Security applications with Java Card

SAR 2003, Nancy

Julien SIMON

j.simon@oberthurcs.com

Outline

1. Introduction
2. Java Card overview
3. WAP security
4. IP security
5. 802.11 security
6. Q&A

Introduction

Oberthur Card Systems

- No. 1 supplier of MasterCard and Visa payment cards worldwide.
- No. 3 supplier of 2G/3G cards worldwide.
- First to apply Java technology to the SIM card (1998).
- Please refer to www.oberthurcs.com for more information.

Speaker

- 3 years at OCS R&D.
- Mobile Communications Development Manager.
- In a previous life, lots of time spent in TCP/IP and kernel code (Mach / Chorus / *nix) : hence, a strong interest in computer (in)security…

Java Card overview

Architecture

Language, VM, API

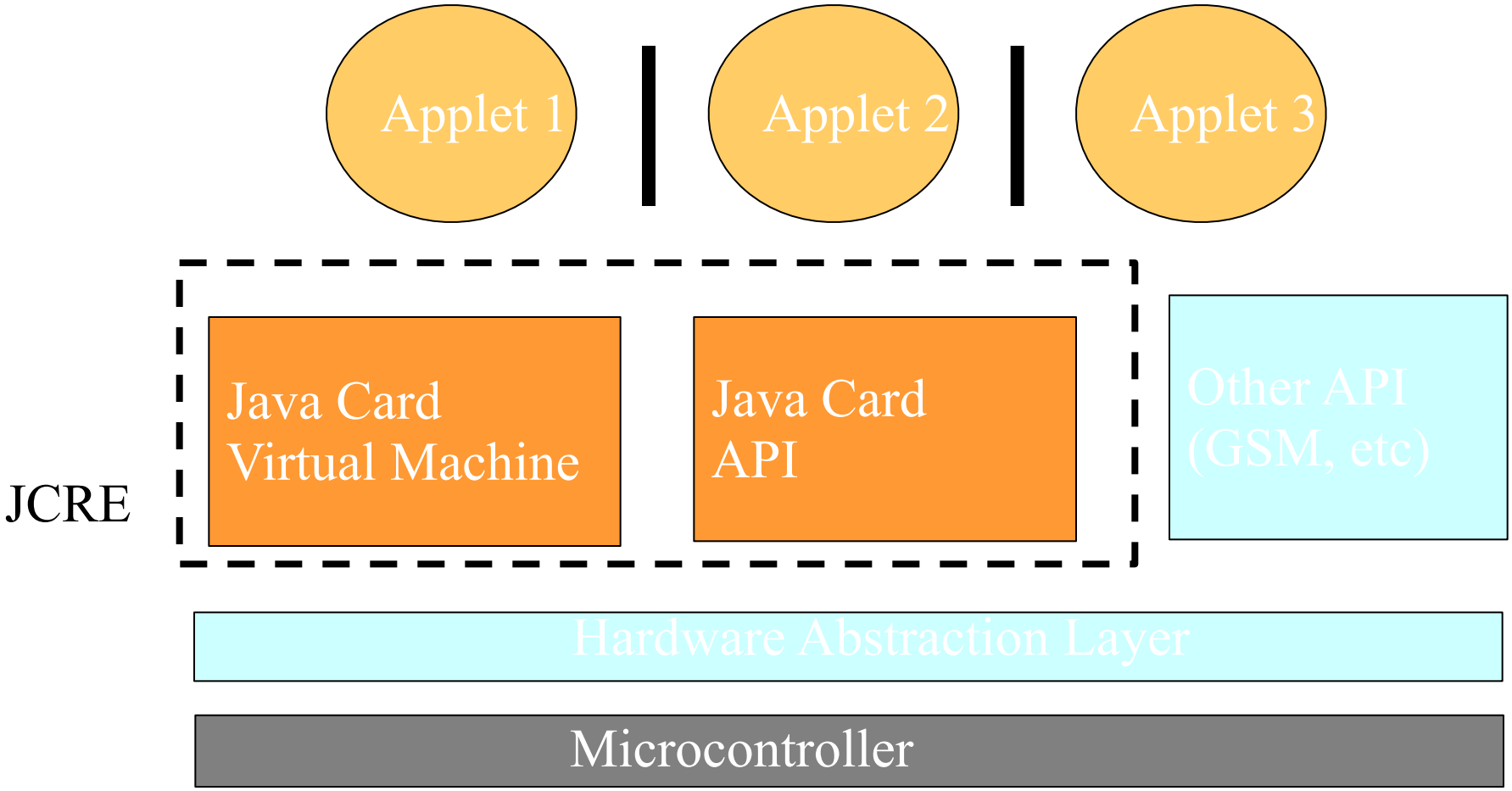
Java Card

- Software standard initiated by Sun Microsystems in October 1996.
- JC is now maintained by the **Java Card Forum**.
- JC defines an environment allowing Java applications to to run on a microprocessor smartcard : **Java Card Runtime Environment** (JCRE) :
- Java Card is nice because :
  - **It allows faster and easier development than native code.**
  - **It has all the benefits of OOD / OOP.**
  - **It is portable at source and binary level.**
  - **It allows applications to be loaded after the smartcard has been issued.**
- A well-designed Java Card is a very safe foundation :
  - **Common Criteria EAL 4+ evaluation obtained by OCS in 2002.**
  - **State-of-the art cryptography, protected against SPA/DPA/DFA attacks.**

Java Card Runtime Environment

- The JCRE includes :
  - The Java Card Virtual Machine,
  - The Java Card API,
  - A basic application installer.
- It's implemented in ROM by the smartcard issuer.
- Its behavior is defined by the **Java Card Runtime Environment Specification**.
- Versions :
  - 2.1 (May 2000).
  - 2.2 (May 2002).

Java Card architecture



Language

- JC supports most features of the Java language :
  - *Packages,*
  - Dynamic object creation (new),
  - Virtual methods, Inheritance, Interfaces,
  - Exceptions,
  - Etc.
- The following types are not supported
  - char,
  - long, float and double,
  - Multi-dimensional arrays.
- The int type is optional.

Java Card Virtual Machine

- The JCVM has a classic architecture :
  - It runs bytecode on an operand stack.
  - JC bytecode is a subset of Java bytecode
  - E.g. no int related bytecode : iload, istore, etc.
- Compared to the JVM, the JCVM is very simplified :
  - No on-demand class loading : all required classes must present on the card.
  - No threads, etc.
- The JVCM also has specific features (transactions, inter-applet communication).
- The behavior of the JCVM is defined by the **Java Card Virtual Machine Specification**.

Java Card 2.1 API

- The JC 2.1 API includes four *packages*, defined by the **Java Card 2.1 Application Programming Interfaces Specification**.
- java.lang : minimal Java classes.
- javacard.framework : smartcard-related classes
  - **Communication with the terminal, PIN handling, etc.**
- javacard.security & javacardx.crypto : security classes.
  - **Keys : DES, 3DES, RSA et DSA.**
  - **Crypto objects : KeyPair, MessageDigest, Cipher and Signature.**
- Java Card 2.2 adds Java Card RMI, AES, ECC, *garbage collection*, etc.

Java Card references

- Java Card
  - Specs & JCDK : http://java.sun.com/javacard/
  - Java Card Forum : http://www.javacardforum.org/
  - « Java Card Technology For Smart Cards » , Addison-Wesley, 2000.
- Cryptography
  - RSA Labs : http://www.rsalabs.com/
  - « Cryptographie appliquée » (2ème édition), Bruce Schneier.
  - « Handbook of Applied Cryptography » http://www.cacr.math.uwaterloo.ca/hac/

WAP security

WAP overview

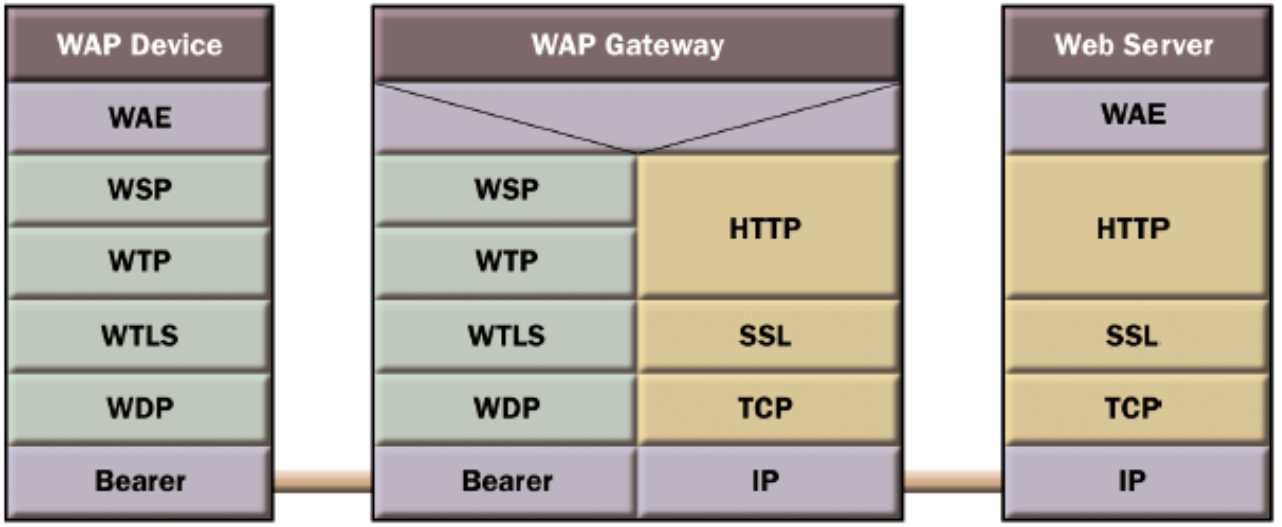
Wireless Identity Module (WIM)

Smartcard WAP browsers

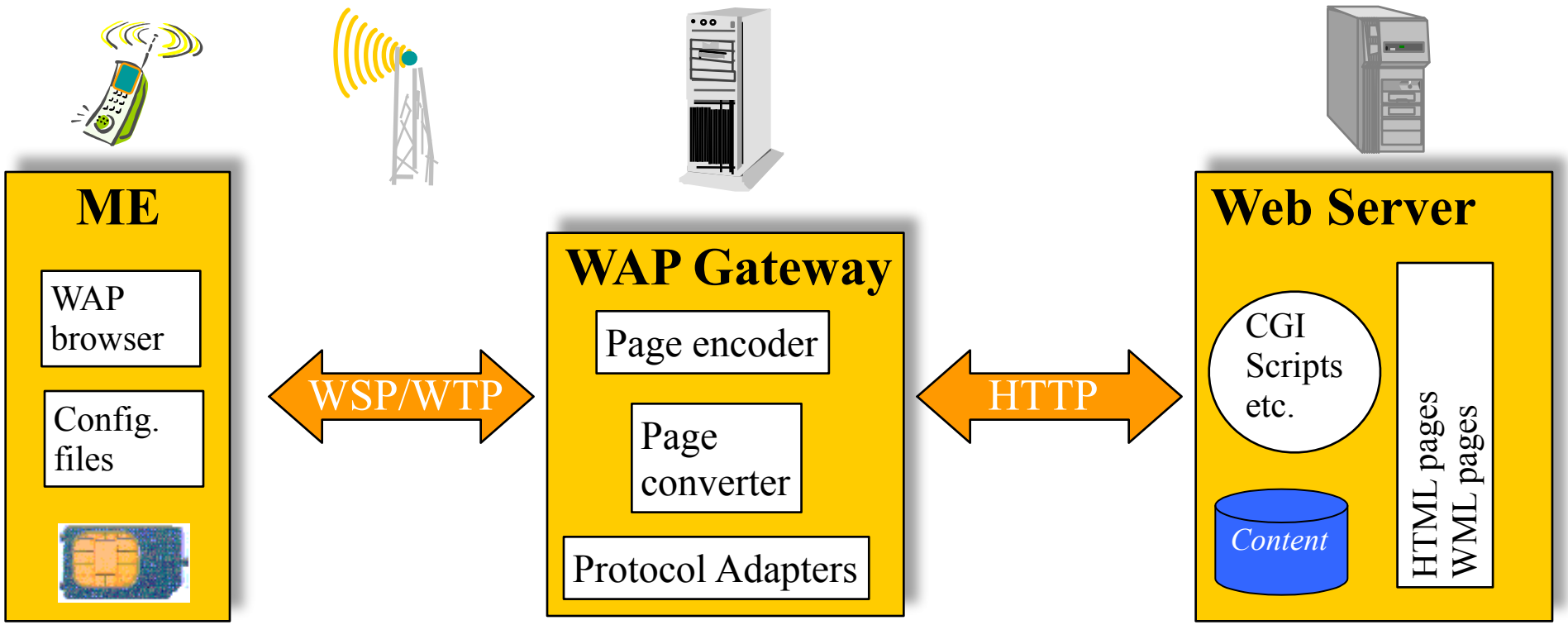
Wireless Application Protocol

- WAP 1.0 was released in 1998 and evolved into 1.3.
- WAP 1.x doesn't support standard Internet protocols and languages : a **WAP gateway** is required.

WAE: Application Environment
WSP: Session Protocol
WTP: Transaction Protocol
WTLS: Transaction Layer Security
WDP: Datagram Protocol
HTTP: Hyper Text Transfer Protocol
SSL: Secure Sockets Layer
TCP: Transmission Control Protocol
IP: Internet Protocol



WAP architecture



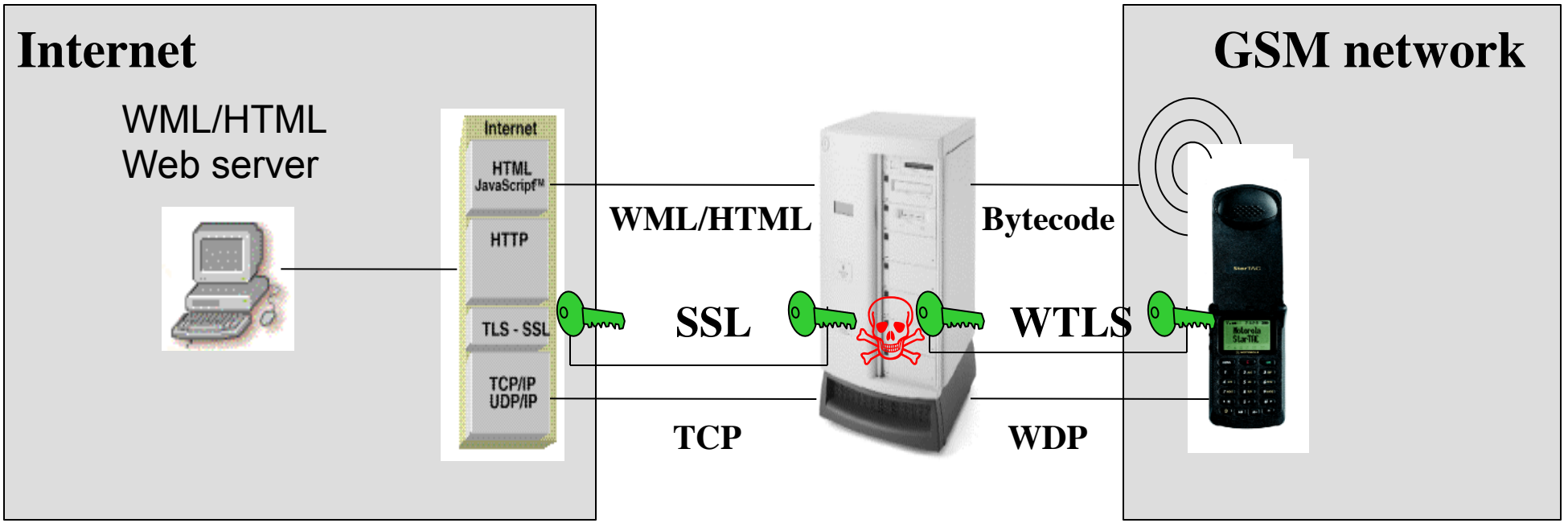
ME = Mobile Equipment
HTTP = Hyper Text Transfer Protocol
WSP = Wireless Session Protocol
WTP = Wireless Transaction Protocol

WML = Wireless Markup Language
HTML = Hyper Text Markup Language
CGI = Common Gateway Interface

Wireless Identity Module

- WAP communication is secured by the Wireless Transaction Layer Protocol (WTLS), which is “equivalent” to SSL.
- WTLS relies on the Wireless Identity Module (WIM) to perform secure operations, such as :
  - **Verification primitives** (PIN operations).
  - **Data Access primitives** (Key/certificate storage in PKCS #15 files)
  - **Cryptography primitives :** Compute Digital Signature, Verify Signature, Hash, Decipher and various key primitives used to setup a WTLS session (Diffie-Hellman, etc).
- The WAP browser also needs the WIM to perform the signText operation (application-level signature).
- The WIM can be implemented as a Java Card applet.

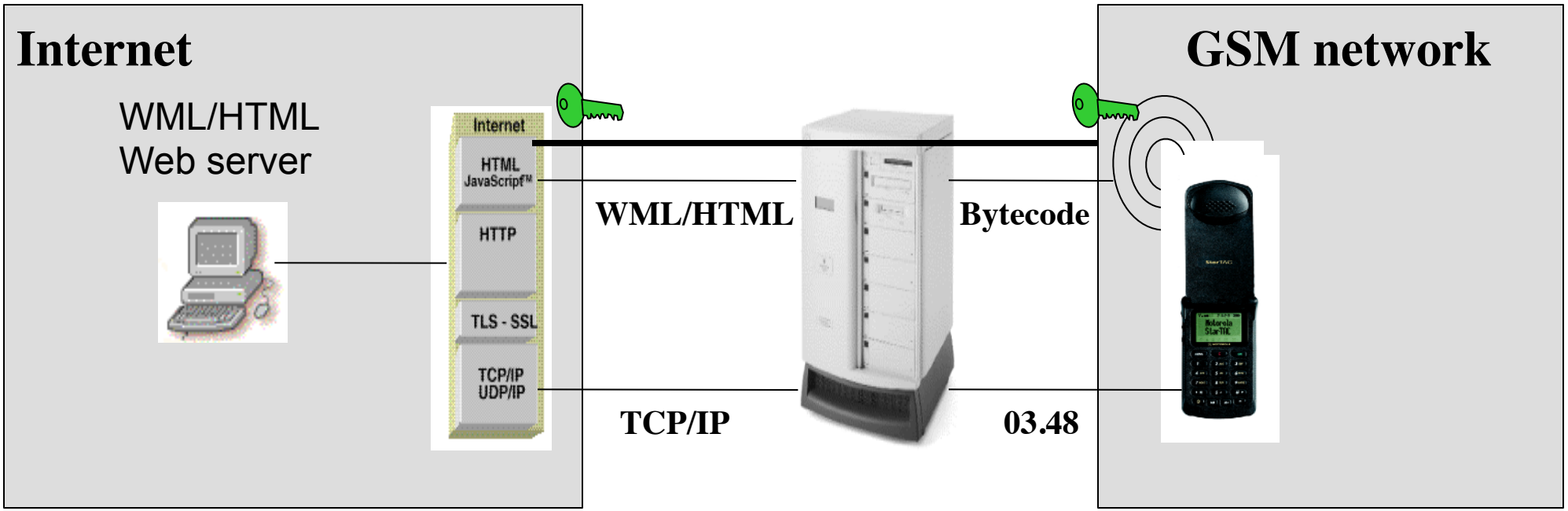
WAP (in)security



There is no end-to-end security, because of the WTLS/SSL gateway. This is the infamous **"WAP gap"** problem.

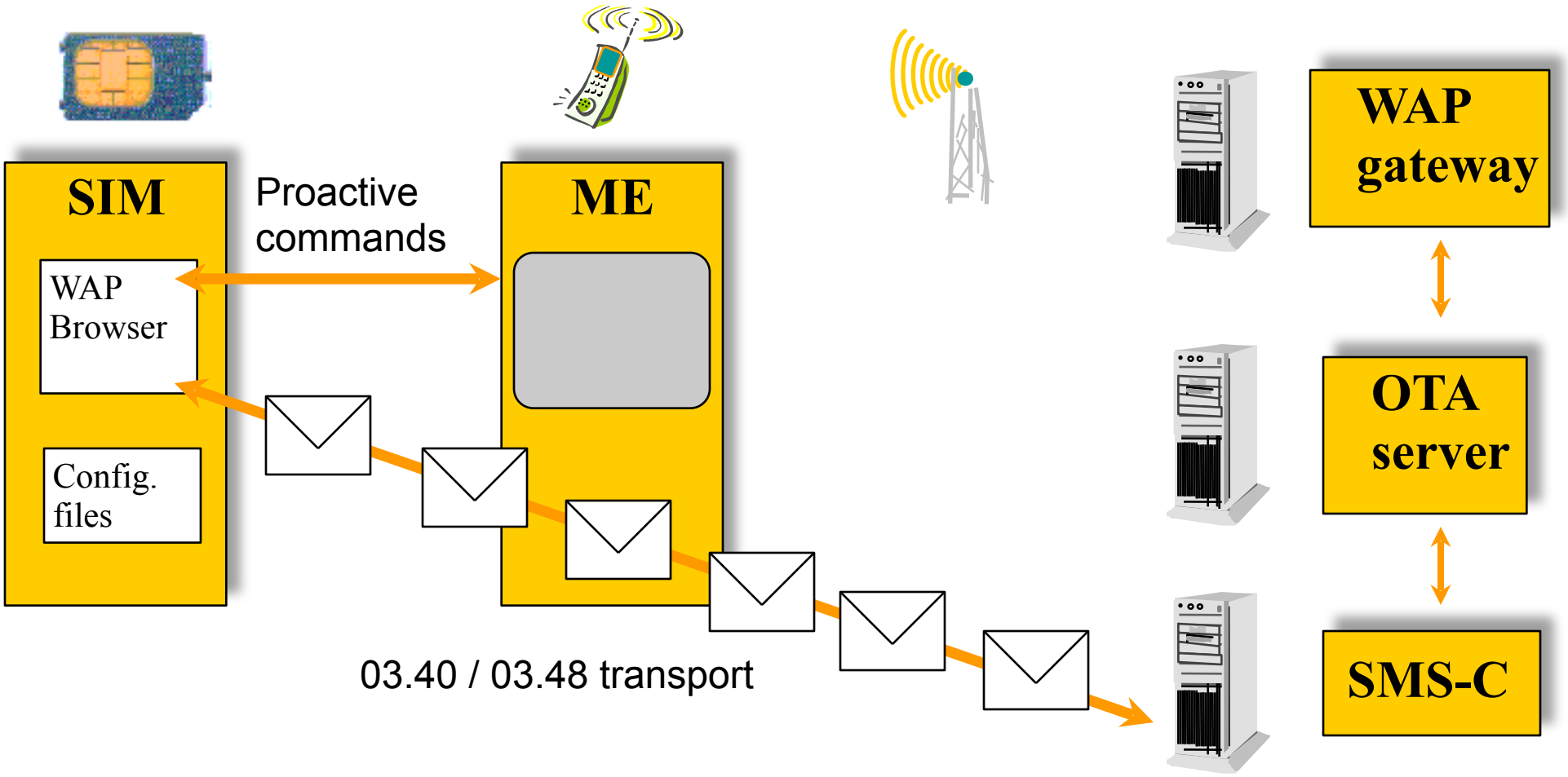
WAP 2.0 solves this by using standard Internet protocols (TLS).

End-to-end security



Data is protected by the application,
i.e. the bytecode is encrypted/decrypted by the **WAP browser** running on the (U)SIM card.

WAP with a smartcard browser



Plug-in architecture

- The core browser can be extended using **plug-ins**.
- A plug-in is a normal SIM Toolkit applet.
  - It extends the WIB by adding new features :
    - *Data storage,*
    - *Cryptography : the WAP gap must be closed !*
  - It may be installed at personalization time.
  - Plug-ins are usually small enough to be installed OTA.
- Applet communication is possible with Java Card :
  - A plug-in **registers** to the WIB using a **Shareable** interface.
  - When the WIB receives the **Plug-In** command, it invokes the plug-in using another **Shareable** interface.

PKI plug-ins

- A smartcard browser usually supports several Public Key Infrastructure (PKI) plug-ins.

- Decryption : **Asymmetric Decryption** (*AD*)
- Digital signature (text) : **PKCS#1** (*P1*)
- Digital signature with time stamp (text) : **PKCS#7** (*P7*)
- Digital signature and non-repudiation (data) : **Fingerprint** (*FP*)

A Band Of Browsers

- SmartTrust **Wireless Internet Browser** (WIB)
- SIM Alliance **S@T browser**.
- 3GPP **USAT Interpreter** :
  - **21.112 & 31.11{2,3,4} Release 5 & 6 : architecture, protocol, core browser.**
  - **31.113 Release 6 : plug-ins.**

- These browsers all support the WML standard (or at least a large subset of it) as well as proprietary extensions (**S@TML**, **SmartTrust WML**, etc). This is similar to the Netscape/IE situation.

WAP references

- USAT Interpreter – http://www.3GPP.org/
- SIM Alliance – http://www.SIMAlliance.org
- SmartTrust – http://www.SmartTrust.com

- PKCS – http://www.RSALabs.com
- WAP – http://www.OpenMobileAlliance.com/
- World Wide Web Consortium – http://www.w3.org/

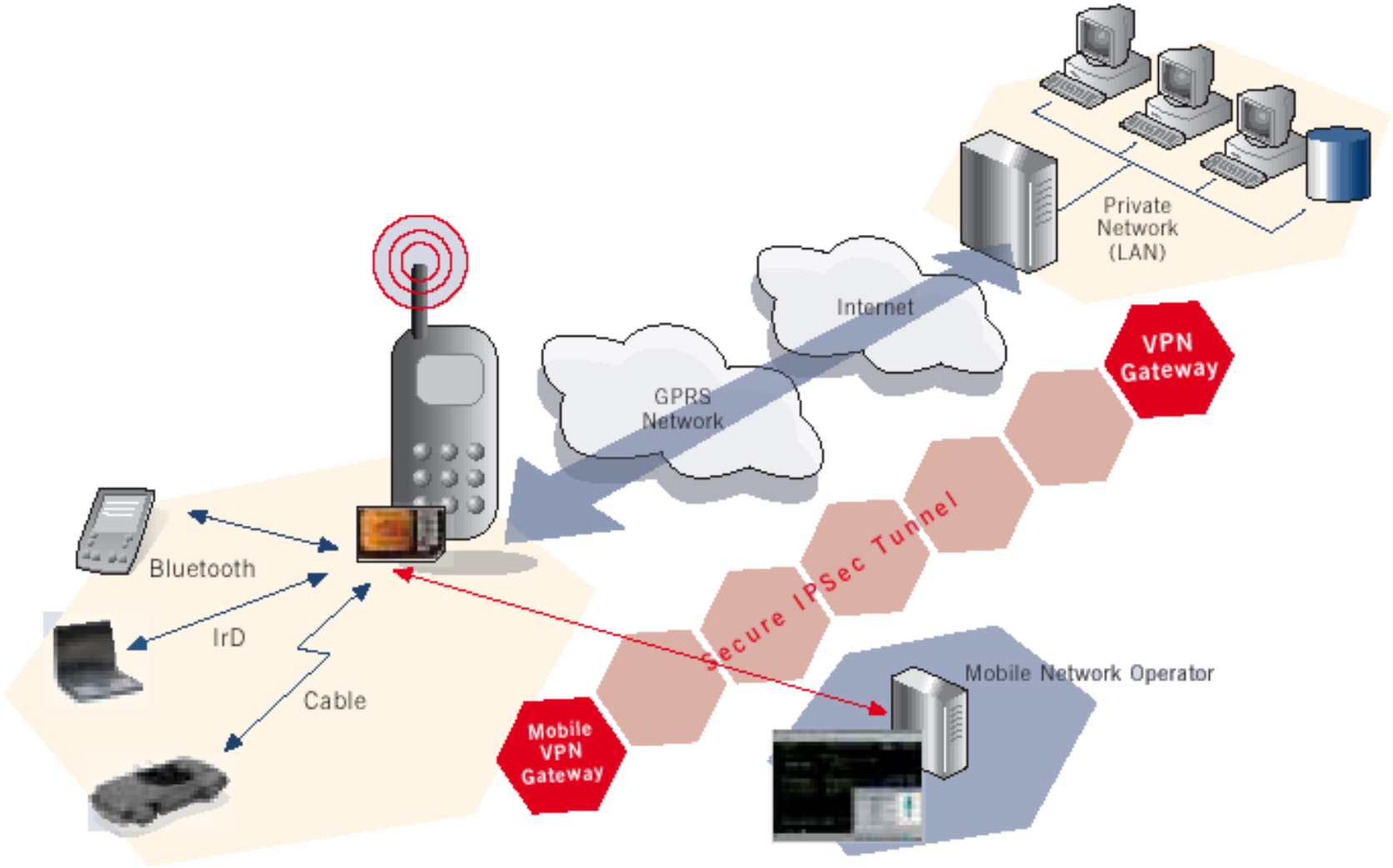
IP Security

IP in mobile networks

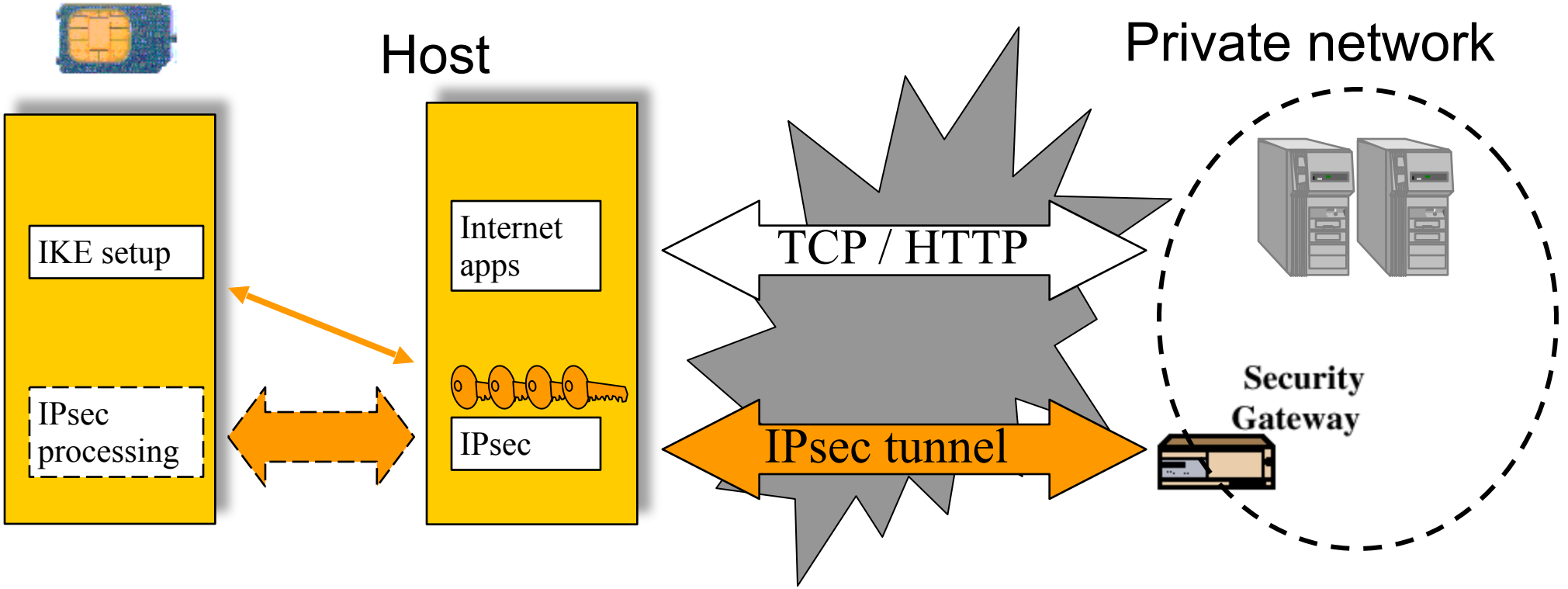
Internet Key Exchange

Diffie-Hellman with Java Card

IP in mobile networks



IPsec and smartcards



TCP = Transmission Control Protocol
HTTP = Hyper Text Transfer Protocol

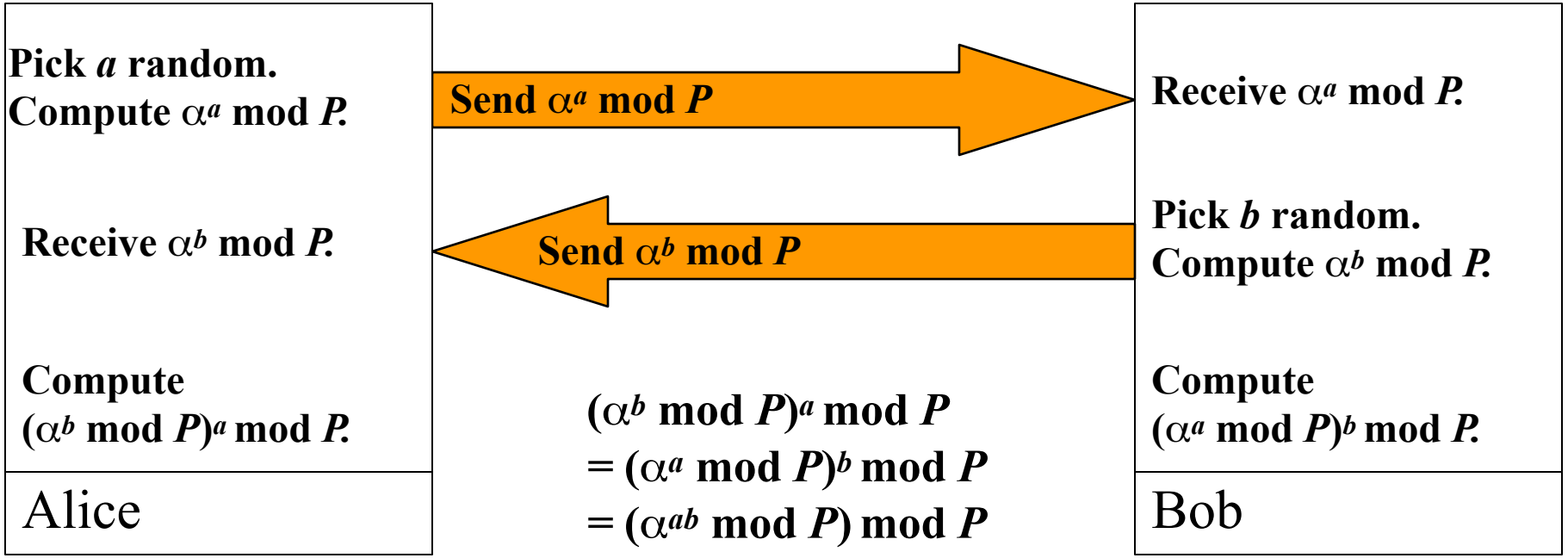
Internet Key Exchange

IKE defines two phases :

1. Creation of a secure channel between the IKE peers.
   a. **Negotiation of the IKE Security Association : encryption algorithm, hash function, authentication method (pre-shared keys, RSA), Diffie-Hellman group.**
   b. **Generation of shared secrets with a Diffie-Hellman exchange.**
   c. **Mutual authentication and establishment of the secure IKE channel.**
2. Creation of the IPsec security association.
   a. **Using the secure IKE channel, negotiation of the IPsec Security Association : SPI, IPsec transform.**
   b. **Generation of the required keys either by deriving the Phase 1 secret, or by performing another Diffie-Hellman exchange.**

Diffie-Hellman for mathematicians

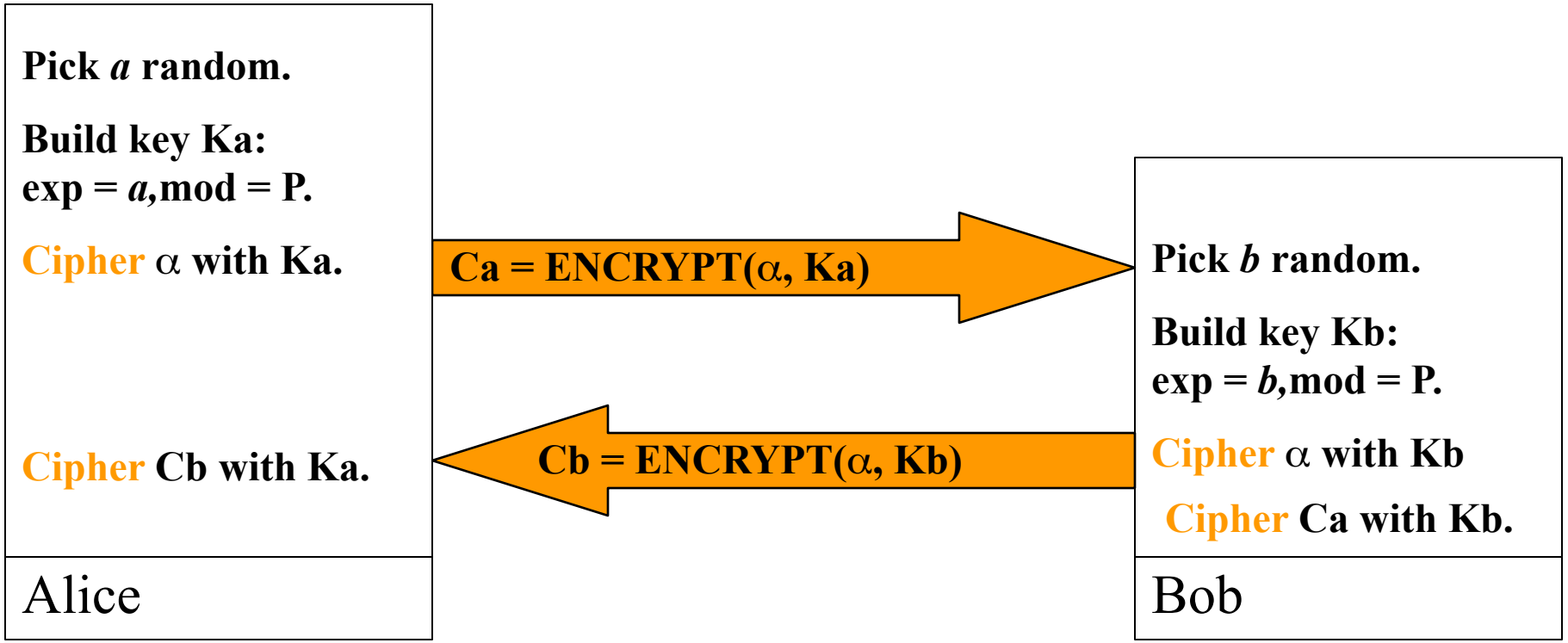
- Diffie-Hellman (1976) is a widespread protocol for shared secret establishment.
- *P* and α are two public numbers : α<*P*, *P* prime.



Shared secrets are derived from $(\alpha^{ab} \bmod P) \bmod P$

Diffie-Hellman for the layman

- DH may be implemented primitives from public key systems based on exponentiation (**RSA**, **DSA**) or elliptic curves (**ECDSA**).
- DH is vulnerable to the **man-in-the-middle attack**, so each peer should use a certificate to authenticate the other peer.



Diffie-Hellman with Java Card

- JC 2.1 supports RSA, so DH can be implemented.
- JC 2.2 introduces **javacard.security.KeyAgreement**, which is the base class for key agreement algorithms like DH.
  - **At this point, the KeyAgreement API only supports ECC keys.**
  - **Main operation : generation of a shared secret using the caller's Private Key and public data received from the peer.**
  - `public abstract short generateSecret(`
    `byte[] publicData, short publicOffset,short publicLength,`
    `byte[] secret, byte[] secretOffset)`

IPsec references

- **RFC 2401** – *Security Architecture For The Internet Protocol.*
- **RFC 2402** – *IP Authentication Header*.
- **RFC 2406** – *IP Encapsulating Security Payload.*
- **RFC 2407** – *The Internet IP Security Domain of Interpretation for ISAKMP.*
- **RFC 2408** – *Internet Security Association and Key Management Protocol.*
- **RFC 2409** – *The Internet Key Exchange.*
- **RFC 2631** – *Diffie-Hellman Key Agreement Method.*

Wi-Fi

802.11 weaknesses

Improvements to 802.11 security

Implementing 802.11 security on a (U)SIM smartcard

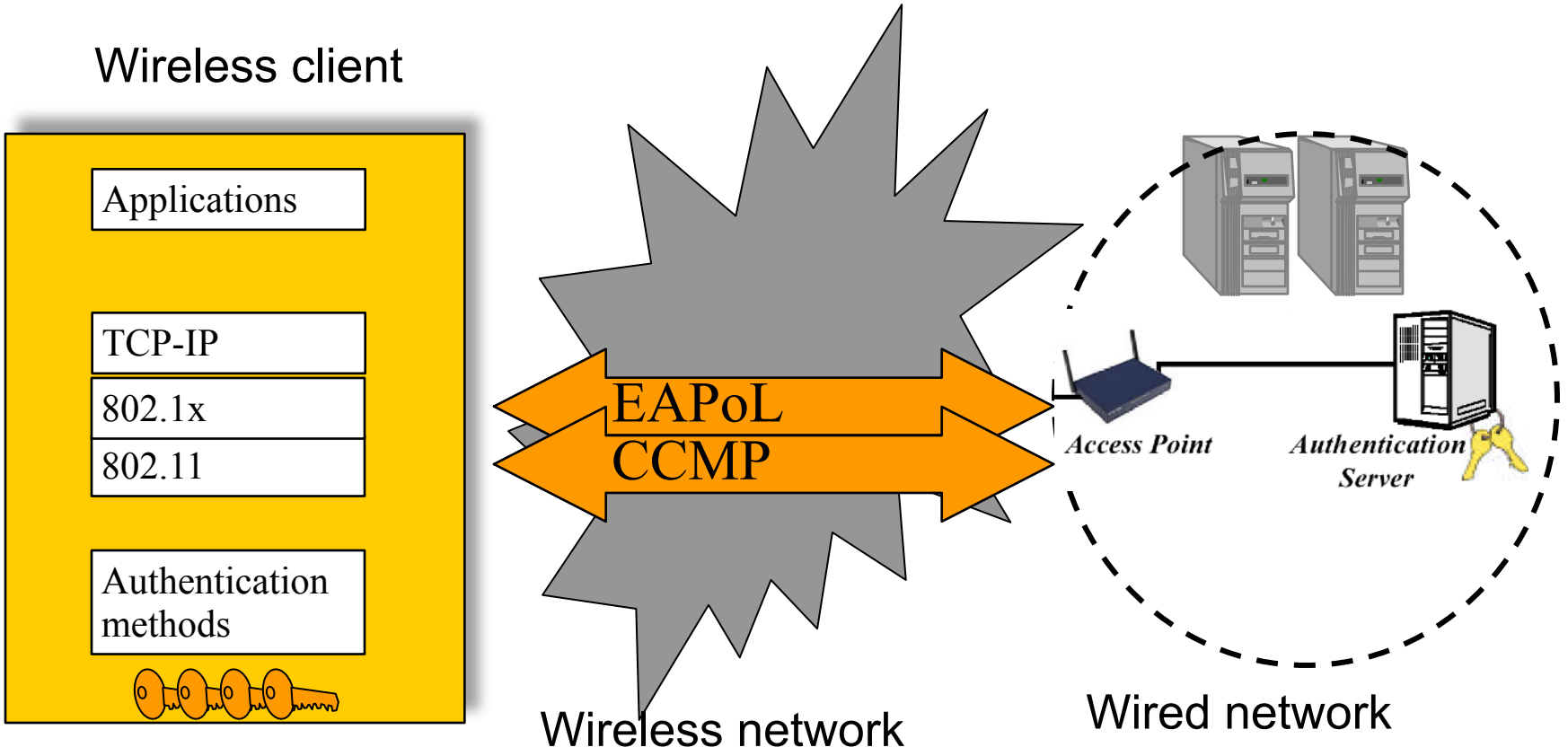
802.11 (in)security

- 802.11 security has been cracked.
- A new protocol is introduced to support **client authentication** and **distribution of session keys** : the **Extensible Authentication Protocol** (EAP).
- EAP is a generic transport protocol for a large number of authentication and key distribution methods.
- EAP can be integrated into PPP [RFC2284] or 802.2 [802.1X].

Improving 802.11 security

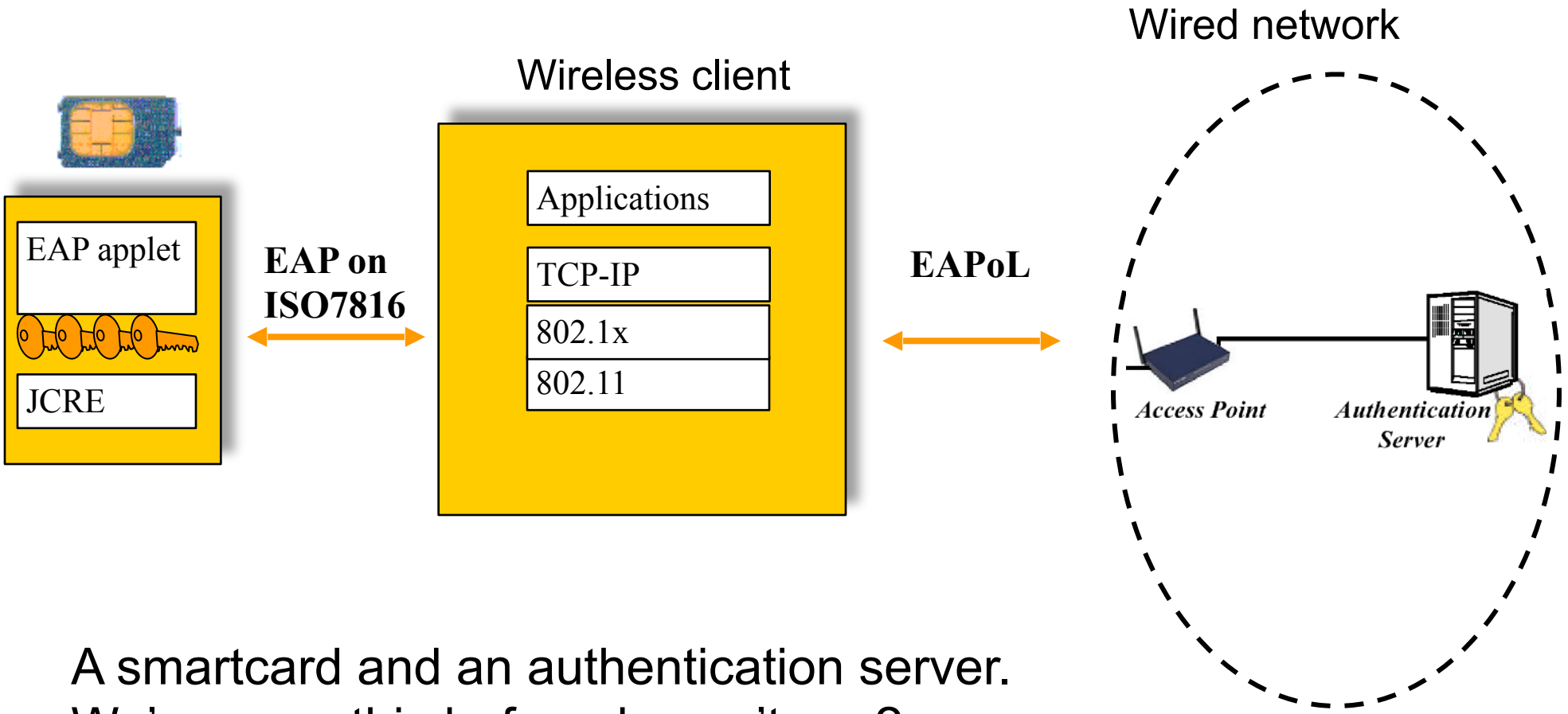
- **Wi-Fi Protected Access** (WPA) is a short-term solution (available now).
  - Authentication : 802.1X provides transport for EAP over LANs (EAPoL).
  - Encryption : the Temporal Key Integrity Protocol (TKIP) is based on RC-4 and uses 128-bit keys which are recycled every 10,000 packets.
- **802.11i** will be the long-term standard.
  - Authentication : 802.11i will maintain compatibility with WPA and building on 802.1x.
  - Encryption : 802.11i will introduce new authentication and confidentiality protocols based on the Advanced Encryption Standard (AES) :
    - *Counter CBC Mode Protocol (CCMP).*
    - *Wireless Robust Authenticated Protocol (WRAP).*

802.11i security



How safe are the keys ? How good is the crypto ?

Better 802.11 security



A smartcard and an authentication server.
We've seen this before, haven't we ?

EAP methods

- EAP is a generic protocol : it relies on a variety of methods for **authentication** and **key generation**.
- The following methods could be implemented on a smartcard to provide mutual authentication and key generation :

1. **EAP-TLS** [RFC2716] : Transaction Layer Security [RFC2246].
2. **EAP-SIM** (IETF draft) : 2G authentication algorithms.
3. **EAP-AKA** (IETF draft) : 3G authentication algorithms.

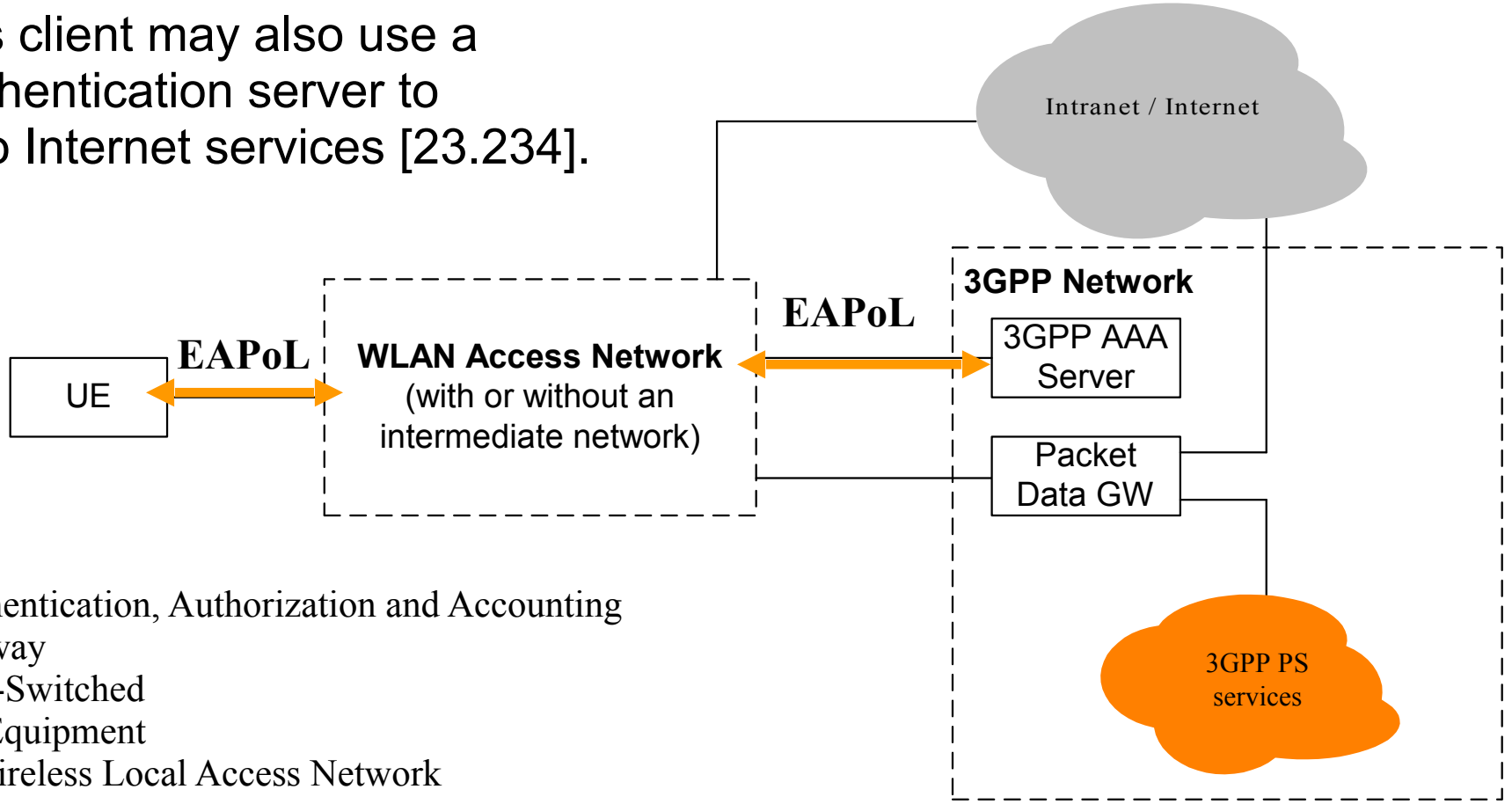
- EAP-SIM and EAP-AKA will be part of **3GPP {2,3}3.234** Release 6.

EAP commands

- Four packet types : **Request**, **Response**, **Success**, **Failure**.
- **Request** (authenticator ➔peer)
  a. **Identity** : query the identity of the peer.
  b. **Notification** : send a displayable message to the peer (password expiration, etc).
  c. **TLS** / **SIM** / **AKA** : set the authentication method and data.
- **Response** (peer ➔ authenticator)
  a. **Identity** : send a peer identity to the authenticator.
  b. **Notification**: acknowledge message.
  c. **TLS** / **SIM** / **AKA** : send authentication value.
  d. **NAK**: decline authentication method and propose another one.
- **Success** (authenticator ➔peer) : authentication has succeeded.
- **Failure** (authenticator ➔peer) : authentication has failed.

EAP & 3GPP

A wireless client may also use a 3GPP authentication server to connect to Internet services [23.234].



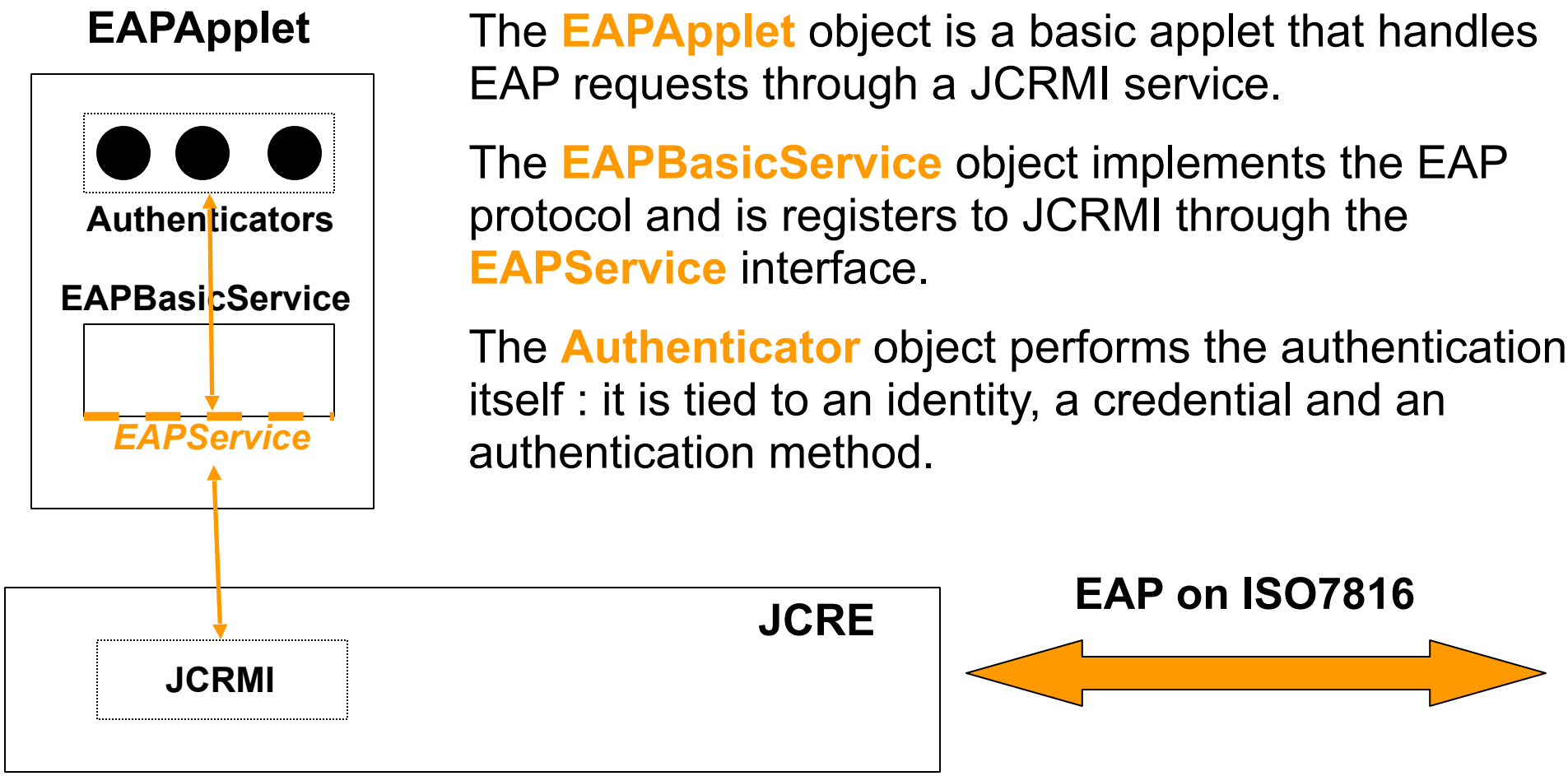
AAA = Authentication, Authorization and Accounting
GW = Gateway
PS = Packet-Switched
UE = User Equipment
WLAN = Wireless Local Access Network

EAP Java Card API

- This API is being designed by the **Java Card Forum** and the **WLAN Smart Card Consortium**.
- It allows Java Card developers to build applications that allow smart cards to be used as **EAP authentication tokens**.
- It has been built in the context of **WLAN end-user authentication**, which relies on the **EAP** protocol.
- In particular, this API has been built in the context of the APDU protocol defined in draft-urien-eap-smartcard-01.txt, titled ***EAP Support in Smart Cards***.
- It is an extension of the **Java Card 2.2** API and relies on **Java Card RMI**.

- All necessary cryptographic algorithms are supported by Java Card 2.2.

EAP Java Card API (1)

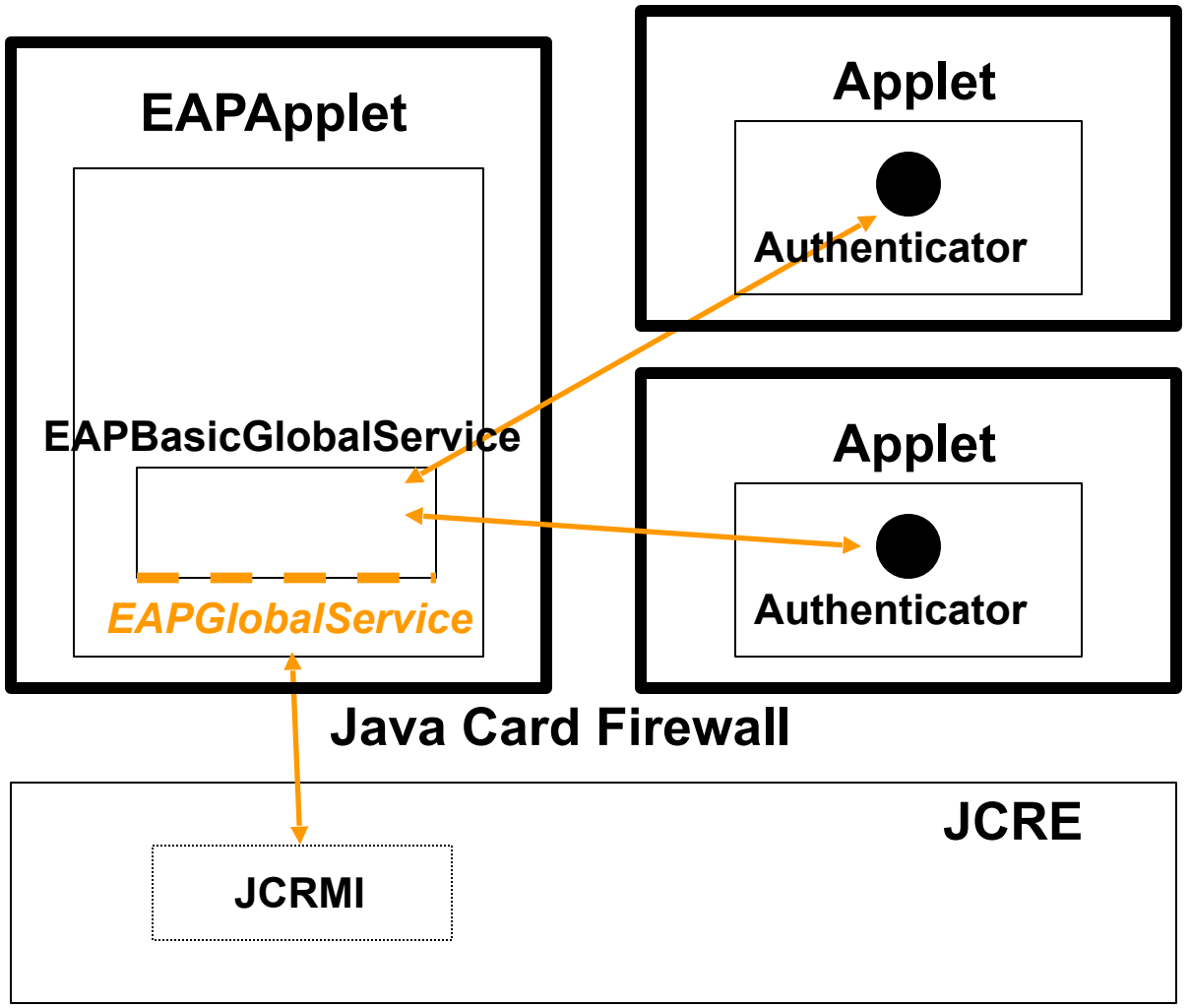


The **EAPApplet** object is a basic applet that handles EAP requests through a JCRMI service.

The **EAPBasicService** object implements the EAP protocol and is registers to JCRMI through the **EAPService** interface.

The **Authenticator** object performs the authentication itself : it is tied to an identity, a credential and an authentication method.

EAP Java Card API (2)



An **EAPApplet** may be used to provide EAP services to normal applets.

Inter-applet communication is performed through the **Shareable** interface.

EAP on ISO7816

Wi-Fi references (1)

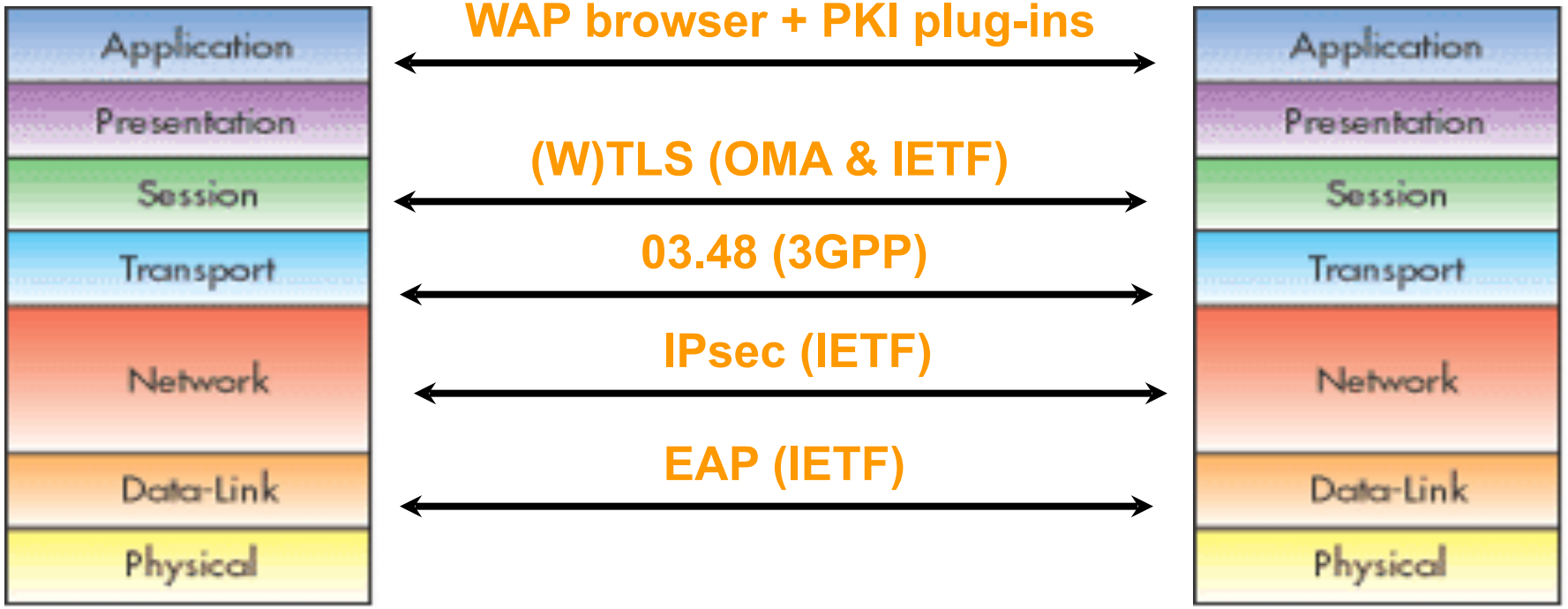
- **Wi-Fi Alliance** – http://www.weca.net
- **IEEE 802.x** – http://www.ieee802.org
- **IEEE 802.11** – ISO/IEC 8802-11: (1999) IEEE Standards for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Network - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.
- **IEEE 802.1x** – IEEE Standards for Local and Metropolitan Area Networks: Port-Based Network Access Control.
- **IEEE 802.11i** – Draft Supplement to Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications - Specification for Enhanced Security (Draft 3.0, November 2002).
- **3GPP –** http://www.3gpp.org
- **3GPP TS 23.234 v1.8.0 -** 3GPP system to WLAN Interworking; System Description (Release 6)

Wi-Fi references (2)

- **IETF –** http://www.ietf.org
- **RFC 1661** – *The Point-to-Point Protocol.*
- **RFC 1994** – *PPP Challenge Handshake Authentication Protocol (CHAP)*
- **RFC 2138** – *Remote Authentication Dial In User Service (RADIUS)*
- **RFC 2246** – *The TLS protocol, version 1.0*
- **RFC 2284** – *PPP Extensible Authentication Protocol.*
- **RFC 2716** – *PPP EAP TLS Authentication Protocol.*
- ***draft-ietf-eap-rfc2284bis-03.txt*** – *Extensible Authentication Protocol (EAP)*
- ***draft-aboba-pppext-key-problem-06.txt*** – *EAP Keying Framework*
- ***draft-urien-eap-smartcard-01.txt*** – *EAP support in smartcards*
- ***draft-haverinen-pppext-eap-sim-10.txt*** – *EAP SIM Authentication*
- ***draft-arkko-pppext-eap-aka-09.txt*** – *EAP AKA Authentication*
- ***draft-ietf-aaa-diameter-17.txt*** – *DIAMETER Base Protocol*

Conclusion

- Java Card is a safe foundation for many network security applications.
- And don't forget E-commerce / identification applications…
- As Java Card moves closer to the Java mainstream, new opportunities will arise (DRM, etc).



Q&A

- Thank you very much for inviting me and for attending this presentation.
- If you have any questions, I'll do my best to answer them.
- Feel free to get in touch !