A New Mobile NFC Key Delivery Service

Pascal Urien
Department of Networking and Computer Science
Telecom ParisTech, 23 avenue d'Italie, - Paris, 75013
France
Pascal.Urien@telecom-paristech.fr

Christophe Kiennert
EtherTrust
63bis rue Gay Lussac, Paris, 75005
France
Christophe.Kiennert@ethertrust.com

Abstract— **This paper introduces a new mobile service, delivering keys for hotel rooms equipped with RFID locks. It works with Android smartphones offering NFC facilities. Keys are made with dual interface contactless smartcards equipped with SSL/TLS stacks and compatible with Mifare legacy locks. Keys cards securely download keys value from dedicated WEB server, thanks to Internet and NFC connectivity offer by the Android system. We plan to deploy an experimental platform with industrial partners within the next months.**

Keywords- Mobile service; security; NFC; smartcards; SSL/TLS

I. INTRODUCTION

Mobile service is a very attractive topic for the deployment of the emerging always on society. It is expected [1] that in 2015, about one billion of smartphones, with full Internet connectivity, will be sold every year. Android is a popular open operating system for mobiles based on UNIX, whose version 1.0 was commercialized by the end of 1998. Two years later, fall 2010, the 2.3 version (also refereed as Gingerbread) was released with the support of Near Field Communication (NFC) standard [2]. This technology appears in the first decade of the 21st century. It is a radio link, working at the 13,56 MHz frequency and integrated in low power tamper resistant microelectronic chips, usually named contactless smartcards. These devices, battery free and feed by the electromagnetic field, are widely used in Europe and Asia for ticketing, access control and banking purposes. According to the NFC terminology, Gingerbread supports the peer to peer mode (data exchange between two NFC enabled devices), and the reader mode (feeding and communication with an NFC device working in the card mode). Despite the fact that the hardware could also provide the card mode, this feature is not currently supported by Android.

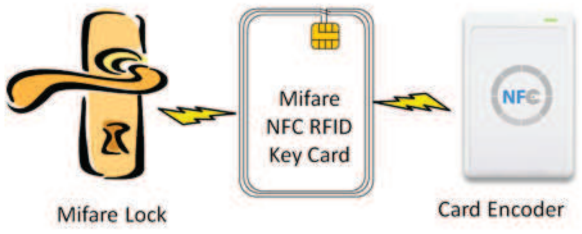

Figure 1. RFID lock, legacy system

This paper presents an experimental mobile service targeting key delivering for electronic locks. The legacy service (see for example [3]) is illustrated by figure 1. Electronic locks are equipped with RFID readers, and work with RFID cards (frequently including Mifare [6] components) in spite of magnetic strip cards. A device named the Card Encoder, belonging to a dedicated information system, write keys values in RFID cards.

The security of electronic lock environment is a quite new subject for the scientific community. The paper [4] proposes an architecture working for locks equipped with microcontrollers, and reading keys built from telephone smartcards. The reference [5] describes a system that relays information, collected from passive ISO 15693 RFIDs by readers embedded in locks, toward central access control software.

Our new experimental platform (see figure 2) works with dual interfaces RFIDs (whose structure is detailed by section IV), establishing secure SSL/TLS sessions with a key server. Internet connectivity and human interface is provided and managed by an Android phone. The user is identified by the X509 certificate stored in his key card, and thanks to its smartphone securely collects a key from the dedicated WEB server.

This paper is an extended version of the work previously published in [23]. It is constructed according to the following outline. Section 2 introduces the NFC technology. Section 3 briefly recalls the structure of the Android operating system, and its NFC features. Section 4 introduces dual interfaces RFIDs. Section 5 presents basic concepts of the EAP-TLS application for contactless smartcard; it analyzes performances issues for the prototype platform, and details JAVA APIs. Section 6 describes the new secure key delivering services. Section 7 provides some discussion about this work. Finally Section 8 concludes this paper.

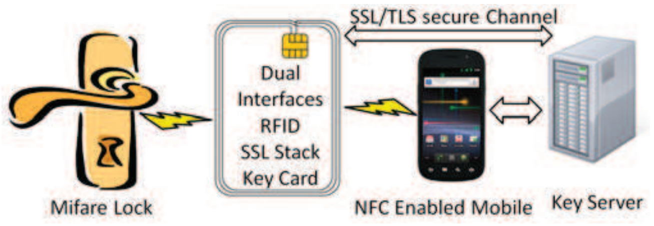

Figure 2. RFID Lock, new mobile service

II. ABOUT NFC RFID

The Near Field Communication (NFC) technology is a radio interface working at 13,56 MHz. It supports several data encoding schemes, delivering binary throughput ranging from

106, 212, 424 or 848 Kbits/s. The two main classes of such modems are referred as typeA and typeB and are detailed by the ISO 14443 and NFC standards. This technology is embedded in small electronic chips with low power consumption (less than 10mW), which are feed by electromagnetic induction. A device equipped with an antenna and usually named the reader, generates a magnetic field of about 5 A/m, which according to the Lens laws induces a tension (E) of about 2,2V on a rectangular loop with an area of 5x8 cm2.

$$E = 2 \pi f \mu o H S = 2,14 \text{ V}$$

With $f=13,56\ 10^{6}$, $\mu o=4 \pi\ 10^{-7}$, $H=5$, $S=40\ 10^{-4}$ (SI),

The working distance of this system is within 10 cm.

The RFID components are split in two categories,

- small chips (less than 1mm^2) designed with cabled logic;

- secure microcontrollers chips (about 25 mm^2) equipped with CPU, RAM, Non Volatile Memory, and cryptographic accelerators units.

A good illustration of the first RFID category is the Mifare 1K [6] (1K meaning one Kbits of memory) widely deployed for ticketing applications or RFIDs keys [3].

Electronics passports (normalized by the ICAO standards [7]) includes RFIDs belonging to the second category either typeA or typeB. Information, especially biometric records, is protected by various cryptographic procedures based on 3xDES, RSA, or ECCDH algorithms.

In this paper we present a highly secure key delivering service dealing with smartphones and dual interfaces RFIDs. These electronic chips equipped with an antenna, support both the Mifare 1K and ISO 14443 (typeA) protocols and embeds a secure microcontroller. Today some hotels are already equipped with NFC locks, including a battery and a reader, which read customers' RFID cards.

The basic idea of our new service is to get a key from a WEB server thanks to an SSL/TLS stack running in the RFID secure microcontroller and monitored by Android software. This data is afterwards transferred in the Mifare emulated card.

III. ABOUT ANDROID

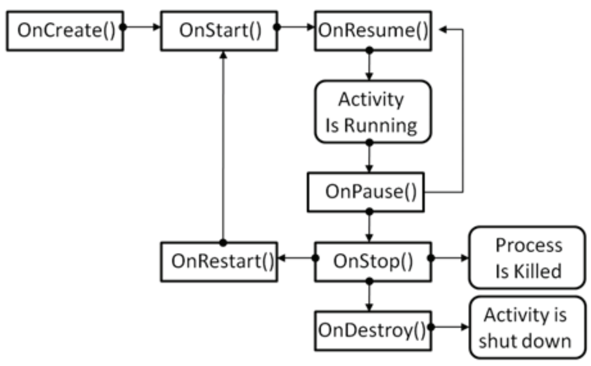

Figure 3. The Android Activity state machine

Android [8] [9] [10] is an operating system originally created by the company Android Inc. and supported by the Open Handset Alliance, driven by the Google company. It used a Linux kernel and provides a runtime environment based on the java programming language. Applications are compiled from JAVA modules, then transformed by the "dx" tool and executed by a particular Virtual Machine called the Dalvik Virtual Machine (DVM). This virtual machine processes code bytes stored in Dalvik (.dex) files, whose format is optimized for minimal memory footprint.

```
private void resolveIntent (Intent intent)
throws IllegalArgumentException, IllegalAccessException  {

String action = intent.getAction();
 if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(action))
 { Tag tag = intent.getParcelableExtra
 IsoDep TagI = IsoDep.get(tag) ; if (TagI== null) return ;

  try { TagI.connect();}
  catch (IOException e) {return;}

  byte request[]= {(byte)0x00,(byte)0xA4,(byte)0x04,(byte)0x00,
                   (byte)0x07,(byte)0xA0,(byte)0x00,(byte)0x00,
                   (byte)0x00,(byte)0x30, (byte)0x00,(byte)0x01};

// Send ISO7816 Request, Receive Response
try { byte[] response= TagI.transceive(request);
catch (IOException e) {return;}

  try {TagI.close();}
  catch (IOException e) {return;}

  if ( (response[0]==(byte)0x90) && (response[1]== (byte)0x00))
 { tv.setText("OK");setContentView(tv);   }
 return;
}
else  return;  } }
```

Figure 4. NFC device detection and data exhange via an Intent mechanism

An Activity is an application component that manages a screen with which users can interact in order to do something. It is associated with a lifecycle state machine (see figure 3), which starts with the onCreate() event and ends with the onDestroy() event. Threads created by an activity are typically destroyed when shutdown occurs. The "back" key kills the foreground activity while the "home" key switches to another one. The onRestart() message notifies that the activity is going to control again the foreground.

An activity may register to the Android system in order to be launched by asynchronous messages named Intent. The list of Intent processed by an application is fixed by an Intent Filter facility.

An Android application structure is described by the file AndroidManifest.xml. This object declares all components (such as activities) used by the application. It also includes required permissions such as

- Internet Access or NFC use,

- Filter Intent needed for asynchronous triggering,

- And minimum Android version required by the application.

The Android version 2.3, also named Gingerbread, supports NFC software APIs, building an abstract framework over a NFC adapter chip (such as the PN65N, manufactured by NXP and soldered on the Nexus S electronic board).

The NfcManager class enumerates the NFC adapters available on the Android device board. Usually there is only one chip, so the static method getDefaultAdapter() returns an instance the class NfcAdapter, which represents the hardware adapter.

An application that is registered to a RFID discovery event (such as TAG_DISCOVERED) gets an Intent object, from which is extracted a Tag object. This later is afterwards casted to an abstract RFID object, in order to perform proper Read/Write operations. As an illustration static methods like MifareClassic.get(tag) or IsoDep.get(tag) respectively produce MIFARE or ISO14443 instances of RFIDs.

Figure 4 illustrates the detection of an ISO14443 RFID from software activated by an Intent. Upon success ISO7816 requests and responses are exchanged between the application and the device.

IV. Dual Interface RFIDs

A dual interface RFID is a secure microcontroller whose security is enforced by physical and logical countermeasures manage by the embedded operating system. Our experimental platform works with a JCOP41 device.

A. About Secure Microcontroler

Secure microcontrollers are electronic chips including CPU, RAM, and nonvolatile memory such as E^2PROM or FLASH [11]. Security is enforced by various physical and logical countermeasures, driven by a dedicated embedded operating system. According to [12] about 5,5 billions of such devices were manufactured in 2010, mainly as SIM cards (75%) and banking cards (15 %). The format of information exchanges with these components is detailed by the ISO7816 standard. It comprises requests and responses whose maximum size is about 256 bytes. Multiple communication interfaces are supported, including ISO7816 serial port, USB, and NFC radio link.

Most of operating systems implement a Java Virtual Machine, executing a standardized subset of the JAVA language (see next section). Among them, JCOP (standing for Java Card OpenPlatform) was designed by an IBM Zurich research team [13], and since 2007 is supported by the NXP company.

B. About JCOP

According to [14][15] it uses a Philips hardware chip (from the P5CT072V0P family) composed of a processing unit, security components, I/O ports, volatile and non-volatile memories (4608 Bytes RAM, 160 KBytes ROM, 72 KBytes E^2PROM), a random number generator, and crypto co-processors computing Triple-DES, AES and RSA procedures. This component also embeds an ISO 14443 contactless radio interface.

The JCOP41 operating system (see figure 5) includes a Java Virtual Machine (JVM) implemented over the physical platform via facilities of a Hardware Abstraction Layer (HAL). A JVM works with a subset of the java language; it supports a JavaCard Runtime Execution (JCRE) for Applet processing and is associated with a set of packages standardized by the Java Card Forum (JCF). These software libraries provide cryptographic resources (SHA1, MD5, RSA…), and management of information transfer over the radio interface.

An application is a set of Java classes belonging to the same package, executed by the JCRE. It is downloaded and installed thanks to the Card Manager component, whose structure is defined by the Global Platform (GP) standard. The security of this process is based on symmetric cryptographic procedures enforcing mutual authentication, data transfer confidentiality and integrity.

Our application is written for such javacards, with a memory footprint of about 20 Kbytes; it manages TLS sessions with remote WEB server and transfers keys values in the Mifare sectors.

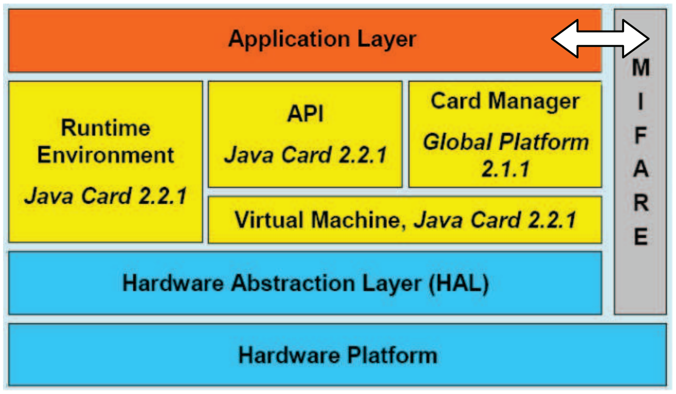

Figure 5. The JCOP41 Operating system

C. Mifare Emulation

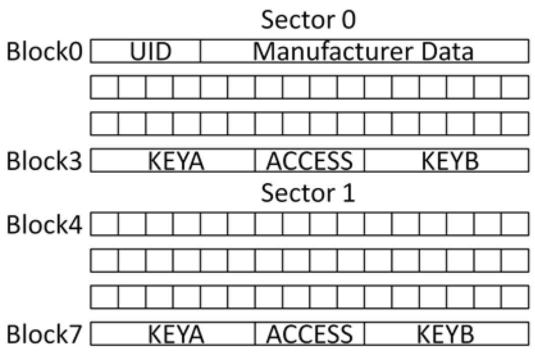

Figure 6. MIFARE Memory Structure

A Classic Mifare 1K device [6] is a chip working with a TypeA radio interface, which includes a secure 1Kbits E^2PROM. This memory is organized in 16 sectors with 4 blocks of 16 bytes each. Blocks are identified by an index ranging from 0 to 63. The fourth block of every sectors (the sector trailer) stores two 48 bits keys (KeyA and KeyB) ; the remaining four bytes define the access conditions for the blocks. Read and Write operations may be free or controlled by authentication procedures dealing with KeyA or KeyB. The block number 0, named Manufacturer Block, contains a four bytes Unique Identifier (UID) and eleven bytes of manufacturer data.

The authentication process uses a three pass protocol based on so-called the Crypto-1 stream cipher, and two random number produces by the reader and the Mifare card.

- The reader selects a sector to be accessed and chooses Key A or Key B.

- The card reads the secret key and the access condition from the sector trailer. It sends a random number r1

- The reader computes a response from the secret key and the random number r1. It transmits a response with an random number r2

- The cards checks the response and computes a new value from r2

The Crypto-1 cipher consists of a linear feedback shift register (LFSR) and filters function. A reverse engineering was performed and attacks published in [16]. In a brute-force attack an attacker records two challenge response exchanged between the legitimate reader and a card. This attack takes under 50 minutes for trying 2^{48} keys values using a dedicated FPGA chip

Never less this device is still widely used for ticketing or keying services. Authentication weakness impact is reduced when these RFIDs store cryptographic tokens that can be freely read, such as those written in magnetic stripes for opening locks.

D. *Mifare Passwords*

A dual interface RFID supports both Mifare and ISO 14443 radio protocol. It is often useful for the operating system (i.e. javacard applications) to write or read data in Mifare blocks. A dedicated API performs this task; for security reasons the knowledge of KeyA or KeyB is not required. Instead of this value, a parameter called the Mifare Password (MP, [14]) is computed according to the following relation:

$$MP = h(IV) = DES_{DKEY1} \text{ o } DES^{-1}_{DKEY2} \text{ o } DES_{DKEY1} (IV)$$

Where the parameter IV is an 8 bytes null value, and DKEY1 and DKEY2 are two DES keys (56 bits each) built from KeyA or KeyB (48 bits) with 8 bits of padding set to zero.

In other words MP is computed with a one way (h) function applied to KeyA or KeyB; the knowledge of its value gives access to Mifare sectors.

But the calculation of $h^{-1}(MP)$ according to a brute force method requires 2^{48} iterations.

The JCOP operating system includes a Mifare API, which comprises only one method:

```
short JZSystem.readWriteMifare
(short mode, byte[] data, short offset, short mifareBlock)
```

The parameters have the following meaning:

- *mode*: read or write access to a particular block.

- *data*: Data storage in RAM holding the (8 bytes) Mifare Password, and providing further 16 bytes of space to carry the data to be read or written

- *offset*: Offset into the data storage provided above, indicating where the first byte of the Mifare Password resides.

- *mifareBlock*: the Mifare block number to be read or written.

The method returns a null value if access was successful.

V. EAP-TLS SMARTCARD

A. *About EAP-TLS contactless smartcard*

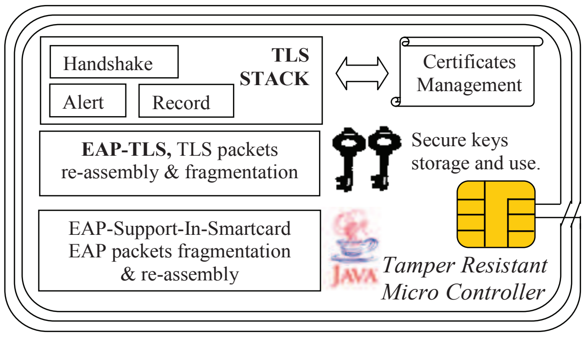
Handshake
TLS STACK
Alert
Record
Certificates Management
EAP-TLS, TLS packets re-assembly & fragmentation
Secure keys storage and use.
EAP-Support-In-Smartcard EAP packets fragmentation & re-assembly
JAVA
Tamper Resistant Micro Controller

Figure 7. The EAP-TLS application for contactless smartcard

The SSL (or Secure Socket Layer) and its IETF standardized version TLS (Transport Layer Security) is the de facto standard for the Internet security. The EAP-TLS protocol [18] was initially designed for authentication purposes over PPP links. It is today widely used in IEEE 802.1x compliant infrastructures (such as Wi-Fi networks) and is supported by the IKEv2 protocol for opening IPSEC secure channels. One of its main benefits is the transport of SSL/TLS messages in EAP (Extensible Authentication Protocol) packets, according to a datagrams paradigm. Therefore it enables the deployment of SSL/TLS services without TCP/IP flavors, and consequently is well suited for secure microcontroller computing platform.

The functionalities of the EAP-TLS embedded application are detailed by an IETF draft [19]. More details may be found in [20] and [21].

The architecture of an EAP-TLS application is illustrated by figure 7.

The EAP protocol provides fragmentation and reassembly services. TLS packets maximum size is about 32768 (2^{14}) bytes. They are split in smaller EAP messages working with an acknowledgment mechanism. A second optional segmentation process divides EAP data unit in small blocks (whose length is less than 256 bytes) compatible with the NFC read/write operations over the radio link.

The TLS stack is equipped with an X509 certificate and a RSA private key used for client's authentication in the TLS full mode, illustrated by figure 7 (left part).

A session is initially opened according to a four way handshake (the full mode, see figure 8, left part) in which client and server are mutually authenticated by their certificates. At the end of this phase (the Phase I according to figure 8) a master key has been computed, cryptographic algorithms have been negotiated for data privacy and integrity, and a set of associated ephemeral keys (referred as the keys-block) has been released. These keys are exported from the smartcard to the Android phone that afterwards manages the TLS session, and which typically performs HTTP encryption and decryption operations (refereed as Phase II by figure 8)

The TLS resume mode works with a previously computed master secret, according to a three ways handshake (see figure 8, right part). It is based on symmetric cryptographic, and reduces the computing load on the server side; by default a WEB server uses a full session only every 10 minutes. A resume session is opened by the EAP-TLS application, which afterwards transfers the keys-block to the mobile phone that performs Phase II procedure.

It is important to notice that the TLS master secret is never exported from the smartcard and remains securely stored in the device.

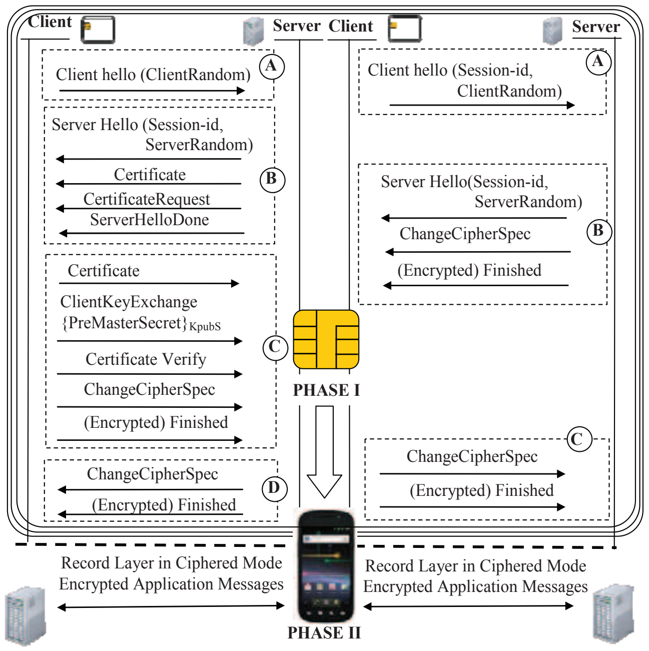
Client
Server
Client
Server
Client hello (ClientRandom)
A
Server Hello (Session-id, ServerRandom)
Certificate
CertificateRequest
ServerHelloDone
B
Certificate
ClientKeyExchange {PreMasterSecret}KpubS
C
Certificate Verify
ChangeCipherSpec
(Encrypted) Finished
ChangeCipherSpec
(Encrypted) Finished
D
Client hello (Session-id, ClientRandom)
A
Server Hello(Session-id, ServerRandom)
ChangeCipherSpec
B
(Encrypted) Finished
ChangeCipherSpec
C
(Encrypted) Finished
PHASE I
Record Layer in Ciphered Mode Encrypted Application Messages
Record Layer in Ciphered Mode Encrypted Application Messages
PHASE II

Figure 8. Phase I and phase II during an TLS session

B. *Performancs issues*

Because RFIDs are low power consumption devices, with small computing resources, and furthermore are driven by operating systems that manage countermeasures, computing performance is a critical topic. Some basic parameters are given by figure 9.

IO Throughput ms/ byte	MD5 ms/ block	SHA1 ms/ block	RSA 1024 PUB ms	RSA 1024 PRIV ms
0,125	2,0	4,0	26,0	120,0

Figure 9. Basic performances of the JCOP41 device

The four ways handshake (Phase I) of a full TLS session (with RSA 1024 bits) costs 11,7s. It requires one encryption with the private key (120 ms) two computations with public keys (2x 26 ms). About 230 MD5 (230 x 2 ms) and SHA1 (230 x 4ms) calculations (dealing with 64 bytes blocks) are performed. It exchanges 2,500 bytes, whose transfer costs 0,125 x 2500 = 310 ms. The remaining time (9,8s = 11,7 – 1,9) is burnt by the java code execution.

The three ways handshake (Phase I) of a resume session consumes 2,6s. It needs the exchange of 250 bytes (250x 0,125 = 31 ms), and the processing of 75 MD5 and SHA1 that consumes 450ms. The remaining time (2,1s= 2,6 – 0,5) is spent in the java code execution.

These experimental results show that most of the computing time (with our JCOP device) is burnt by the embedded virtual machine. However this is a not a general behavior and tests with other javacards (running the same application) present different figures, in which most of computing times are consumed by cryptographic resources.

C. *JAVA API*

An EAP-TLS device is associated with a high level JAVA API, quite similar to the well-known OPENSSL environment.

Two main java objects are defined, the tls-tandem class that constitutes the core framework for the management of TLS session with a EAP-TLS RFID, and the recordlayer class created at the end of the Phase I TLS handshake, and which performs all (HTTP) encryption decryption/operations. The figure 12 illustrates the use of these APIS in our key service.

The tls-tandem class is made of three main methods:

- public tls_tandem(int mode, ReaderSC reader, String aid, String pin, String identity)

- public recordlayer OpenSession(String ServerName, short Port)

- public void CloseSession(recordlayer RecordLayer)

The constructor of the tls-tandem class setups the software environment needed for TLS operations with external RFID, with the following parameters,

- *mode* is the role of the RFID (either TLS CLIENT or TLS SERVER)

- *reader* is an abstract representation of the RFID reader, based on the NFC Android model. This object detects the presence of an external RFID feed by the reader, and provides support for IO operations.

- *aid*, is the Application Identifier for the application store in the RFID. According to the ISO7816 standard, this identifier size ranges between 5 to 16 bytes.

- *pin* is the optional PIN required for the RFID activation.

- *identity* is an optional alias that identifies the set of parameters (client's certificate, private key, CA Certificate) to be used by the TLS stack.

The OpenSession method performs Phase I handshake with a remote TLS server identified by its name and its port (usually 443). It returns a recordlayer object initialized with the appropriate cryptographic parameters.

The CloseSession method deletes the TLS framework and the associated resources.

The recordlayer object is created upon a successful TLS Phase I handshake and realizes Phase II operations. It comprises five main methods.

- public byte [] encrypt(byte[] msg). Perform data (typically a HTTP request) encryption and integrity operations, and return a TLS formatted packet.

- public byte[] send(). Transmit a TLS packet over the TCP socket.

- public byte[] recv(). Receive a TLS packet from the TCP socket.

- public byte [] decrypt(byte[] msg). Decrypt a TLS packet and check its integrity.

- public void Close(). Close the TLS session.

VI. THE KEY DELIVERING USE CASE

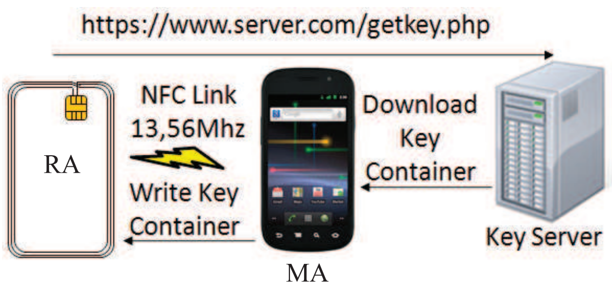

Figure 10. The Key Delivering Architecture

The key mobile service architecture is illustrated by figure 10. A subscriber owns an Android NFC Smartphone and contactless dual interfaces RFID embedding a Javacard application (RA) that performs key downloading. The mobile is equipped with a dedicated application (MA) implementing features detailed in sections III and IV. All Dalvik applications must be signed, but the Android operating system allows software downloading from entrusted source, i.e. which are not available from the Android Market store.

The key delivering process is summarized by figure 11.

Upon detection of the RFID by the Smartphone, the user is prompted to select and start the appropriate (RA) application. The TLS stack embedded in the RFID is activated, the mobile application (MA) opens a TCP socket with the remote server, and thereafter supervises the TLS handshake Phase I between the RFID and the key server. Upon success the keys-block computed by the RFID is transferred to the mobile which fully manages the TLS session Phase II.

The mobile application builds an HTTP request transmitted over the TLS session, i.e. the key repository is identified by an URL such as https://www.server.com/getkey.php.

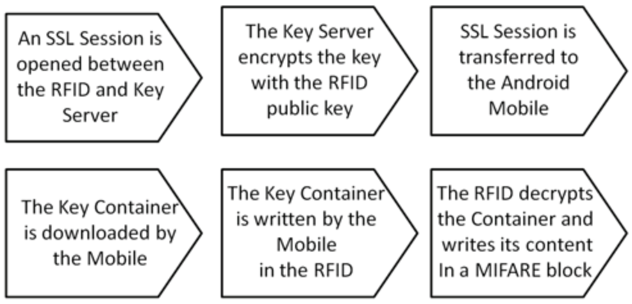
An SSL Session is opened between the RFID and Key Server
The Key Server encrypts the key with the RFID public key
SSL Session is transferred to the Android Mobile
The Key Container is downloaded by the Mobile
The Key Container is written by the Mobile in the RFID
The RFID decrypts the Container and writes its content In a MIFARE block

Figure 11. The Key Delivering Process

The requested file is located in a server area for which mutual TLS authentication is mandatory. Therefore the RFID is identified by its embedded certificate dynamically recorded (on the key server side) by the PHP variable $_SERVER['SSL_CLIENT_CERT'].

The getkey.php script uses the well known OPENSSL facilities in order to extract the client's RSA public key. It then builds a data structure that we call the Key Container (KC), which securely stores a set of data (the lock key, LK) to be written in one or several Mifare blocks.

A container is made of two parts, a header and a trailer.

- The header is the encrypted value of the key (LK) with the RFID public RSA key, according to the PKCS #1 standard.

- The trailer contains a PKCS#1 signature of the header with a private key whose certificate is trusted by the RFID EAP-TLS application (RA).

The hexadecimal ASCII dump of the container is returned in the body of the HTTP response, which is collected by the mobile phone.

Finally the Key Container is pushed by the mobile application to the RFID. This later verifies the signature with signatory's public Key, and decrypts the LK value with its private keys. It then uses the Mifare API and the associated Mifare Password to write LK in the appropriate blocks.

The mobile software main module (MA) is illustrated by figure 12. Upon detection of the RFID device, a tls-tandem framework is created, which start the RA application.

Afterwards the https class starts the TLS Phase I handshake with the key server, by invoking the OpenSession method that returns a record-layer object. This last mentioned performs encryption and decryption operations for the HTTP request and the associated response.

The body of the HTTP response, i.e. the ASCII representation of the key container is then written in the RFID, which afterwards internally deciphers its content, checks its authenticity and finally writes the key value in a Mifare block.

The dual interface contactless card is ready for opening its paired electronic lock.

```
private void resolveIntent (Intent intent) throws
IllegalArgumentException, IllegalAccessException {
String action = intent.getAction();

if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(action))
{ Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
TagI = IsoDep.get(tag)   ;
if (TagI==null) { Box("Error","Non ISO Tag"); return;}

ReaderNFC  myReader= new ReaderNFC(TagI);
tls-tandem myTlsTandem=
new tls-tandem(tls_tandem.CLIENT,myReader,
"A0000000300002FFFFFFFF8931323800","0000","client");

https reg = new https(myTlsTandem);
if (!reg.HTTPS(="www.server.com","/tandem/secure/getkey.php"))
Box("Error","HTTPS Error!!!");

String apdu1 = "00D0 8001 80 " + reg.body.substring(0,128);
String apdu2 = "00D0 8002 80 " + reg.body.substring(128,256);
String apdu3 = "00D0 8002 80 " + reg.body.substring(256,384);
String apdu4 = "00D0 8003 80 " + reg.body.substring(384,512);

myTlsTandem.SC_reader_socket.Echo_TAPDU(apdu1);
myTlsTandem.SC_reader_socket.Echo_TAPDU(apdu2);
myTlsTandem.SC_reader_socket.Echo_TAPDU(apdu3);
myTlsTandem.SC_reader_socket.Echo_TAPDU(apdu4);
myTlsTandem.Close(null); }

else { Box("Error","No card"); return; }
}

public class https {
tls-tandem tandem = null ;
record-layer RecordLayer=null;
 public String body=null;

public https(tls-tandem mytandem) {
tandem =mytandem;}

public boolean HTTPS(String host, String myfile)
{ boolean done;
if (tandem == null) return false;
RecordLayer = tandem.OpenSession(host,(short)443);
if (RecordLayer == null) return false;
done= http(myfile);
RecordLayer.Close();
return done ;
}
boolean http(String filename){…}
}
```

Figure 12. Mobile Software Main Application

VII. DISCUSSION AND FURTHER WORK

The main advantage of the experimental platform presented in this paper is the compatibility with legacy solutions based on Mifare RFIDs.

Keys card working with magnetic strips, are freely read and written, and may be easily copied. Low cost RFIDs (such as Mifare UltraLight devices cards), working with one time programmable memories (i.e. memories in which a bit is irreversibility set), provide a more reliable solution incentive to hazardous magnetic interactions.

From a security point of view, RFIDs (for example the Mifare 1K) whose reading is protected by mutual authentication procedures realize anti-cloning mechanisms. However the effectiveness of this legacy feature is questionable, because of recent attacks on the Mifare Crypto-1 cryptographic algorithm.

Nerveless, by equipping a key card with a TLS/SSL stack, including a certificate and a private key we give an identity to this device. The container facility defined in section VI establishes an asynchronous security channel between the key server and the key card. Thanks to its mobile phone, delivering Internet and NFC connectivity, the client collects a lock key at every time from everywhere. A PIN code may be applied in order to enforce a dual form factor authentication, i.e. something you know (the PIN code) and something you have (the dual interface key card).

A restriction of the key delivering to a specific location could be easily supported, by only allowing the service from Wi-Fi network located in the hotel lobby.

Hotel rooms are of course a natural use case for electronic locks, because they are frequently booked, so that key renewal is needed for privacy issues, which can’t be fulfilled by classical mechanical devices. But others applications could be targeted, for example car rental (more and more cars are equipped with electronic locks), office rooms, and more generally speaking all services that require entrance authorizations for a limited amount of time.

A next generation of key cards could avoid today security threats by implementing strong authentication mechanisms between the card and the lock. For example in the Javacard technology it is possible to share objects between applications. According to this feature a key card could embed two applications first for key delivering (with a TLS stack), and second for interaction with the electronic lock.

An exciting perspective should be the use smartphones as universal keys. The main issue deals with security, i.e. is this class of devices enough secured for keys downloading and storage?. They already include secure microcontrollers (usually named Secure Elements) such as USIM modules or NFC Controllers. These components are compatible with the javacard technology, and therefore may securely execute embedded applications. The two main problems that arise are memory management and arbitration (what applications may be stored and executed), and download control (what entity controls/authorizes applications located in the secure elements).

From a trust point of view, loyalty cards delivered by hotels and equipped with RFID devices appear as a legitimate device for keys delivering and use. For practical reasons, a mobile application could be common to all kinds of such loyalty cards. We plan to test this new mobile service in a few months, with industrial partners specialized in electronic locks and hotel management systems.

VIII. Conclusion

In this paper we presented a new key delivering platform working with dual interfaces smartcards and Android mobile. It is a new class of mobile applications in which the user is equipped with a RFID and a smart-phone. The RFID access to the Internet via an application running on the mobile, but manages the security of the service. It is afterwards autonomously used, what avoids the lack of battery issue.

References

[1] http://www.gartner.com/it/page.jsp?id=1622614

[2] NFC Forum Specifications, http://www.nfc-forum.org/specs/

[3] The Classic RFID VingCard technology, http://www.vingcard.com/page?id=4380

[4] Sypin, E.V.; Tunin, K.A.; Negodiaev, V.P.; Povernov, E.S.; "The electronic lock on disposable telephone cards", Proceedings of 4th Annual Workshop on Electron Devices and Materials, Siberian Russian , 2003.

[5] Peusaari J., Kelkka R., Ikonen J., "An access control and time management software solution using RFID" CompSysTech '09 Proceedings of the 10th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing, 2009.

[6] Mifare Standard Card IC MIF1 IC S50, Functional Specification, Revision 5.1, Philips seminconductors, May 2001

[7] International Civil Aviation Organization, "Machine Readable Travel Documents", ICAO Document 9303, Part 1,2,3

[8] "What is android ?", http://developer.android.com/guide/basics/what-is-android.html

[9] Hassan, Z.S.; "Ubiquitous computing and android", Third International Conference on Digital Information Management, 2008. ICDIM 2008.

[10] Enck, W.; Ongtang, M.; McDaniel, P.; "Understanding Android Security ", IEEE Security & Privacy, Volume: 7 , Issue: 1, 2009

[11] Jurgensen, T.M. ET. al., "Smart Cards: The Developer's Toolkit", Prentice Hall PTR, 2002, ISBN 0130937304

[12] http://www.eurosmart.com/

[13] Baentsch, ET All, "JavaCard-from hype to reality", Concurrency, IEEE Volume: 7, Issue: 4

[14] Certification Report, BSI-DSZ-CC-0348-2006 for Philips Secure Smart Card Controller P5CT072V0P, P5CC072V0P, P5CD072V0P and P5CD036V0P each with specific IC Dedicated Software from Philips Semiconductors GmbH Business Line Identification,https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Reporte03/0348a_pdf.pdf?__blob=publicationFile

[15] Certification Report, BSI-DSZ-CC-0426-2007 for NXP P541G072V0P (JCOP 41 v2.3.1) from IBM Deutschland Entwicklung GmbH http://www.commoncriteriaportal.org/files/epfiles/0426a.pdf

[16] Nohl, K.; Evans, D.; Plotz, S.; Plotz, H., "Reverse-Engineering a Cryptographic RFID Tag", USENIX Security Symposium. San Jose, 2008.

[17] AN02105, Secure Access to Mifare Memory, on Dual Interface Smart Card ICs, Application Note, Philips seminconductors, January 2002

[18] RFC 2716, "PPP EAP TLS Authentication Protocol". October 1999.

[19] IETF draft, , "EAP-Support in Smartcard", August 2011.

[20] Urien P., "Tandem Smart Cards: Enforcing Trust for TLS-Based Network Services", Eighth International Workshop on Applications and Services in Wireless Networks, ASWN '08., 2008

[21] Urien P., "Collaboration of SSL smart cards within the WEB2 landscape", In proceeding of CTS'09, 2009

[22] Urien, P.; "OpenID Provider based on SSL Smart Cards", in proceedings of IEEE CCNC 2010, 2010

[23] Urien P., Kiennert C., "A New Key Delivering Platform Based on NFC Enabled Android Phone and Dual Interfaces EAP-TLS Contactless Smartcards", posters session, in proceddings of Third International Conference on Mobile Computing, Applications and Services, MOBICASE 2011.