# A New Key Delivering Platform Based on NFC Enabled Android Phone and Dual Interfaces EAP-TLS Contactless Smartcards

Pascal Urien and Christophe Kiennert

Telecom ParisTech, 23 avenue d'Italie Paris 75013, France
EtherTrust, 62b rue Gay Lussac Paris, 75005, France
Pascal.Urien@Telecom-ParisTech.fr,
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 legacy locks. Keys cards securely download keys value from dedicated WEB server, thanks to Internet and NFC connectivity offered 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.

## 1 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. Twelve 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.

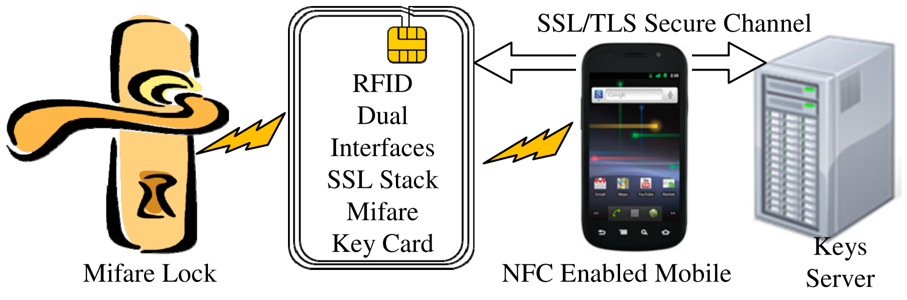

**Fig. 1.** RFID Lock, new mobile service

© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2012

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 fed 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.

This paper presents an experimental mobile service targeting key delivering for electronic locks. In the legacy service (see for example [3]) electronic locks are equipped with RFID readers, and work with RFID cards (frequently including Mifare [4] components) in spite of magnetic strip cards. A device named the Card Encoder, belonging to a dedicated information system, writes keys values in RFID cards. Our new experimental platform (see figure 1) 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 smart-phone securely collects a key from the dedicated WEB server. This paper is constructed according to the following outline. Section 2 introduces the NFC and Android technologies. Section 3 describes dual interfaces RFIDs. Section 4 presents basic concepts of the EAP-TLS application for contactless smartcard and gives performances for the prototype platform. Section 5 details the new and secure key delivering services.

## 2 About The NFC Technology and Android

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 to 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 fed 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 of about 2,2V on a rectangular loop with an area of 5x8 cm$^2$. 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 [4] (1K meaning one KBytes of memory) widely deployed for ticketing applications or RFIDs keys [3]. Electronic passports (normalized by the ICAO standards [5]) include RFIDs belonging to the second category either typeA or typeB.

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 embed 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.

Android [6] [7] [8] is an operating system originally created by the company Android Inc. and supported by the Open Handset Alliance, driven by the Google company. It uses 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. An Activity is an application component that manages a screen with which users can interact in order to do something. It 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. The Android version 2.3, also named Gingerbread, supports NFC software APIs, building an abstract framework over a NFC adapter chip.

## 3 Dual Interfaces RFID

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

Secure microcontrollers are electronic chips including CPU, RAM, and non-volatile memory such as E2PROM or FLASH [9]. Security is enforced by various physical and logical countermeasures, driven by a dedicated embedded operating system. According to [10] 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 [11], and since 2007 is supported by the NXP Company.

According to [12] it uses a Philips hardware chip composed of a processing unit, security components, I/O ports, volatile and non-volatile memories (4608 Bytes RAM, 160 KBytes ROM, 72 KBytes E2PROM), 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 2) 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.

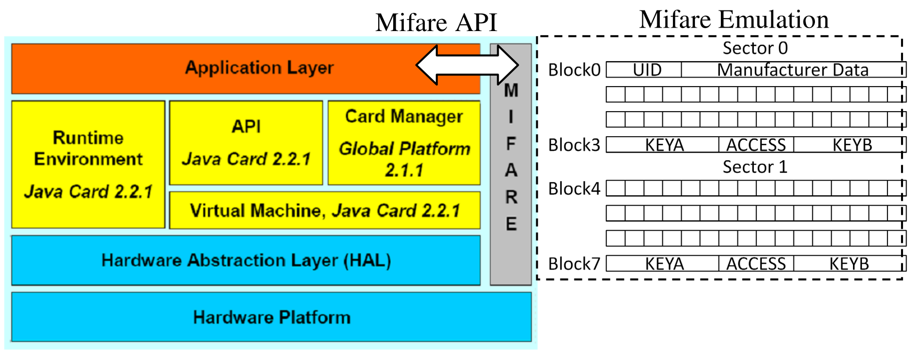

**Fig. 2.** The JCOP41 Operating system

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. 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.

A Classic Mifare 1K device [4] is a chip working with a TypeA radio interface, which includes a secure 1KBytes E2PROM. This memory (see figure 2, right part) 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 the so-called Crypto-1 stream cipher, and two random numbers produced by the reader and the Mifare card. A reverse engineering was performed and attacks published in [13]. 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. Nevertheless, 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.

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:

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

Where the parameter IV is an 8 bytes null value, and $D_{KEY1}$ and $D_{KEY2}$ are two DES keys (56 bits each) built from KeyA or KeyB (48 bits) with 8 bits of padding set to zero. The JCOP operating system includes a Mifare API, which internally performs Read/Write operations with the knowledge of the MP.

## 4 EAP-TLS 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 [15] 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 [16]. More details may be found in [17] and [18]. The EAP protocol provides fragmentation and reassembly services. TLS packets maximum size is about 16384 ($2^{14}$) bytes. 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 3 (left part).

A session is initially opened according to a four way handshake (the full mode, see figure 3, 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 3) 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 (referred as Phase II by figure 3).

The TLS resume mode works with a previously computed master secret, according to a three ways handshake (see figure 3, right part). It is based on symmetric cryptography, 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.

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.

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.

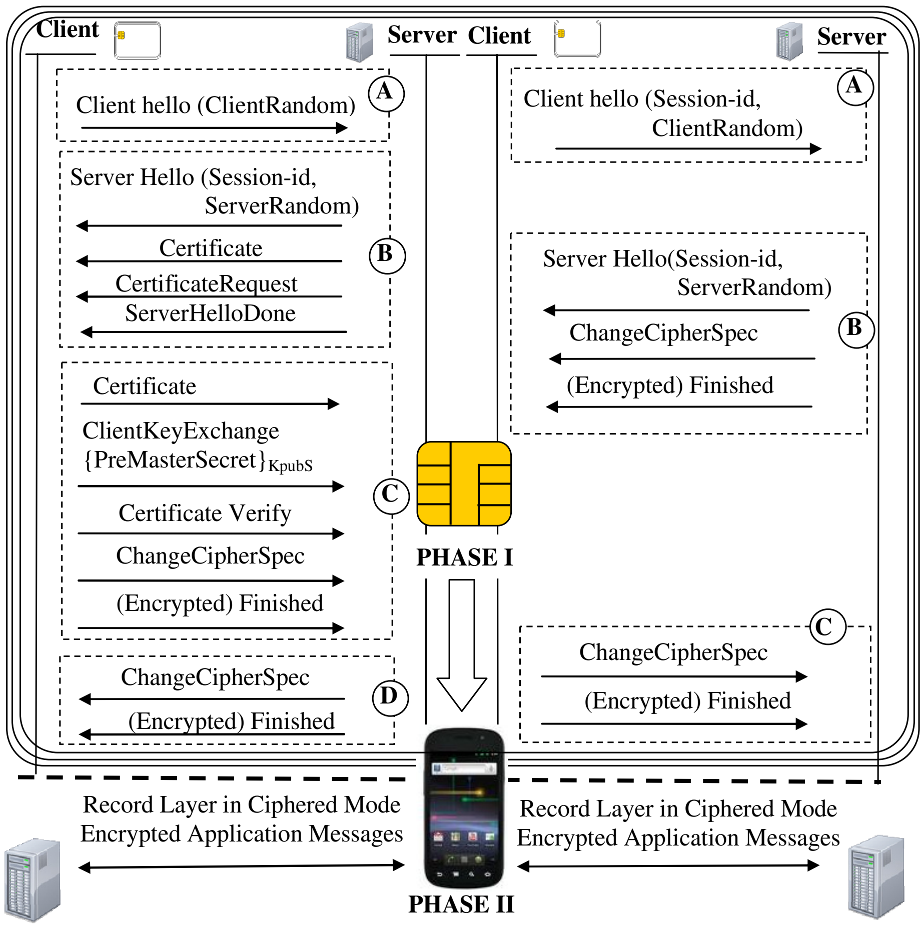
Client
Server
Client
Server
Client hello (ClientRandom)
A
Server Hello (Session-id, ServerRandom)
Certificate
CertificateRequest
ServerHelloDone
B
Certificate
ClientKeyExchange {PreMasterSecret}KpubS
Certificate Verify
ChangeCipherSpec
(Encrypted) Finished
C
ChangeCipherSpec
(Encrypted) Finished
D
Client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

**Fig. 3.** Phase I and Phase II operations during a TLS session, using an EAP-TLS smartcard

These experimental results show that most of is spent in the embedded JVM. However this is a not a general behavior and other javacards present different figures, in which most of computing times are consumed by cryptographic resources.

## 5 The Key Delivering Use Case

The key mobile service architecture is illustrated by figure 4. 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). 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.

Upon detection of the RFID by the Smartphone, the user is prompted to select and start the appropriate (MA) 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.

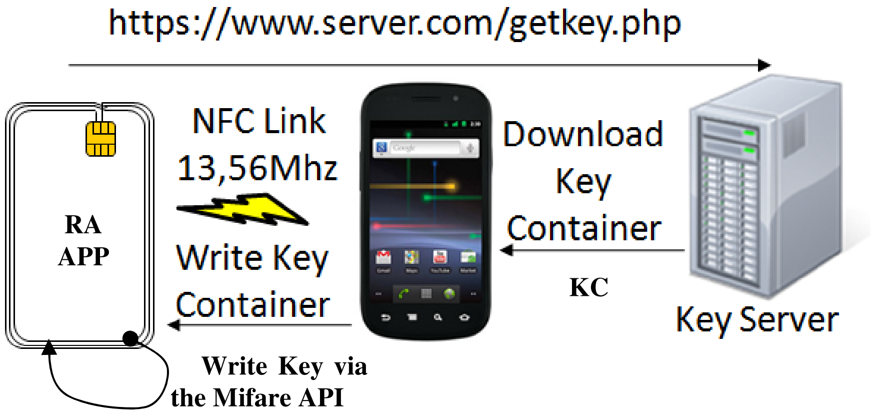

**Fig. 4.** The Key Delivering Service

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. 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.

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

## 6 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 accesses the Internet via an application running on the mobile, but manages the security of the service. It is afterwards autonomously used, which 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. Mifare Standard Card IC MIF1 IC S50, Functional Specification, Revision 5.1, Philips semiconductors (May 2001)
5. International Civil Aviation Organization. Machine Readable Travel Documents, ICAO Document 9303, Part 1,2,3
6. What is android, http://developer.android.com/guide/basics/what-is-android.html
7. Hassan, Z.S.: Ubiquitous computing and android. In: Third International Conference on Digital Information Management, ICDIM 2008 (2008)
8. Enck, W., Ongtang, M., McDaniel, P.: Understanding Android Security. IEEE Security & Privacy 7(1) (2009)
9. Jurgensen, T.M., et al.: Smart Cards: The Developer's Toolkit. Prentice Hall PTR (2002) ISBN 0130937304
10. http://www.eurosmart.com/
11. Baentsch, M., Buhler, P., Eirich, T., Horing, F., Oestreicher, M.: JavaCard-from hype to reality. IEEE Concurrency 7(4)
12. Certification Report, BSI-DSZ-CC-0426-2007 for NXP P541G072V0P (JCOP 41 v2.3.1) from IBM Deutschland Entwicklung GmbH, http://www.commoncriteriaportal.org
13. Nohl, K., Evans, D., Plotz, S., Plotz, H.: Reverse-Engineering a Cryptographic RFID Tag. In: Proceeding of USENIX Security Symposium, San Jose (2008)
14. AN02105, Secure Access to Mifare Memory, on Dual Interface Smart Card ICs, Application Note, Philips semiconductors (January 2002)
15. RFC 2716, PPP EAP TLS Authentication Protocol (October 1999)
16. IETF draft, EAP-Support in Smartcard (August 2011)
17. Urien, P.: Tandem Smart Cards: Enforcing Trust for TLS-Based Network Services. In: Proceeding of ASWN 2008 (2008)
18. Urien, P.: Collaboration of SSL smart cards within the WEB2 landscape. In: Proceeding of CTS 2009 (2009)