The OpenEapSmartcard platform

Pr Pascal Urien
ENST Paris

OpenEAP Smartcard
Introduction 1/4: Network ages

Analog networks (Tree age)
- 1876, Alexander Graham Bell invents the telephone. Network architecture is based on point to point links, manually established. Telephone companies control/design all system components.

Digital networks (first grid in the tree)

IP networks (tree is practically a grid)
- 1981. Jon Postel, Steve Crocker, and Vint Cerf create the Internet. IP network works with routers instead of switches. The network is managed by multiple organizations.

Abstract networks (the Boogie Woogie age)
- Wireless IP (1999) and broadband access (2005)
- Network becomes abstract. Accesses, (Fix broadband, Wi-Fi, WiMax...), and services (eMail, Skype, Messenger...) are managed by multiple organizations...
Introduction 2/4: Illustration of Network ages

*Piet Mondrian paintings

“*Red tree”

“*Gray tree”

“*Apple tree”

“*Boogie Woogie”

Alexander Graham Bell (1876).

Digital Networks

Claude Elwood Shannon (1948)

Steve Crocker

Vint Cerf (1981)

Wireless LAN (1999)

Broadband Access

Abstract Networks

Jon Postel,

ember 16th 2005
Introduction 3/4: Questions and answers

What organizations will control the Internet accesses?
- Operators, Cities, Companies, Campus, Individuals...

Why Smartcards?
- Privacy protection.
  - Personal credentials are stored in trusted and secured devices.
  - You are not a computer (Internet Model).
    - In GSM you are a mobile equipment (SIM lockage Model).
    - In TCG you are a computer (TPM model).
- Flexibility.
  - Security is a (javacard) program.
- Scalability.

Why open code?
- World wide networking issues have always been solved by open code
  - TCP/IP and the Internet
  - HTTP and the WEB

A street in Paris
- VoIP is cheap
- VoIP over wireless works today!
Introduction 4/4: Smartcards Benefits

The basic question is:

- Is it acceptable that a network subscriber/user knows its credentials (Certificates, shared secrets, …)
- Company entrances are not secured by passwords. Usually employee use Card-ID or smartcards. As intranet access is a major and vital resource, we believe that smartcards should be deployed for avoiding parking lot attack.
- Smartcard is the only way to split issues security in two different planes:
  - **Network Access Plane.** Symmetric or asymmetric keys are stored in the smartcard. Card holder can't read/modify this value.
  - **User Plane.** Smartcard is unblocked by means of Personal Identifier Number codes, or biometric identification (fingerprint, …).

Are smartcards performances sufficient? - **YES**

- Usually smart cards include crypto-processors that compute the RSA 2048 bits algorithm in less than 0.5s.
- Commercial Javacards memory sizes are around 32-64 KB (available for code byte storage).
  - The size of an X509 certificate is about 1kB
  - As an illustration EAP-TLS applet size (processing EAP and TLS protocols) is around 20KB.
- New generation of smartcards based on FLASH technology, supports one megabyte of memory.
OpenEapSmartcard
What is EAP?

**Extensible Authentication Protocol**

- A framework for authentication in IP networks, which may practically support any authentication scenario.
- It was designed in 1998, for Point To Point authentication issues (ISP access from telephone network)
- Adapted for LANs (IEEE 802.1x) in 2000, then re-used for Wi-Fi

**Standards**

- IETF RFC 2284, "PPP Extensible Authentication Protocol (EAP)", 1998
- IETF RFC 3748, "Extensible Authentication Protocol (EAP)", 2004
- IEEE Standard 802.1X, "Local and Metropolitan Area Networks: Port-Based Network Access Control", 2001
- WPA, "Wi-Fi Protected Access", 2003

**Drafts in progress**

  - EAP is supported for opening IPSEC tunnels.
- IEEE 802.16e (WiMax) supports PKM-EAP.
The EAP Smartcard

What is the EAP smartcard?
- A smartcard that processes EAP messages
- It supports multiple authentication methods

What methods are available?
- EAP-TLS (RFC 2716)
- EAP-AKA (draft-arkko-pppext-eap-aka-15.txt)
- EAP-PSK (draft-bersani-eap-psk-09.txt)

What does it look like?
- It is an application written for Javacards (JC2.1, JC2.2)
- "Write once, run everywhere"
- Smartcard interface (APDUs) is specified in an IETF draft
  - "EAP-Support in smartcard"
    - draft-urien-eap-smartcard-09.txt
- It is an open javacard software
  - OpenEapSmartcard

Web Images Groups News Froogle Local more »

Google Search I'm Feeling Lucky
Basic requirements

The first requirement for OpenEapSmartcard initiative is **flexibility**.
- In the past many security threats were discovered for GSM (COMP 128-1 crack) or 802.11 (WEP crack).
- **Open code** is a good security principle that enables code reviewing; it facilitates study of multiple security architectures; it allows evaluation of deployment costs induced by smartcards price or infrastructure management (secret key management, certificates management, communication reliability,…).

The second requirement is **feasibility**.
- In EAP context an authentication is a suite of requests sent to supplicant (the smartcard in our proposal) that produces responses.
- The default return time trip (RTT) must be less than 30s. Another constraint is the complexity of authentication protocols; code byte is limited by non-volatile memory capacities that currently range between 32 and 128 KB; hopefully this size could quickly evolve toward the MB.

The last requirement is **low costs and multi form factors**.
The software architecture

- **Compatible with standards JC 2.1 or JC 2.2**
- **No proprietary APIs**
- **Four java components**
  - The *EapEngine* which implements the *EAP core*, and acts as a router that sends and receives packets to/from authentication methods
  - An *Authentication* interface that defines all services offered by EAP methods
  - A *Credential* object which stores information needed for method initialization.
  - One or more *Methods* that instantiate authentication scenarios like EAP-TLS or EAP-AKA.
Four services are offered by this module

- **Network interface.** Incoming EAP requests are checked and forwarded to the appropriate method. At the end of authentication process, each method computes a master cryptographic key (PMK) which is read by the supplicant operating system.

- **Identity management.** Smartcard manages several methods and/or multiple instances of the same one. An identities list stores credentials (EAP-ID, X509 certificates, RSA keys, shared secrets) required by embedded methods. This service allows to browse available identities and to select one of them.

- **Security management.** Smart card is protected by two PIN codes, one for its issuer and the other for its holder. In future versions this service could manage fingerprint recognition.

- **Personalization management.** Identity items and PIN codes are controlled and set by smartcard issuers. Some authentication methods like EAP-TLS or EAP-PSK create a secure channel. This protected link may be used for remote management, which could modify “over the air” parameters embedded in tamper resistant chip. This model is closed to standard TS 03.48, which is widely deployed in GSM network for SIM card update purposes.
Authentication Interface and Credential Objects

Authentication Interface

This component defines all services that are mandatory in EAP methods, in order to collaborate with EapEngine.

The two main functions are Init() and ProcessEap().

- The first initializes method and returns an Authentication interface.
- The second processes incoming EAP packets. Methods may provide additional facilities (fct()) dedicated to performances evaluations (or additional services).

Credential Objects

Each method is associated to a Credential Object (Init()) that encapsulates all information required for processing a given authentication scenario.
## Authentication Interface

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fct</code></td>
<td>Method functions</td>
</tr>
<tr>
<td><code>Get_Fct_Buffer</code></td>
<td>Returns a function buffer</td>
</tr>
<tr>
<td><code>Get_Fct_Length</code></td>
<td>Returns a function buffer length</td>
</tr>
<tr>
<td><code>Get_Fct_Offset</code></td>
<td>Returns a function buffer offset</td>
</tr>
<tr>
<td><code>Get_Out_Buffer</code></td>
<td>Returns the response buffer</td>
</tr>
<tr>
<td><code>Get_Out_Length</code></td>
<td>Returns the response buffer length</td>
</tr>
<tr>
<td><code>Get_Out_Offset</code></td>
<td>Returns the response buffer offset</td>
</tr>
<tr>
<td><code>Init</code></td>
<td>Method Initialization</td>
</tr>
<tr>
<td><code>IsFragmented</code></td>
<td>Fragmentation in progress</td>
</tr>
<tr>
<td><code>IsLongFct</code></td>
<td>Indicates that the response of a function is stored in a private buffer</td>
</tr>
<tr>
<td><code>IsLongResponse</code></td>
<td>Indicates that the response of the method is stored in a private buffer</td>
</tr>
<tr>
<td><code>process_eap</code></td>
<td>Method Processing</td>
</tr>
<tr>
<td><code>reset</code></td>
<td>Resets the method</td>
</tr>
<tr>
<td><code>status</code></td>
<td>Gets the method status</td>
</tr>
</tbody>
</table>

**Parameters**

- `apdu`: incoming APDU
- `in`: buffer associated to the incoming APDU
- `inlength`: P3 value
- `credentials`: java.lang.Object
- `in`: incoming APDU buffer
- `inlength`: length of the incoming APDU
Methods

Each authentication scenario is processed by a specific Method class. Once initialized, it analyses each incoming EAP request and delivers corresponding response. The number of embedded methods is limited by the smartcard non volatile memory (E²PROM) size.

Performances

- EAP-TLS SC best known performance < 10s
- EAP-PSK SC best known performance < 2s
- EAP-AKA SC best known performance < 10s

Example of EAP-TLS smartcard performance

<table>
<thead>
<tr>
<th>Component</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other (Software Overhead)</td>
<td>6.8s</td>
</tr>
<tr>
<td>DataTransfer</td>
<td>1.4s</td>
</tr>
<tr>
<td>CryptoAPI</td>
<td>2.2s</td>
</tr>
<tr>
<td>Total</td>
<td>10.4s</td>
</tr>
</tbody>
</table>
XP Integration

OpenEapSmartcard and Windows

- User's PIN
- Identity Setting
- Identity Discovery
- EAP-TLS Type
- EAP-Javacard Application Identifier
- EAP Provider DLL Smartcard Interface
EAP-TLS smartcards
EAP-TLS in Windows platforms

Client’s PC is configured with
- CA X509 certificate
- Client X509 certificate
- Client RSA private key
- Private Key is protected by the client’s password

Wi-Fi Link

CA Certificate

Server Certificate

Authentication Server (RADIUS)

Access Point

Client Certificate

CA Certificate

Client's PC is configured with
- CA X509 certificate
- Client X509 certificate
- Client RSA private key
- Private Key is protected by the client’s password

RASTLS.DLL (EAP-TLS software)

Crypto API

Certificate Context
- Certificate (signed by CA)
- Properties

Certificates Management

Keys Management

CSP

RSA Keys

User’s BLOB, protected by DPAPI
Data Protection API

RSA Private Key

Password → NT Key → TP

Smartcard

RADIUS
EAP-TLS smartcards benefits

**Security enhancement**
- All protocol is processed by the smartcard.
- No Trojan horse attack.
- Secure private key storage and use.

**Mobility enhancement**
- No PC personalization
- Certificates are stored and managed by the smartcard.

**Better Key protection**
- No PKCS12 certificates (protected by passwords) issuance / installation.

**Do you trust your computing platform?**
EAP-TLS with OpenEapSmartcard

Client's PC is configured with EAPCARD.DLL
- EAPCARD DLL is a transparent bridge that forwards EAP message to/from the smartcard

All EAP messages are processed by the EAP smartcard

Client's PC

Wi-Fi Link

Access Point

Authentication Server (RADIUS)

Server Certificate

CA Certificate

RSA Private Key

(EAP-TLS software)

CA Certificate

Client Certificate

(EAP-TLS software)
Conclusion

- EAP-TLS smartcards, based on OpenEapSmartcard platform work in real Wi-Fi networks

- Some improvements are necessary
  - Smartcard IO speed enhancement
  - Smartcard computing capacity enhancement (Moore’s law)
  - New javacard APIs (reduction of software overhead)

- Visit the OpenEapSmartcard web site
  - http://www.enst.fr/~urien/openeapsmartcard