TLS-Tandem

TLS-TANDEM

Goal

Learn how to design highly trusted WEB applications based on the popular TLS protocol, whose security is enforced by the Java Card technology.

ETHER TRUST

Agenda

Introduction, about TLS and WEB applications

TLS-Tandem concepts.

TLS-Tandem package.

DEMO: TLS-Tandem at work !

Agenda

Introduction, about TLS and WEB applications

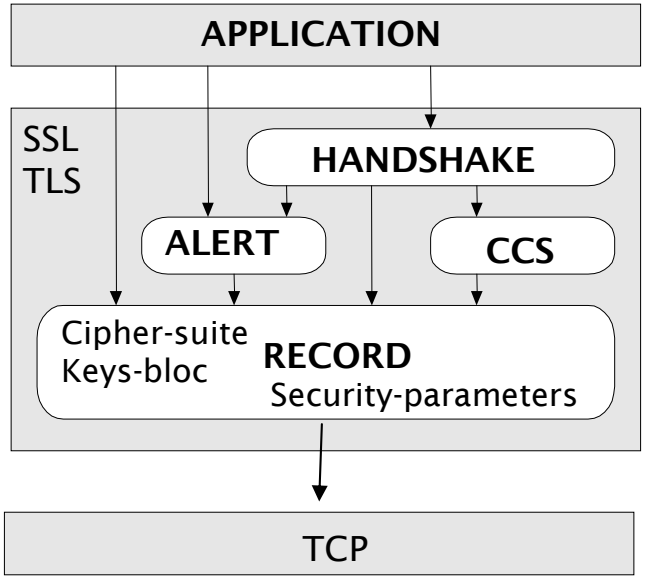
TLS-Tandem concepts.

TLS-Tandem package.

DEMO: TLS-Tandem at work !

About TLS and WEB applications

- SSL was designed by Netscape in 1994
  - SSLv2 (1994), SSLv3(1996)
- TLS is the IETF version of SSL
  - TLS (RFC 2246, 1999)
- TLS is widely used for WEB applications security

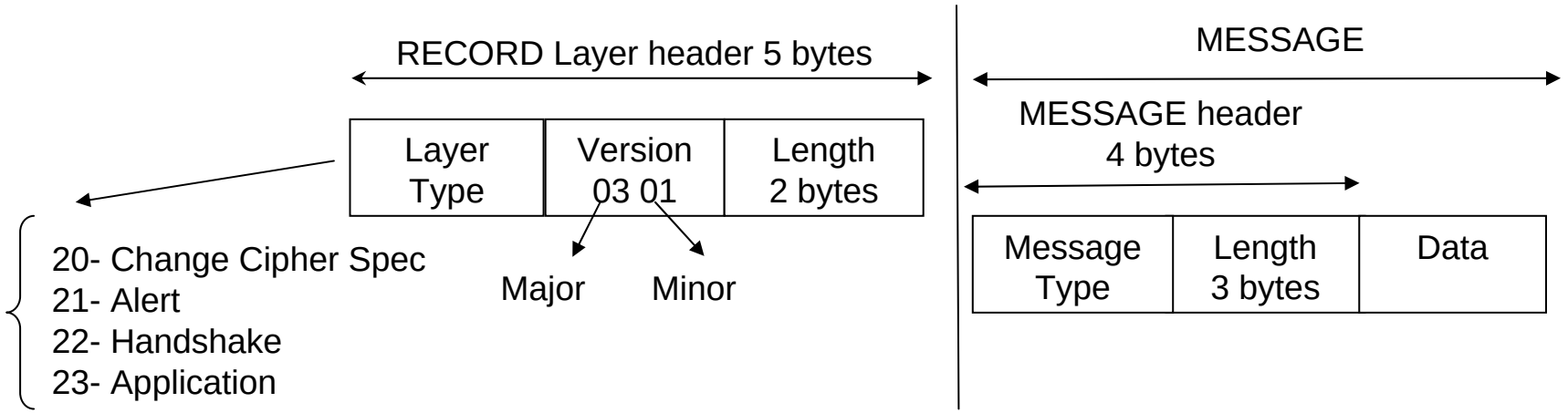


ETHER TRUST

How does TLS Work ?

- A TLS stack comprises four entities :
  - The Record layer, delivers all TLS packets, whose content is produced by application, handshake, alert or CCS entities.
    - In the secure mode it provide a protected channel whose data privacy and integrity are enforced by encryption (RC4,…) and MAC (HMAC-MD5, HMAC-SHA1,..)
  - The Handshake layer, performs authentication operations, negotiates cryptographic algorithms (*CipherSuite*), computes cryptographic keys (*KeysBloc*).
  - The Alert layer notifies errors
  - The Change Cipher Spec (CCS) layer, indicates that the Record layer is going to work in a secure mode, according to the cryptographic algorithms (*CipherSuite*) and associated keys (*KeysBloc),* previously negotiated.

A Record layer message



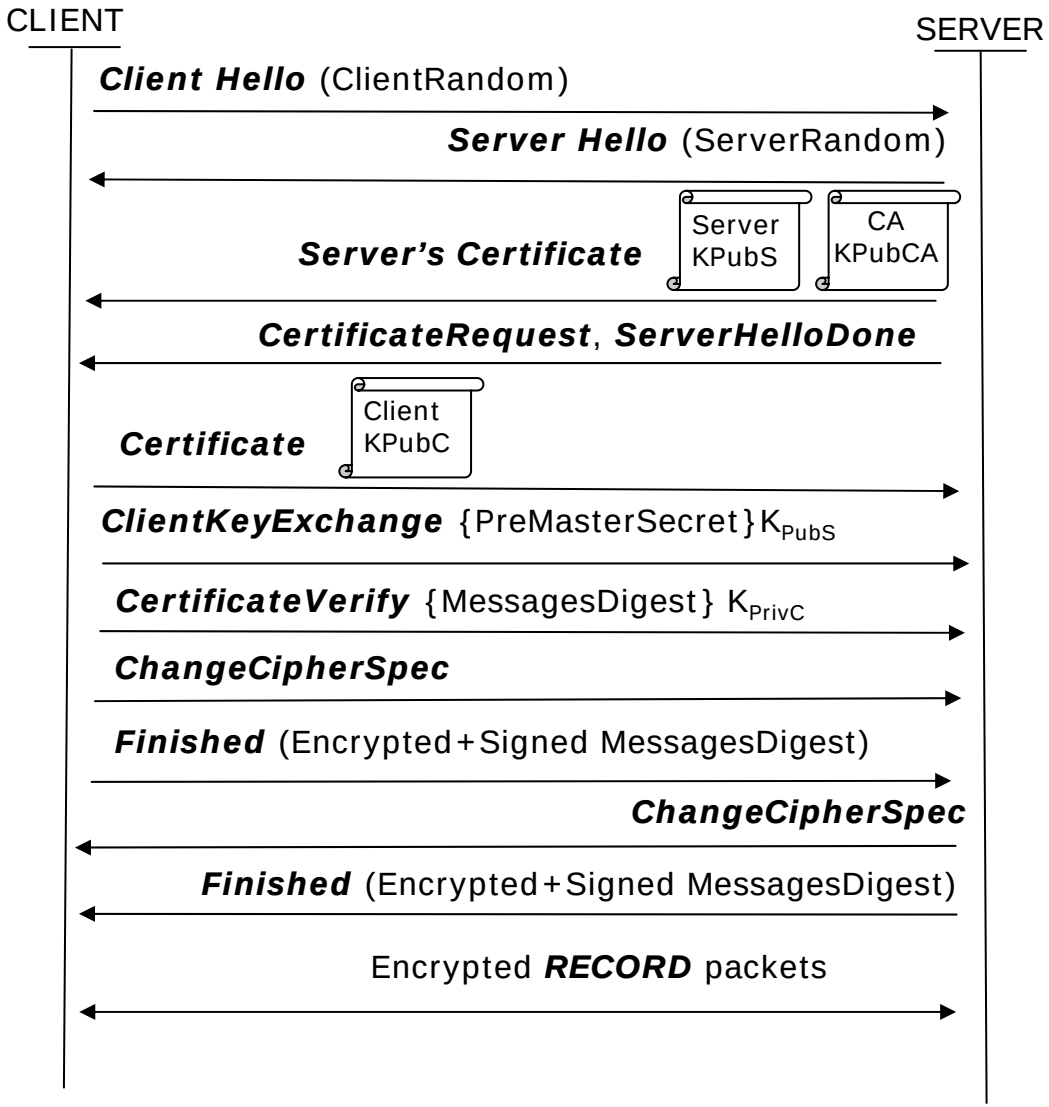
ETHER TRUST

Illustration of a full TLS session

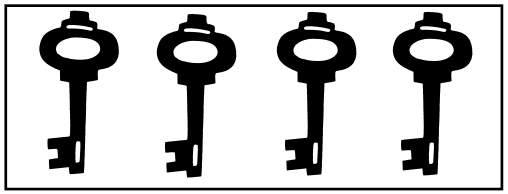
The client chooses a PreMasterSecret. It ciphers this value with the Server public key {PreMasterSecret} K_{PubS}

MasterSecret = PRF(pre_master_secret, “master secret", client_random | server_random)

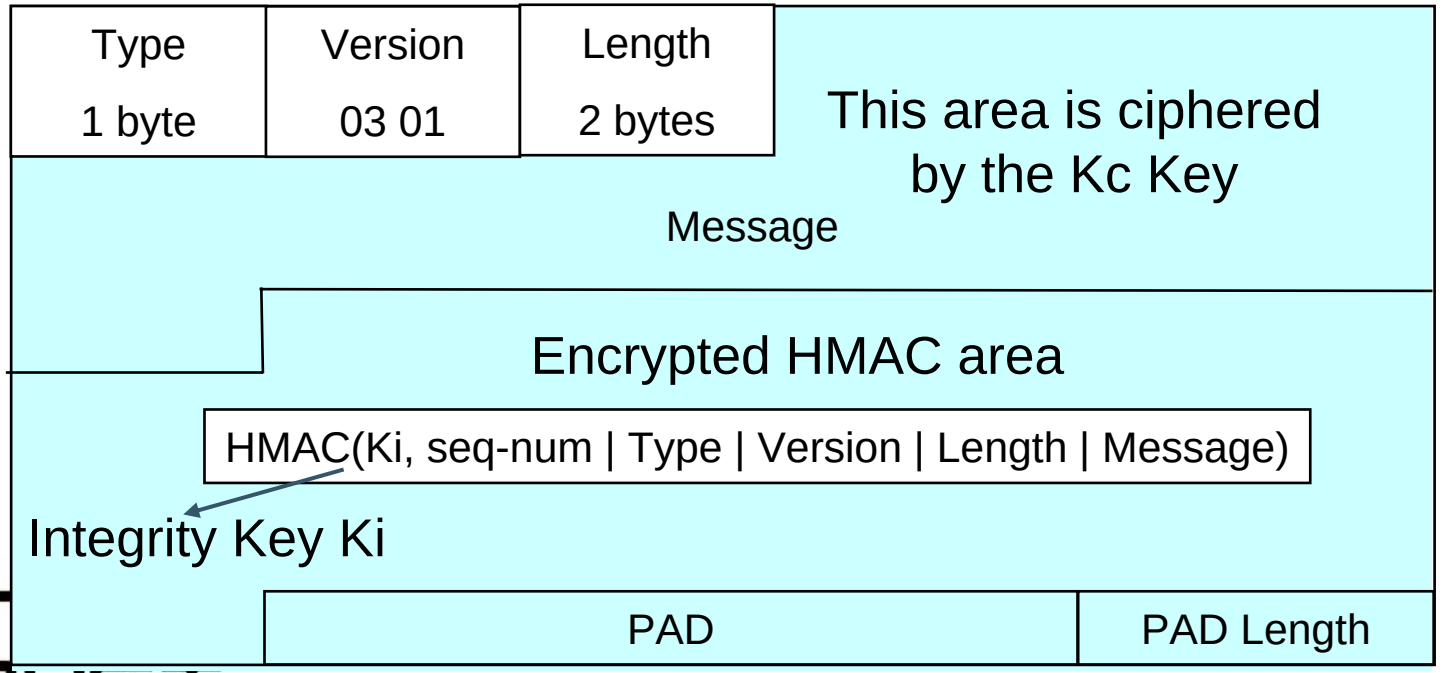
KeysBloc = PRF(master_secret, "key expansion", server_random | client_random)



The Record layer, in secure mode



KeysBloc = Two encryption keys +
two integrity keys



Agenda

Introduction, about TLS and WEB applications

TLS-Tandem concepts.

TLS-Tandem package.

DEMO: TLS-Tandem at work !

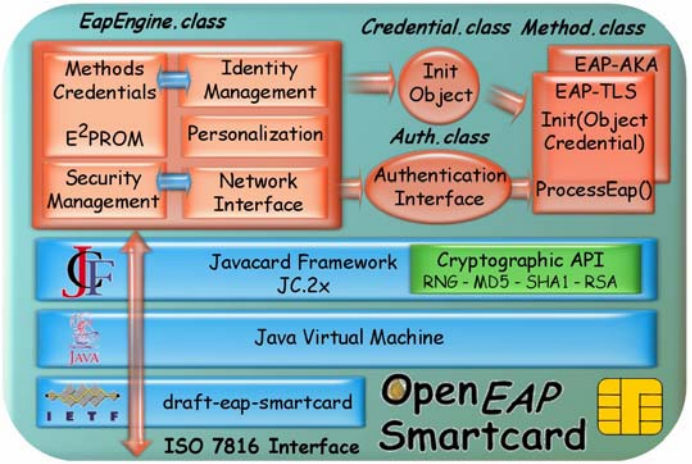
TLS-Tandem main idea.

- The opening of TLS session is fully managed by a Java Card™.
  - Java Cards are highly secure. They may be issued by WEB applications providers.
    - X509 Certificate are stored in Java Cards
    - RSA private keys are only used in this trusted environment.
  - But today smart card components are not able to handle high (encrypted) data rates.
- Once the session has been open, its control is transferred to a JAVA terminal
  - CipherSuite and KeysBloc parameters are collected from the Java Card
  - The *Record Layer* is afterwards fully managed by the JAVA terminal

ETHER TRUST

TLS in Java Cards

- EAP-TLS Java Cards were introduced during JavaOne 2007
  - TS-0285, "JavaCard for Emerging WLAN Environments“
- EAP-TLS is a transparent transport TLS
  - Defined by RFC 2716
- EAP-TLS Java Cards are built with the OpenEapSmartcard framework.
  - EAP smart cards are defined by the IETF draft, *draft-urien-eap-smartcard-13.txt*



Example of smart card computing performances with a 32 bits microcontroller

- RSA encryption / decryption operation with 1024 bits public keys: 40 ms
- RSA encryption / decryption operation with 1024 bits private keys: 740 ms
- MD5 calculation: 2,25 ms/bloc
- SHA1 calculation: 1,45 ms/bloc
- 3xDES (2 keys) operation: 11 ms/bloc
- AES (128 bits) encryption/decryption: 6,25 ms/bloc
- Opening of a TLS *full session:* 5 seconds

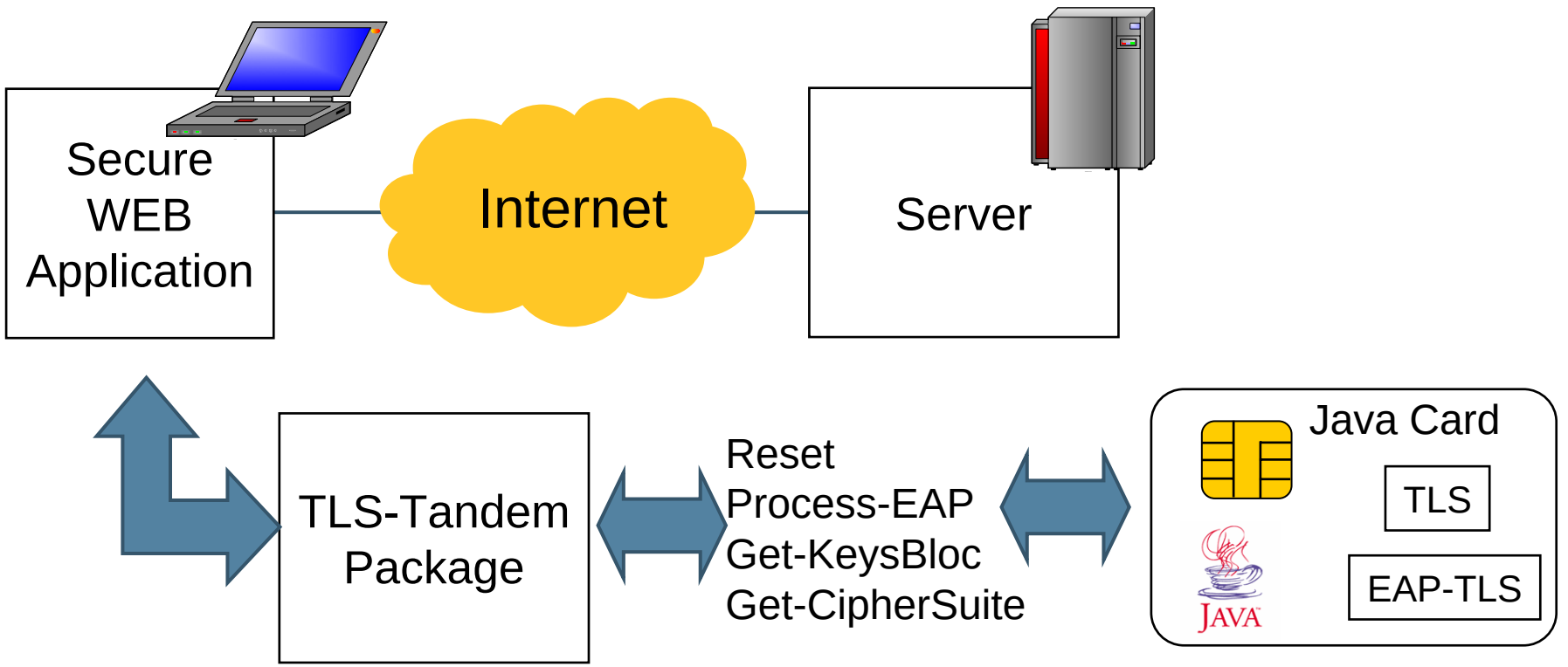
ETHER TRUST

Main commands of TLS-Tandem Java Cards

- *Reset (imported from the IETF draft)*
  - Reset the EAP-TLS state machine
- *Process-EAP (imported from the IETF draft)*
  - Processing of an EAP-TLS message
- *Get-KeysBloc (new command)*
  - Reading of the TLS *KeysBloc* value
- *Get-CipherSuite (new command)*
  - Reading of the TLS *CipherSuite* value

ETHER TRUST

Functional architecture



ETHER TRUST

Agenda

Introduction, about TLS and WEB applications

TLS-Tandem concepts.

TLS-Tandem package.

DEMO: TLS-Tandem at work !

The tls-tandem package

- Two main classes, tls-tandem, recordlayer
- The *tls-tandem* class
  - Manages TCP sockets.
  - Translates EAP-TLS messages in pure TLS messages
  - Handles smart card reader operations and dialogs with Java Cards.
- The *recordlayer* class
  - Provides all facilities to send and receive TLS packets when the record layer is in secure mode.

tls-tandem class overview

Field Summary

static int	**CLIENT** CLIENT mode (0)
static int	**SERVER** SERVER mode (1)

Constructor Summary

tls_tandem(int Mode, java.lang.String ReaderName)
TLS-Tandem creation
Mode: tls-tandem.CLIENT or tls-tandem.SERVER
ReaderName: the smart card reader name, null is default

Method Summary

void	**close_reader**(java.lang.String ReaderName) TLS-Tandem end ReaderName: the smart card reader name, null is default
void	**CloseSession**(recordlayer RecordLayer) Close an TLS-Tandem session RecordLayer: The recordlayer object
recordlayer	**OpenSession**(java.lang.String ServerName, short Port) Open a TLS-Tandem session ServerName: the name or the IP address of a TLS server.

ETHER TRUST

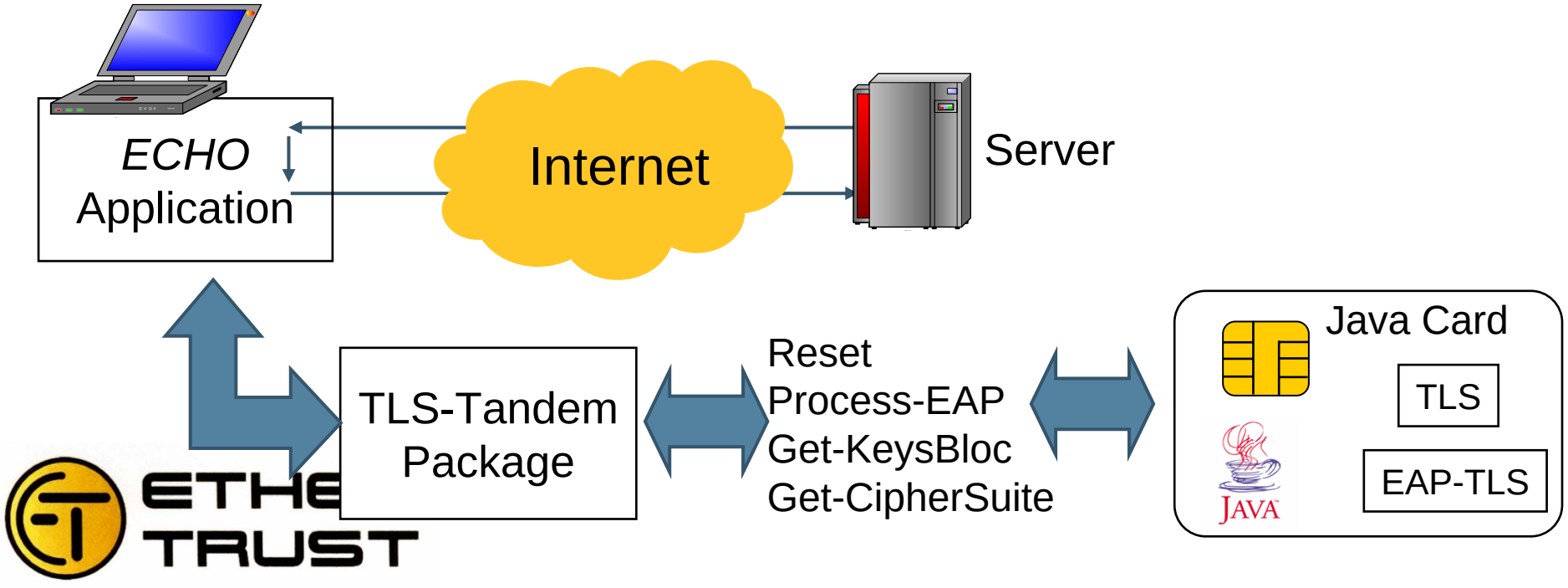
recordlayer class overvi

Field Summary	
short	**CipherSuite** The CipherSuite value RC4_MD5 = (short)0x0004 RC4_SHA1 = (short)0x0005
byte[]	**KeyBloc** The set of keys used by the RecordLayer

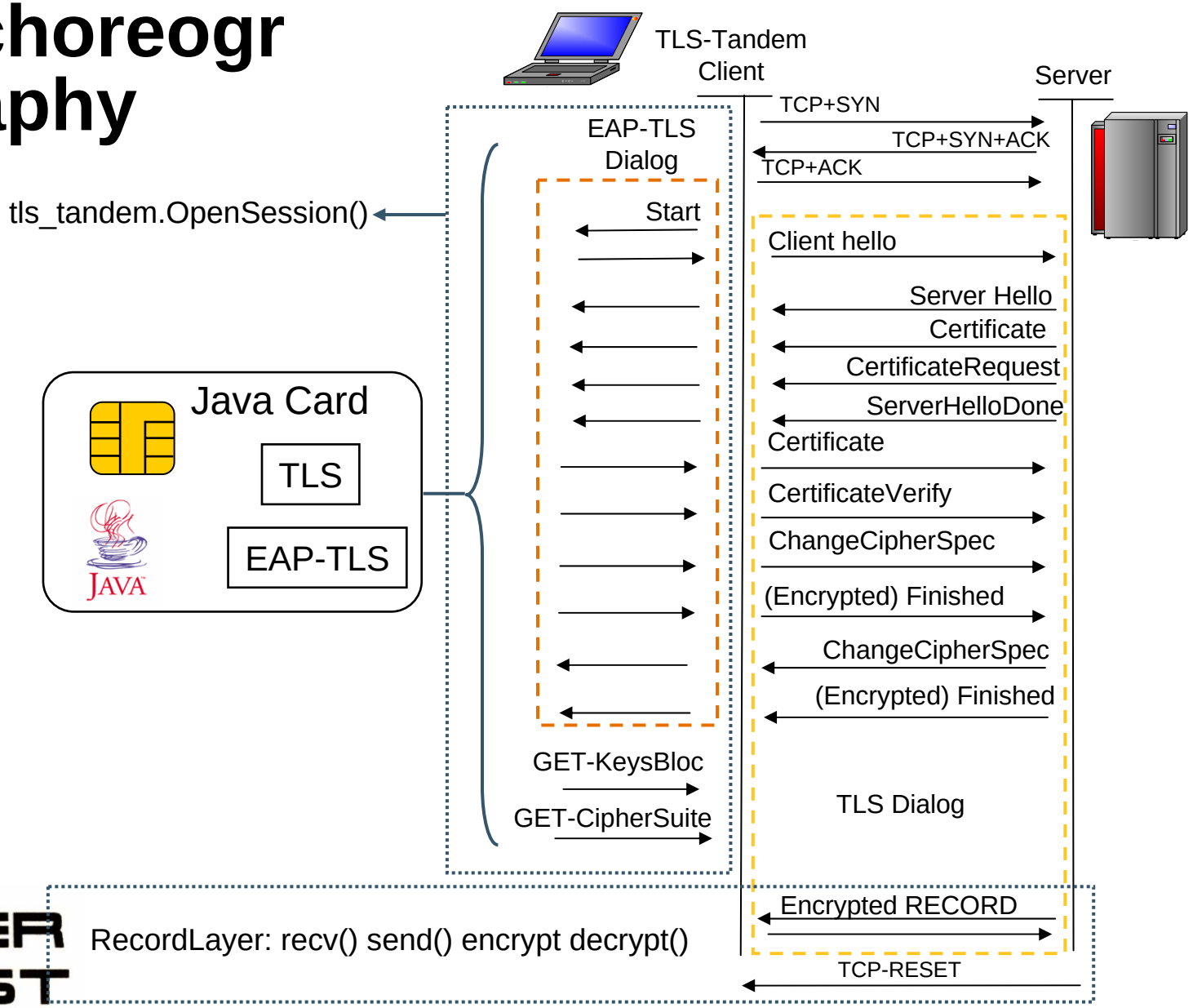
Method Summary	
byte[]	**decrypt**(byte[] msg) Decryption and integrity checking of a RecordLayer packet.
static java.lang.String	**dump**(byte[] Value, int Offset, int Length) Translation of a binary value in an hexadecimal format value: a binary array offset: offset of the value to dump len: length of the value to dump
byte[]	**encrypt**(byte[] msg) Building of a RecordLayer encrypted packet.
byte[]	**recv**() Return an incoming RecordLayer (encrypted) packet
(package private) int	**send**(byte[] buf) Send an encrypted RecordLayer packet

The “echo” application

- A TLS session is opened thanks to a Java Card and the tls-tandem ***OpenSession*** method that returns a ***recordlayer*** object.
- Once the session has been opened the *echo* application decrypts and echoes all incoming TLS packets, thanks to ***recordlayer*** facilities.



TLS-Tandem choreography

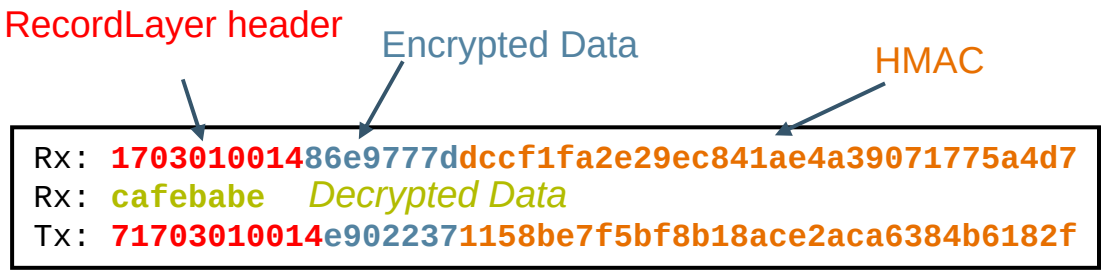


ETHER TRUST

The “echo” code source

```
import java.io.* ; import tls_tandem.* ;
public class console  {
public static void  main(String args[])
{ tls_tandem thistest = new tls_tandem(tls_tandem.CLIENT,null);
  recordlayer RecordLayer = thistest.OpenSession("127.0.0.1",(short)443);

   while(true) {
   byte[] buf1 = RecordLayer.recv(); if (buf1==null) break;

   System.out.println("Rx: " + recordlayer.dump(buf1,0,buf1.length));
   byte[] buf2 = RecordLayer.decrypt(buf1); if (buf2==null) break;

   System.out.println("Rx: " + recordlayer.dump(buf2,0,buf2.length));

   buf2=RecordLayer.encrypt(buf2); if (buf2==null) break;

   System.out.println("Tx: " + recordlayer.dump(buf2,0,buf2.length));

   int err = RecordLayer.send(buf2); if (err == 0) break;}

   thistest.CloseSession(RecordLayer);
   thistest.close_reader(null);
   System.out.println("TLS-Tandem Session End"); }}
```

Some dumps from “echo”



Operations managed by the echo application, thanks to the recordlayer facilities

Hidden operations managed by the tls-tandem class

```
TLS channel opened
Get KeysBloc
>> A0 82 CA 00 40
<< B3 C7 9E A4 60 32 3A 0B 2F 71 2E 75 D7 7D 1D C2
   F9 38 96 90 0F 04 BD BE 7A 4D E1 62 59 29 69 A1
   1A D4 A3 58 4C 40 FC 51 C8 8B 75 7A A2 F0 AE ED
   39 6B 02 8E 31 0A 3C 32 52 AC 4E 1A 6A 32 C0 1D
   90 00
Get CipherSuite
>> A0 82 CC 00 00
<< 6C 03
>> A0 82 CC 00 03
<< 02 00 04 90 00
TLS-Tandem is Ready
```

ETHER TRUST

Agenda

Introduction, about TLS and WEB applications

TLS-Tandem concepts.

TLS-Tandem package.

DEMO: TLS-Tandem at work !

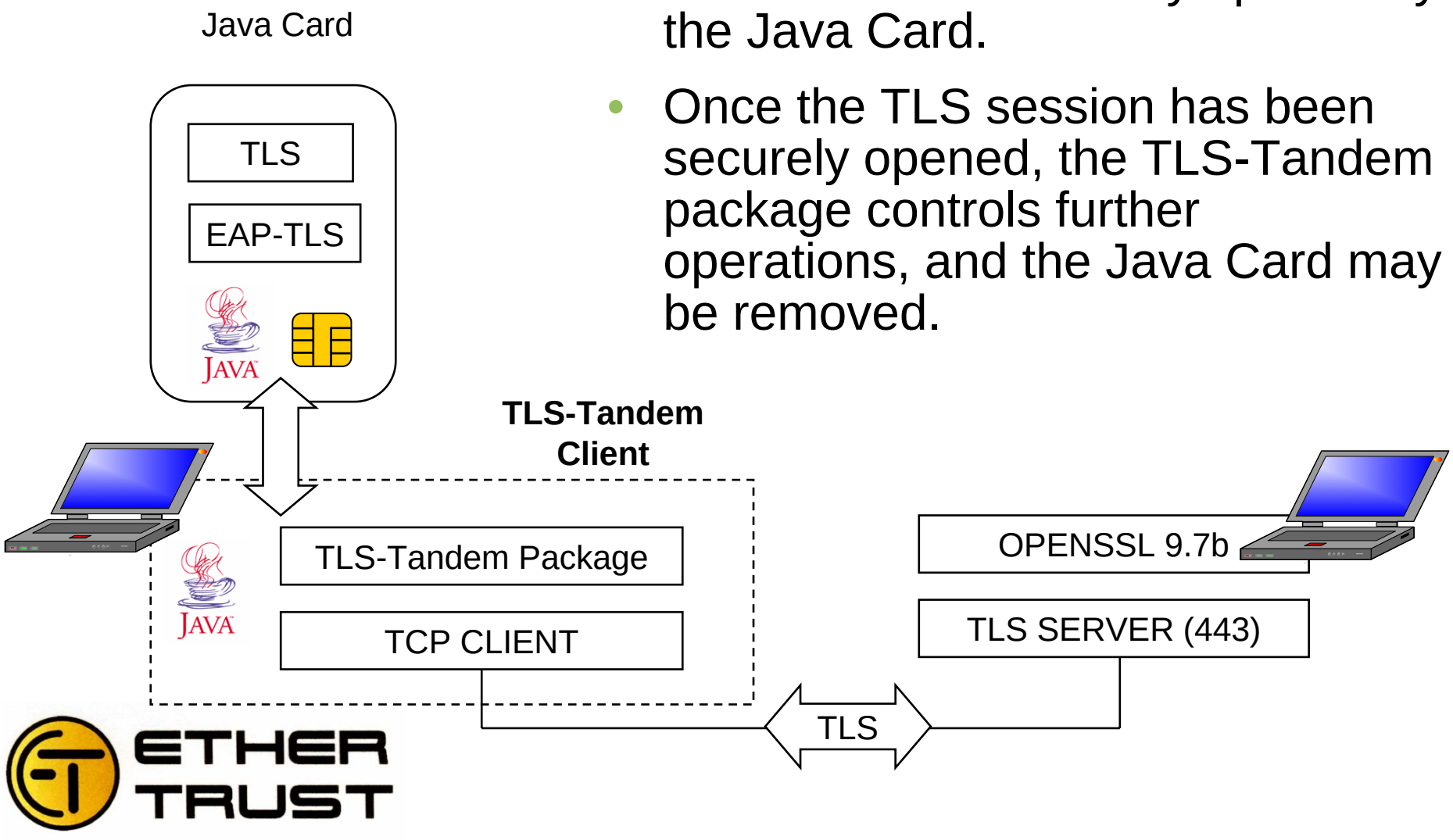
DEMO

TLS-Tandem at work !

DEMO 1 – The echo application

A TLS-Tandem Client

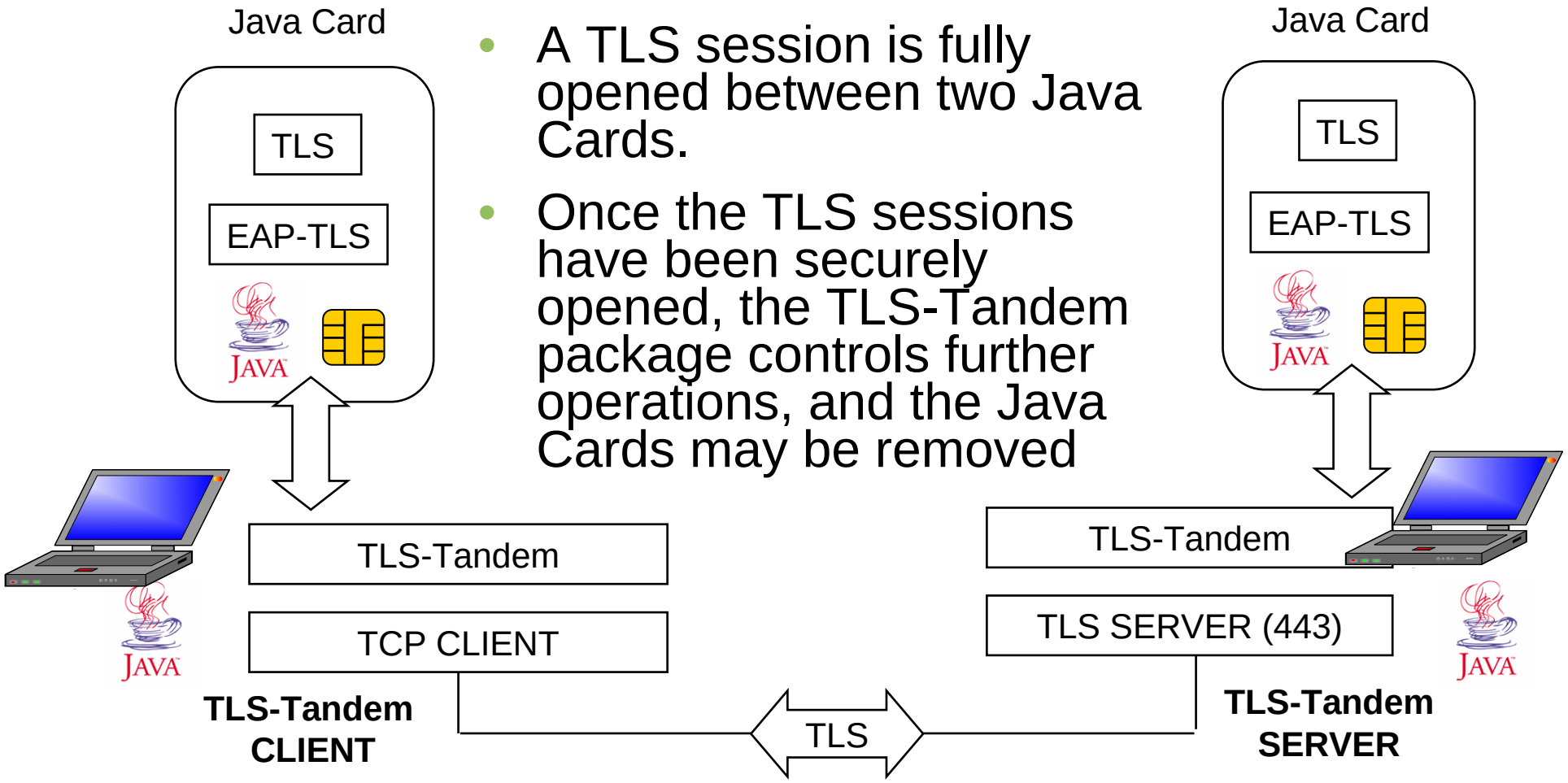
- The TLS session is fully opened by the Java Card.
- Once the TLS session has been securely opened, the TLS-Tandem package controls further operations, and the Java Card may be removed.



A Peer To Peer application.

DEMO 2

- A TLS session is fully opened between two Java Cards.
- Once the TLS sessions have been securely opened, the TLS-Tandem package controls further operations, and the Java Cards may be removed



ETHER TRUST

Summary

- We have introduced the TLS-Tandem concepts.
- We have presented TLS-Tandem Java Cards.
- We have introduced JAVA package used by TLS-Tandem.
- We have demonstrated a working TLS-Tandem platform.

ETHER TRUST

For More Information

- http://www.enst.fr/~urien
- “TLS-Tandem”, technical paper to appear
- http://www.ethertrust.com

Q&A

- Pascal Urien