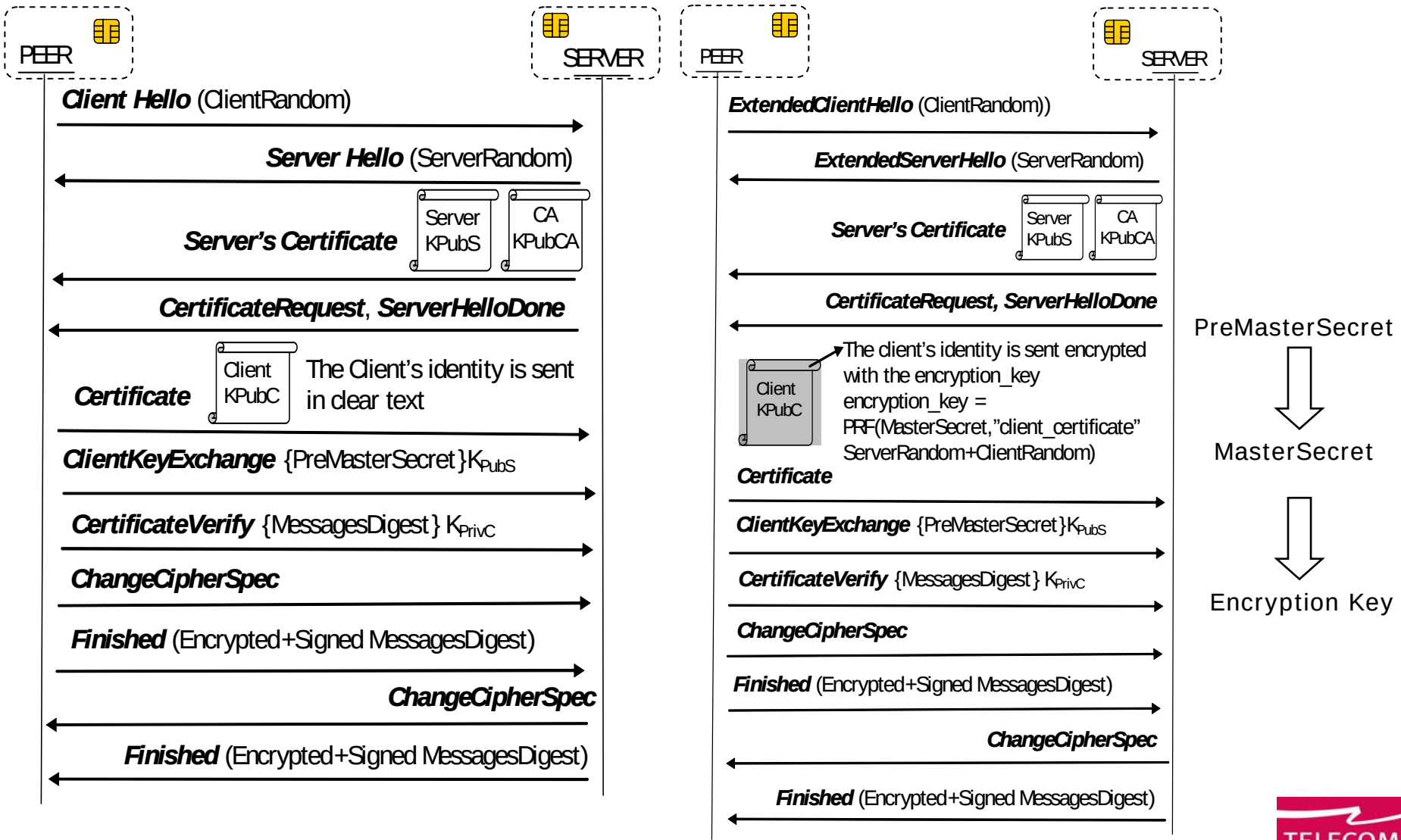
4.5 Classical solutions, but not standardized

- **Establishment of a first protected channel, that secures the peer's identity**
  - Asymmetric protected channel
    - Protected EAP Protocol (PEAP) Version 2, daft-josefsson-pppext-eap-tls-eap-10.txt (2004, *expired*)
    - EAP Tunneled TLS Authentication Protocol Version (EAP-TTLSv1), draft-funk-eap-ttls-v1-01.txt, (2006, *active*)
  - Symmetric protected channel
    - EAP-Double-TLS Authentication Protocol, draft-badra-eap-double-tls-05.txt (2006, *active*)

4.6 Proposed solution for EAP-TLS in TEAPMs

- **Main idea**
  - The peer's certificate is sent encrypted, the encryption key is deduced from the master_secret.
    - encryption_key = PRF(master_secret, "client_certificate",client_random+server_random);
  - In order to allow an EAP-TLS peer to request identity protection exchange, a new extension type is added (TBD) to the Extended Client and Server Hello messages.
  - The 'extension_data' field of this extension contains a list of encryption algorithms supported by the client, ordered by preference.
  - If the server is willing to accept using the extension, the client and the server negotiate the symmetric algorithm that will be used to encrypt/decrypt the client certificate.
  - At the end of the hello phase, the client generates the pre_master_secret, encrypts it under the server's public key, and sends the result to the server.
- **Encryption of the peer's certificate**
  - If a stream cipher is chosen, then the peer's certificate is encrypted with the enc_key, without any padding byte.
  - If a block cipher is selected, then padding bytes are added to force the length of the certificate message to be an integral multiple of the bloc cipher's length.

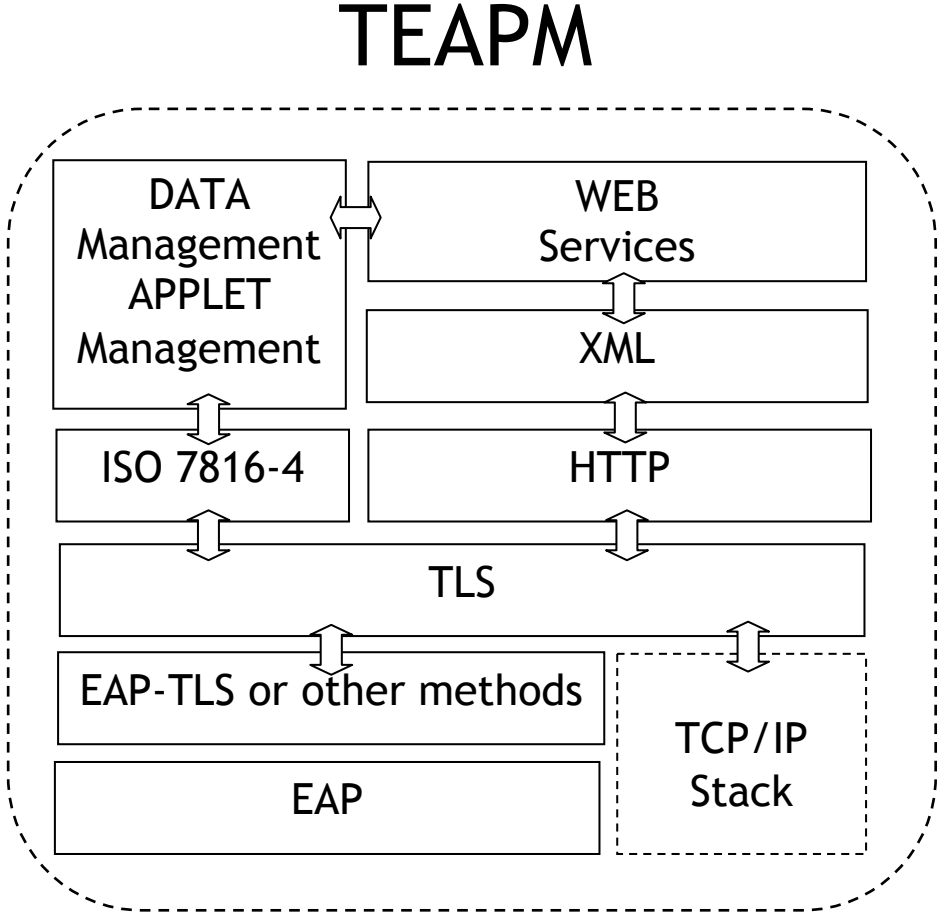
4.7 Identity Protection Dialog



5. TEAPM Management Model

5.1 The TEAPM Management Model

- **Remote administration issues**
  - *Cancellation of credentials,* such as X509 certificates and associated private keys.
  - *Updating of credentials*. There is a need to guaranty continuity or extension of customer's subscriptions.
  - *Downloading of new applications*. Authentication protocols may evolve and include new functionalities.
- **There are several ways to tackle TEAPMs administration.**
  - First deals with legacy aspects and works with classical APDUs transported through protected TLS channels.
  - Second may use an HTTPS transport and implies the definition of a new classes of WEB services, dedicated to smartcard management.



6. Conclusion

Conclusion

- **The TEAPM model is a realistic, but open model for smartcards deployment in IP networks.**
  - Independent of any operating system (Windows, Linux,…).
  - Highly secure authentication servers.
  - Privacy and tracability .
  - Remote administration.
- **We need tamper resistant chips with more computing capacities.**