NFC Peer to Peer Secure Services for Smart Cities: LLCPS Concepts and Experiments with Smartphones

Pascal Urien(✉)

Telecom ParisTech, 23 Avenue d'italie, 75015 Paris, France
Pascal.Urien@Telecom-ParisTech.fr

Abstract. Proximity communication technologies, such as NFC (*Near Field Communication*) are today widely deployed in smart city environments. Contactless services based on NFC facilities are used for payment, transport or access control applications. There are supported by most of mobile operating systems. The NFC *Peer to Peer* mode is typically used for pushing small pieces of information in applications like Android Beams, or Personal Health Device Communication (PHDC), a family of devices comprising blood pressure meters, blood glucose meters, or body weight scales. Security (i.e. mutual authentication, data privacy and integrity) is a critical topic for P2P exchanges; however it is not specified by today standards. In order to solve these issues we introduced a security protocol (LLCPS) compatible with NFC standards and based on the well known TLS protocol. This Chapter describes an experimental platform built with commercial smartphones and presents some performances.

Keywords: NFC · P2P · Security · Iot · TLS · LLCP · LLCPS

1 Introduction

Proximity communication technologies, such as NFC (*Near Field Communication* [1]) are today widely deployed in smart city environments. Contactless services based on NFC facilities are used for payment, transport or access control applications. Major cities like Paris (*Navigo card*), London (*Oster card*), Venezia (*Imob card*), Seoul (*T Money Card*) use NFC tickets for their transport networks. NFC interfaces are more and more supported by EMV bank cards for contactless payments. Millions of electronic locks controlling hotels rooms, office doors, parking entrances are working with NFC key cards typically based on tags such as *Mifare Ultralight* [8].

Therefore smart cities already include multiple NFC kiosks connected to various information systems. During the last decade mobile devices powered by operating systems such as Android, RIM or Windows have been manufactured with NFC interfaces (according to [2] two in three phones to come with NFC in 2018). There is a trend to replace NFC card and tickets by application running in smartphones with internet connectivity. As an illustration, the EMVCO consortium is working on new payment technology called tokenization [9] compatible with NFC-enabled mobiles.

R. Giaffreda et al. (Eds.): IoT360 2014, Part II, LNICST 151, pp. 297–305, 2015.
DOI: 10.1007/978-3-319-19743-2_40

NFC has three working modes Reader/Writer, Card Emulation, and Peer to Peer (P2P). The two first are imported from legacy applications working with contactless devices powered by reader. The P2P mode [3] deals with two devices (the target and the initiator) managing their own power feeding (i.e. including batteries). It is used for pushing small pieces of information in applications like *Android Beams* working with the SNEP [5] protocol, or *Personal Health Device Communication* [7], a family of devices such as blood pressure meters, blood glucose meters, or body weight scales. Security (i.e. mutual authentication, data privacy and integrity) is a critical topic for P2P exchanges; however it is not specified by today standards. In order to solve these issues we introduced in [14, 16] a security protocol compatible with NFC standards and based on the well known TLS protocol. This chapter describes an experimental platform built with commercial smartphones and details some performances; a demonstration of this prototype was performed in [15].

This chapter is constructed according to the following outline. Section 2 recalls the main characteristics of the NFC P2P mode; it introduces the LLCP [4] protocol and the SNEP [5] service. Section 3 introduces LLCPS [14], a secure P2P protocol based on TLS. Section 4 presents the experimental platform; made with a smartphone and a NFC reader, and its software architecture. Section 5 summarizes experimental performances for operations secured by asymmetric procedures (RSA and X509 certificates, anonymous Diffie-Hellman over elliptic curves) and symmetric procedures (i.e. the TLS abbreviated mode). Finally Sect. 6 concludes this chapter.

2 About NFC Peer to Peer

The *Near Field Communication* (NFC) protocol is a proximity communication technology based on inductive coupling. According to the *Lenz* law a magnetic field of 5 A/m, pulsed at the 13,56 MHz frequency, induced a voltage of about 2 V on a loop with an area of 40 cm^2 (5 cm x 8 cm). Another interesting property of inductive coupling is the energy conservation; the energy delivered by the primary circuit (P_{PRI}) is consumed by the secondary (P_{SEC}) circuit according to the relation:

$$P_{PRI} = P_{SEC} = 2\pi f \; M \; I_{PRI} \, I_{SEC}$$

where f is the frequency, M the mutual inductance, I_{PRI} and I_{SEC} respectively the electric current in the primary and secondary circuit. The power feeding of the secondary circuit by the primary one is a physical assumption for devices proximity. NFC supports two functional modes:

- Reader/Writer and Card Emulation. A device named ″Reader″ feeds another device called ″Card″, thanks to a 13,56 MHz electromagnetic coupling. This mode is typically used with contactless smartcards or NFC tags.
- Peer to Peer (P2P). Two devices, the *"Initiator"* and the *"Target"* establish a communication link. In the *"Active"* mode these two nodes are managing their own energy resources. In the *"Passive"* mode the Initiator powers the Target via a 13, 56 MHz electromagnetic field.

In this chapter we focus on the P2P mode which is more and more supported by smartphones running operating such as android, RIM and others.

2.1 About Logical Link Control Protocol (LLCP)

A P2P session occurs in four steps:

(1) *Initialization and Anti-collision*. The Initiator periodically generates a RF field and sends a request packet, acknowledged by a Target response packet. It manages anti-collision mechanisms in order to detect several Target devices.
(2) *Protocol Activation and Parameters Selection*. The Initiator begins a session with a detected Target by forwarding an *Attribute-Request* message, confirmed by a Target *Attribute-Response* message. These messages setup the functional parameters to be used by P2P exchanges.
(3) *Data Exchange*. Frames are exchanged via the *Data Exchange Protocol* (DEP [3]) that provides error detection and recovery. It works with small packets (from 64 to 256 bytes). The Initiator transmits *DEP-Request* acknowledged by *DEP-Response*. These frames typically transport LLCP [4] messages, which are detailed below.
(4) *De-Activation*. The Initiator releases the P2P session with the Target.

The *Logical Link Control Protocol* (LLCP) manages information exchanges during P2P sessions.

It may work according to connection oriented or connectionless paradigms. However all legacy P2P services used a connected paradigm, and this chapter only deals with this mode. The main advantage of connection oriented service is to manage potential traffic congestions, due to operating system overloads, via dedicated packets called RNR (*Receiver Not Ready*); we observed such packets with the RIM (BB10) operating system.

A LLCP packet is transported by a DEP frame. It comprises three mandatory fields, the *Destination Service Access Point*, (DSAP 6 bits), the *Source Service Access Point* (SSAP 6 bits), and *Protocol Type* (PTYPE 4 bits). An optional sequence field (8 bits) contains two (4 bits) numbers N(S) and N(R) respectively giving the number of the information packet to be sent and the number of the next information packet to be received. In the connected mode a service is identified by a name, or an integer (a well-known *Service Access Point*, SAP). A connection is performed via two dedicated LLCP packets, CONNECT and CONNECTION CONFIRM (CC). A CONNECT request to the well-known SAP number 1 (the *Service Discovery Protocol*) is a way to get an ephemeral SAP associated to a service name, and therefore to start a session with this named service. A NFC P2P server processes the CONNECT packet issued by a NFC P2P client. Figure 1 illustrates a connection to a named service, performed between an initiator acting as a NFC client, and a target hosting a NFC server.

2.2 The SNEP Service

The Simple NDEF Exchange Protocol (SNEP, [5]) is a simple service widely used to push data over LLCP. It comprises two main messages SNEP-PUT and SNEP-Success whose name are self-explanatory. A SNEP service is identified either by the well-known SAP number 4, or by the name *urn:nfc:sn:snep*. The information shuttled by SNEP-PUT message is encoded according the NFC Data Exchange Format (NDEF, [7]); this last-mentioned supports multiple formats such as text or URLs. NDEF contents are also hosted by NFC tags (such as the NXP *Mifare Ultralight* [8]), used as transportation tickets or key cards. In a typical SNEP dialog, the initiator detects a target; a P2P session is established, and thanks to LLCP the SNEP client sends a CONNECT request (either to the DSAP 4 or the DSAP 1) to the SNEP server confirmed by a CC packet. Thereafter the SNEP client pushes NDEF content thanks to the SNEP-PUT message, acknowledged in turn by a SNEP-Success. In a smart cities context, SNEP could be used by numerous services such as transportation or access control.

3 About LLCPS

One issue for NFC P2P exchanges is the lack of native security and privacy. Data are exchanged in clear form over the air. There is no data encryption and no pairing (i.e. mutual authentication) between the initiator and the target. The main target of LLCPS (LLCP secure, [14]) is to reuse the TLS protocol over LLCP in order to address these critical issues. TLS (*Transport Layer Security*) is a well-known IETF protocol, which is widely used for the internet security. According to the NFC standards a P2P name is associated to a service; in our context the *urn:nfc:sn:tls:service* string identifies a service such as SNEP whose privacy is enforced by the TLS protocol.

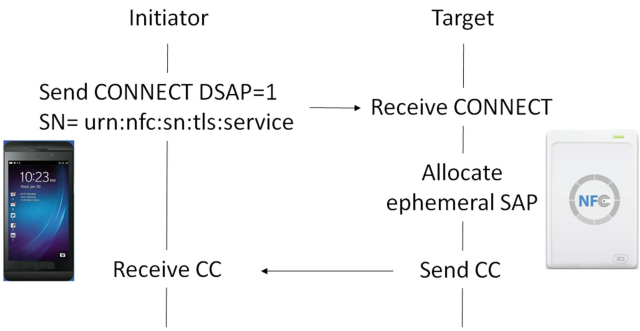

Fig. 1. P2P connection between a client (initiator side) and a server (target side)

TLS has two working modes, the full mode in which the client and the server may be authenticated by X509 certificates and associated private keys, and the abbreviated mode that reuses a previously opened full session, and which only deals with

symmetric cryptography. A full session with strong mutual authentication may be interpreted as a pairing between initiator and target. A shared secret (the *master secret*) is created between these two entities, which is thereafter available for resume sessions, dealing with lighter symmetric cryptographic exchanges. The booting of a full session works with a four ways handshake, while a resume session only requires a three ways handshake. The EAP-TLS protocol [13] demonstrates how TLS messages may be gathered in blocks exchanged according to a half-duplex mechanism. LLCPS builds such TLS blocks, which are segmented in small LLCP packets (typically *128 bytes*) exchanged between initiator and targets devices. TLS supports various cryptographic algorithms such as RSA or Elliptic Curves (ECC). We developed an experimental platform and performed tests with two types of algorithms, RSA with X509 certificates, and anonymous Diffie-Hellman over elliptic curves (ECCDH anonymous).

4 The Experimental Platform

The experimental platform (see Fig. 2) is made of two parts, a smartphone (BB10, [12]) equipped with a NFC interface, and NFC reader (the ACS122, [10, 11]) delivering P2P facilities. The use case is an access control application (ticketing, electronic key) using SNEP, and running over the *urn:nfc:sn:tls:snep* service.

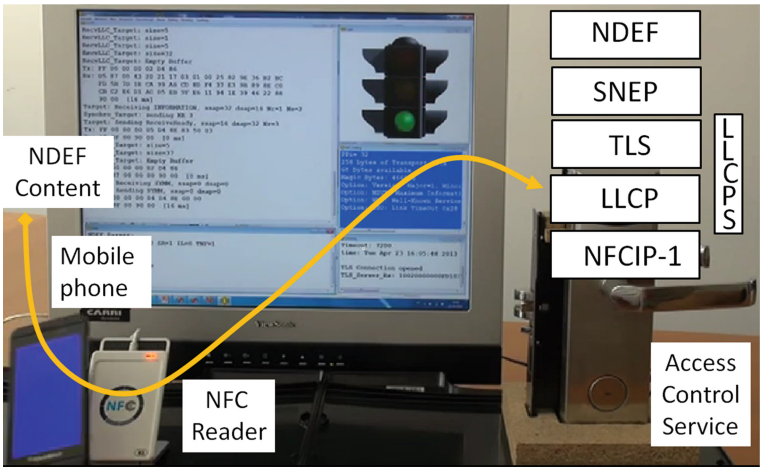

Fig. 2. The experimental LLCPS platform, controlling an electronic lock

4.1 The BB10 Smartphone

The smartphone phone is a BB10 model [12], powered by a POSIX operating system. The mobile application is written in the C language. The mobile embeds classical UNIX software libraries such as SOCKET for the TCP/IP connectivity, OPENSSL for

TLS facilities and LIBNFC for the NFC management. The NFC framework is managed by the following proprietary procedures:

- *nfc-connect()* and *nfc-disconnect()* starts and stops the NFC framework.
- *nfc-llcp-register-connection-listener()* registers the application for a particular NFC service identified by a name such as ″urn:nfc:sn:tls:snep″ in this use case.
- *nfc-llcp-read()* and *nfc-llcp-write()* handle NFC packets reception and transmission over the NFC radio. The maximum packet length is about 128 bytes. TLS messages are segmented and reassembled according to this size.
- The programming model is event driven; main procedures are associated to events such as *LLCP-CONNECTION*, *LLCP-WRITE-COMPLETE*, *LLCP-READ-COMPLETE*.

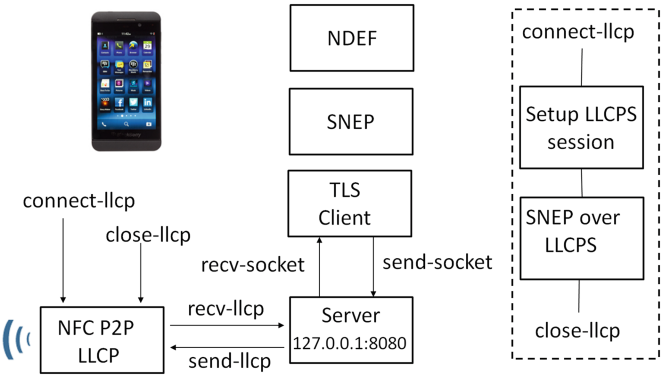

Fig. 3. Software architecture of the mobile application

The Fig. 3 illustrates the mobile application. The mobile acts as a NFC initiator; it periodically generates a RF field, and polls the presence of a target. Upon detection of a remote device, a LLCP session is started with the service *urn:nfc:sn:tls:snep*. A *TLS-Client*, working the OPENSSL library, exchanges data with an internal socket server (using the *127.0.0.1:8080* address) acting as a proxy with a software module (*NFC-P2P-LLCP*) that manages the LLCP session. When the TLS session is established a *SNEP-PUT* packet, transporting a NDEF payload is sent to the target and thereafter acknowledged by a *SNEP-Success* packet.

4.2 The NFC Kiosk System

The NFC kiosk system (see Fig. 4) is assembled around a NFC reader [11] delivering P2P services. Upon detection of a remote initiator, it waits for an incoming CONNECT packet including a supporting service name (such as *urn:nfc:sn:tls:snep*). This event is handled by the *accept-llcp()* procedure. A client software entity is internally connected to a TLS server (running at the *127.0.0.1:443* address) based on the OPENSSL library, and works as a proxy with the NFC LLCP software entity. When the TLS session is

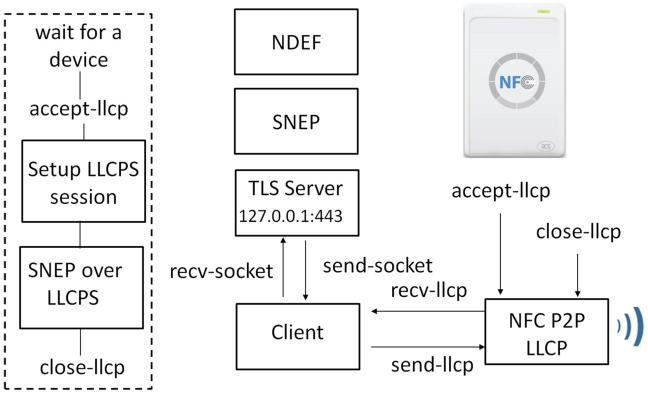

Fig. 4. Software architecture of the NFC kiosk.

established, a SNEP server, whose goal is to securely collect NDEF content, processes and produces secured SNEP messages.

5 Tests

According to the experimental platform previously introduced, a NFC SNEP client running on the Initiator mobile opens a TLS secure session with a NFC server running on the Target device, which performs an access control service. Both systems are using the OPENSSL library, and therefore support multiple cryptographic procedures. Among them we selected three security schemes:

- Key exchange with authentication dealing with RSA 1024 bits keys and X509 certificates.
- Anonymous key exchange using ECCDH, with a 163 bits elliptic curve (Sect. 163r1).
- Key exchange in the abbreviated mode; the mode use symmetric cryptography and required a previous successful full mode, either RSA or ECCDH anonymous.

In all these use cases the SNEP messages are ciphered by the RC4 algorithm and data integrity is provided by the HMAC procedure. We observe, on the target side, that for 128 bytes DEP packets, the reception requires about 85 ms, and the transmission about 65 ms; these experimental timings are higher than the expected values with a physical data rate of about 100 KBits/s.

5.1 RSA with Certificates

Both the Target and the Initiator hold X509 certificates and RSA 1024 bits private keys. Mutual authentication could be mandatory for services without native security; for example when a key value is pushed in a clear form. About 3100 bytes are exchanged during the four ways TLS handshake, which are segmented in 128 bytes DEP packets, and required around 2000 ms.

5.2 ECCDH Anonymous

In this mode TLS only provides data privacy and integrity. There is no mutual authentication, but the service could provide additional security features. About 440 bytes are exchanged during the four ways TLS handshake, which are segmented in 128 bytes DEP packets, and required around 500 ms.

5.3 The Abbreviated Mode (or Resume Mode)

According to the TLS standard a successful full mode create a shared secret (the *master secret*) that can be reused by the abbreviated mode, dealing only with symmetric cryptography. In a LLCPS context this property works like a pairing process. About 340 bytes are exchanged during this three ways TLS handshake, which are segmented in 128 bytes DEP packets, and required around 410 ms.

5.4 Secure SNEP Protocol

For all uses cases, once the TLS secure channel has been opened two protected packets (transported by the TLS record layer) are exchanged SNEP-PUT and SNEP-Success. On the Target side, about 46 TLS bytes are received in 188 ms and 26 bytes are transmitted in 32 ms.

6 Conclusion

In this chapter we presented a P2P platform secure by TLS built with commercial devices. It clearly demonstrates that a high level of security can be achieved for smart cities proximity services. However we observed performances issues with the emerging NFC technology.

References

1. Technical Specifications. http://www.nfc-forum.org/specs/
2. http://press.ihs.com/press-release/design-supply-chain/nfc-enabled-cellphone-shipments-soar-fourfold-next-five-years
3. ISO/IEC 18092, Near Field Communication - Interface and Protocol (NFCIP-1), April 2004
4. Logical Link Control Protocol, Technical Specification, NFC Forum™, LLCP 1.1, June 2011
5. Simple NDEF Exchange Protocol, Technical Specification, NFC Forum™, SNEP 1.0, August 2011
6. NFC Data Exchange Format (NDEF), Technical Specification NFC Forum™, NDEF 1.0, July 2006
7. Personal Health Device Communication, Technical Specification NFC Forum™, PHDC 1.0, February 2013

8. MF0ICU1, MIFARE Ultralight contactless single-ticket IC Rev. 3.9, NXP Semiconductors, 23 July 2014
9. EMV Payment Tokenisation Specification, Technical Framework, Version 1.0, March 2014, EMVCO
10. Lotito, A.; Mazzocchi, D.: OPEN-NPP: An Open Source Library to Enable P2P over NFC. In: 4th International Workshop on Near Field Communication (NFC) (2012)
11. PN532, User manual, Document Identifier UM0701-02, 2007, NXP Semiconductors
12. http://developer.blackberry.com/native/
13. RFC 5216, The EAP-TLS Authentication Protocol, March 2008
14. LLCPS, draft-urien-tls-llcp-00.txt, IETF Draft (2012-2014)
15. Urien, P.: A Secure Cloud of Electronic Keys for NFC Locks Securely Controlled by NFC Smartphones. In: IEEE CCNC, Las Vegas, USA (2014)
16. Urien, P.: LLCPS: A New Secure Model For Internet of Things Services Based On The NFC P2P Model. In: IEEE ISSNIP, Singapore (2014)