Chapter 15
Near Field Communication

Gerald Madlmayr, Christian Kantner and Thomas Grechenig

Abstract Near field communication (NFC) is a radio frequency (RF) based proximity coupling technology allowing transactions within a range up to 10 cm. With NFC, a key technology is on its way into the consumer's most personal device, allowing the customer to use his devices for secure services such as payment or ticketing but also for service initiation or data exchange. Interoperability is one of the most important goals to be achieved prior to the roll out of devices and services, in order to satisfy the consumer's expectations. This chapter deals with different operating modes and use cases that can be implemented with NFC technology with the main focus on mobile phones. This high level description is backed up with a look into the hardware architecture for NFC as well as the software stack in mobile phones. The chapter ends with a description of tags and tag formats for the NFC ecosystem.

15.1 Introduction

Radio frequency identification (RFID) technology is used in many daily applications. For the consumer, RFIDs are unnoticed and simple to use, they offer a popular alternative to conventional communication channels. Starting with simple access control possibilities up to complex data memories, quite different applications can be realized. A further development represents NFC [16], a technology for the fast and uncomplicated exchange of small amounts of data. It opens new perspectives

G. Madlmayr (✉) · C. Kantner · T. Grechenig
Research Group for Industrial Software (INSO), Vienna University of Technology, Vienna, Austria
e-mail: gerald.madlmayr@inso.tuwien.ac.at

C. Kantner
e-mail: christian.kantner@inso.tuwien.ac.at

T. Grechenig
e-mail: thomas.grechenig@inso.tuwien.ac.at

K. Markantonakis and K. Mayes (eds.), *Secure Smart Embedded Devices, Platforms and Applications*, DOI: 10.1007/978-1-4614-7915-4_15,

regarding the application development on all kind of consumer devices. Nowadays, NFC is being introduced to mobile phones.

NFC is an amendment to the existing contactless smart card systems, but still compatible to them. It is presented in ISO 18092 (NFCIP-1), supporting cards compliant to ISO 14443 [9] and Sony's proprietary FeLiCa system as well as NFC's own communication method. NFC allows wireless transactions over a distance of up to 10 centimetres. This is part of the *Touch and Go* philosophy giving the user a new dimension of usability. Hence, NFC-enabled handsets allow the consumer to interactively participate in the *Internet of Things* in a way like never before.

Consumers can use their handsets to retrieve further information by touching tags integrated within posters, products, or shelves. Alternatively, the handset itself can be used as a transponder, and therefore provides additional functionality in terms of applications and identification. This vision requires interoperability on different layers and a common agreement of industry players integrating technology and applications.

15.2 NFC Technology

15.2.1 Physical Layer

On the physical layer, NFC data are exchanged by two inductively coupled coils, one per appliance, generating a magnetic field with a frequency of 13.56 MHz. The field is modulated to facilitate data transfers. For communication, one device acts as the initiator (starting the communication), whereas the other device operates in target mode (waiting for the initiator). Typically, two devices are involved in the communication [3].

The roles of the devices, initiator and target, are assigned automatically during the listen-before-talk concept, which is part of the mode switching of NFC. In general, each NFC device acts in target mode. Periodically, the device switches into initiator mode in order to scan the environment for NFC targets (= polling) and then falls back into target mode. If the initiator finds a target an initiation sequence is used to establish communication before exchanging data.

NFC distinguishes two operation modes for communication: passive and active modes (Fig. 15.1).

15.2.1.1 Passive Mode

In passive mode only the device that starts the communication (the initiator) produces the 13.56 MHz carrier field. A target introduced to this field may use it to draw energy, but must not generate a carrier field of its own. The initiator transfers data by directly modulating the field, the target by load modulating it. In both directions, the

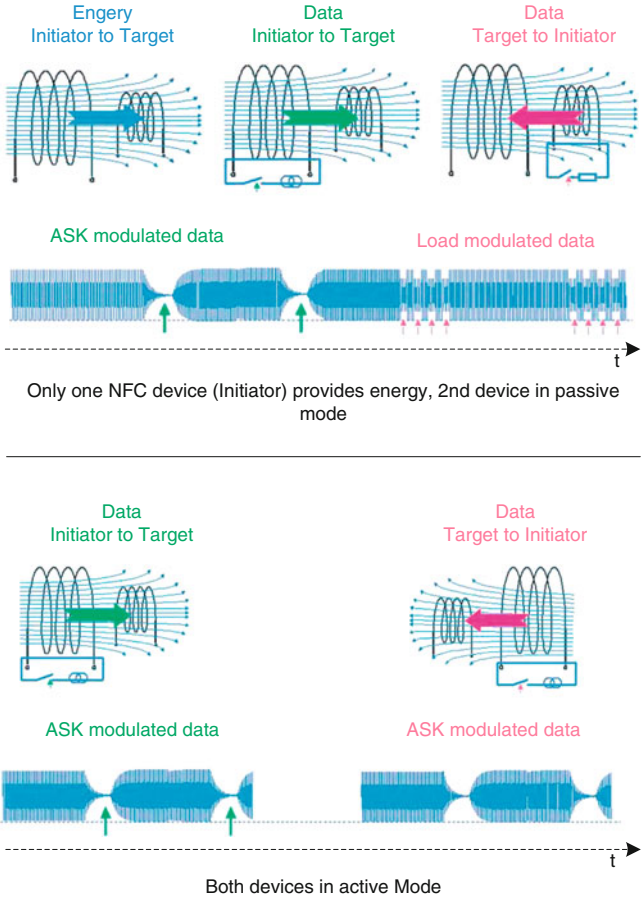

Fig. 15.1 Active and passive operation mode of an NFC device [13]

coding complies with ISO14443 or FeLiCa, respectively. This enables NFC devices to communicate with existing contactless smart cards. The term load modulation describes the influence of load changes on the initiator's carrier field's amplitude. These changes can be perceived as information by the initiator. Depending on the size of the coils, ranges up to 10 cm and data rates of 106, 212, and 424 kbits/s are possible.

15.2.1.2 Active Mode

When in active mode, both appliances generate an RF field. Each side transmits data by modifying its own field, using an amplitude shift keying (ASK) modulation scheme. Advantages compared to passive mode include a larger operating distance

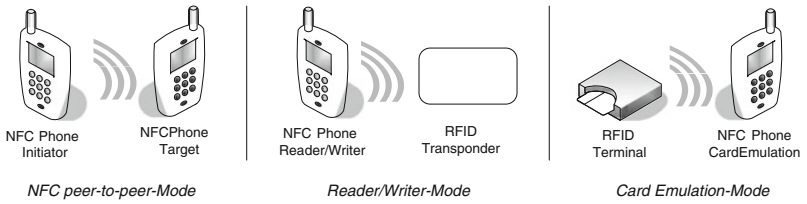

Fig. 15.2 Different operating modes of an NFC device [13]

(up to 20 cm) and higher transmission speeds (eventually over 1 MBit/s). To avoid collisions, only the sending device emits a electromagnetic field; the receiving entity switches off its field while listening. If necessary, these roles can change as often as needed [13].

15.2.2 Use Cases and Applications

An NFC compliant device offers the following modes of communication (Fig. 15.2):

15.2.2.1 Reader/Writer Mode:

In this mode, an NFC system acts as an ordinary reader for contactless smart cards. If two or more cards are present in the reader's carrier field one is selected using an anti-collision algorithm. NFC also takes care of sensing whether the chosen card is ISO 14443-A/B or FeLiCa compliant. The method used for anti-collision is dependent on the type of card detected. This mode causes the NFC device to act as an active device. From an application's view, there is no difference between a conventional and an emulated terminal, accesses to the contactless token proceed equally [3].

Operating in this mode, the NFC device can read and alter data stored in NFC compliant passive (without battery) transponders. Such tags can be found on e.g. a *SmartPoster* allowing the user to retrieve additional information by reading the tag with an NFC device. Depending on the data stored on the tag, the NFC device takes an appropriate action. If e.g. a URI was found on the tag the handset could open a Web browser.

15.2.2.2 Card Emulation Mode:

Card emulation mode is the reverse of reader/writer mode; a contactless token is emulated in passive mode. Due to the fact that the card is only emulated, it is possible to use one NFC device in place of several *real* smart cards. Which card is presented to the reader depends on the situation and can be influenced by software. Additionally,

an NFC device can contain a secure element to store the information for the emulated card in a secure way [3].

In this case, an external reader cannot distinguish between a smart card and an NFC device in card emulation mode. This mode is useful for contactless payment and ticketing applications, and a single NFC-enabled handset is capable of emulating multiple contactless smart card applications.

15.2.2.3 Peer-to-Peer Mode:

This mode is specific to NFC. After having established a link between the two participants (the method is equal to ISO 14443-A), a transparent protocol for data exchange can be started. The data block size can be chosen freely, with an maximum transmission unit (MTU) limited to 256 bytes. The main purpose of this protocol is to enable the user to send his/her own data as soon as possible (i.e. after a few milliseconds). In a peer-to-peer session, the initiator is always in active mode, whereas the target may be active or passive. This helps the target to reduce its energy consumption and is therefore especially useful if the initiator is a stationary terminal (e.g. a ticket machine) and the target a mobile device (e.g. a mobile phone) [3].

The NFC peer-to-peer mode (ISO 18092) allows two NFC-enabled devices to establish a bidirectional connection to exchange contacts, bluetooth pairing information, or any other kind of data [10]. Cumbersome pairing processes are a thing of the past thanks to NFC technology. To establish a connection, a client (NFC peer-to-peer initiator) is searching for a host (NFC peer-to-peer target) to setup a connection; then the near field communcation data exchange format (NDEF) is used to transmit the data.

15.3 Hardware Integration

In order to provide all three application modes, a mobile has to include an NFC chip, a secure element, and a host controller. Their tasks and functionality will be outlined in the following sections.

15.3.1 NFC Chip

The NFC hardware in the mobile phone has to take over various tasks. One core functionality of the NFC chip is the analogue front end which is responsible for modulating and demodulating the 13.56 MHz carrier signals from the antenna to digital signals. Additionally, the communication between the host controller as well as the secure element (if present) is part of the chip. The NFC chip is managed by the host controller and an appropriate software stack with an API which is available

to the developer. The functions of the NFC chip will most likely be an integrated part of future baseband processors in order to reduce costs and save valuable space in mobile devices.

15.3.2 Secure Element

The secure element is typically a smart card controller which is capable of emulating a *real* smart card. For security purposes, it is typically constructed in a tamper proof way. The software architecture of a typical secure element provides an environment, where applications are downloaded, personalized, managed, and removed independently with varying life cycles (e.g. using Global Platform functionality). It is also possible that the secure element is completely emulated in software. There are already general purpose CPUs in the market that support a trusted execution environment (TEE) where data can be processed in a secure way (e.g. ARM with its TrustZone architecture).

Payment or ticketing applications place high demands on security of the secure element. The chip must be highly reliable and robust to withstand different kinds of attacks. Also the manageability of the secure element is an important property [2]. Actually, a secure element function is not mandatory in an NFC device, although such devices cannot be used for card emulation mode.

- Security: Security implies confidentiality, integrity, availability, and authentication. Security as defined in ISO 14980 is: 'The purpose of information security is to ensure business continuity and minimize business damage by preventing and minimizing the impact of security incidents.'
- Manageability: A secure element can contain applications and data from different sources. Hence, means for application and data management need to be available. There should also be the possibility to remotely lock the secure element in case of loss or theft.

A secure element has to provide the following functionality [11]:

- Secure memory: A secure memory is necessary to store sensitive data like private keys, root certificates, and personal data.
- Cryptographic functions: Protocols for secure data exchange usually rely on cryptographic functions to provide security for the sensitive data, as this information must not leave the secure element without encryption.
- Secure environment for code execution: A secure element has to contain a unit to execute code in a secure way which cannot be monitored.

The secure element can be implemented in different ways in a mobile device. Smart cards or secure smart card chips are possible options for a secure element, whereas the following implementations are common for an NFC enabled handset:

- Embedded secure element as a chip

- Subscriber identity module (SIM) Card
- Secure digital card (SD-Card)

For the communication with the secure element application protocol data units (APDU) are used. The communication interface is standardized in the ISO standard for smart cards, ISO 7816. In order to link the secure element to the NFC Chip and the host controller, different interfaces are considered:

- Single wire protocol (SWP) to link a Universal Integrated Circuit Card (UICC) to the NFC chip
- SigIn-SigOut-Communication (S2C) for linking the NFC chip with embedded secure elements or SD-Cards [19]
- ISO 7816 interface for the communication between the host and the secure element.

Java Card OS is a widespread platform for the secure element which is already an industry standard for chip cards and is getting more and more popular for SIM cards as well.

15.3.3 Host Controller

The host controller has to manage the NFC chip on the one side (e.g., reading/writing data to a tag, switching between application modes) as well as to communicate with the secure element on the other side (e.g., to view the content of a card application stored on the secure element). The communication between the NFC chip and the host controller is handled by the host controller interface (HCI). The communication between the host controller and the secure element is not specifically standardized, but is based on APDUs as defined in ISO 7816-4. In case the secure element is the UICC, the communication is routed through the cell phone baseband chipset via the radio interface layer (RIL) which holds control over the UICC. In case of the embedded secure element or an SD-Card, it depends on the integration of the hardware within the phone.

Figure 15.3 shows the logical parts of each component and how they communicate with each other.

To save space, the embedded secure element and the NFC chip can be packaged into one chip. The chipset manufacturer NXP for example offers such a product under its PN65 series. This type of chip has a so-called *SmartConnect* architecture (see Fig. 15.4).

Thus the secure element, in this case NXP's SmartMX, can operate in the different modes [13]:

- Wired: In wired mode, the embedded secure element can be accessed by the host controller through the NFC chip. Therefore, an application running on the phone is able to fetch data stored on the secure element. The secure element is not visible to an external reader and therefore card emulation is disabled.

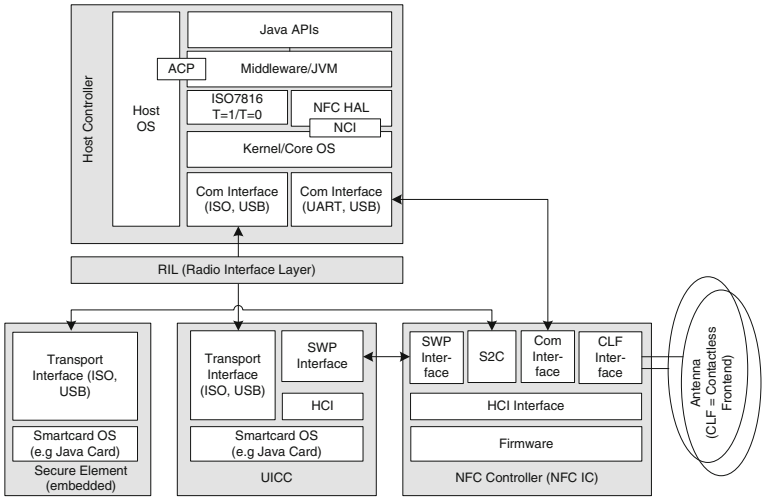

Fig. 15.3 Architecture for the integration of NFC into a mobile phone

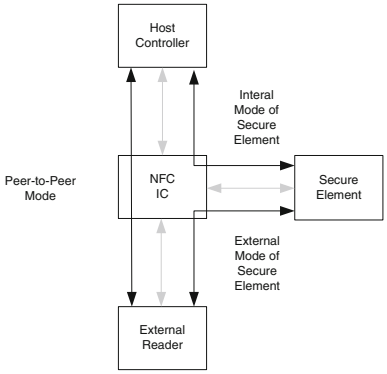

Fig. 15.4 SmartConnect architecture for NFC chips with an embedded secure element [13]

- Virtual: In virtual mode, the embedded secure element represents a virtual card and the NFC chip is in card emulation mode. The secure element cannot be accessed from an application running on the phone.
- Off: In this case the secure element is turned off, the communication with the secure element is not possible at all and the NFC chip can be used in reader/writer mode or for peer-to-peer (P2P) communication.

This is a special form of integrating both, NFC chip and an embedded secure element into a mobile phone. Handset manufactures like Nokia (3220, 6131, 6212) or Samsung (X700n, Nexus S, and Nexus Galaxy) use variants (one chip and two chip solutions) of this architecture in their phones.

15.4 NFC and Linux

The Linux kernel is used in an increasing number of computer platforms and Android is one of the most famous members of the Linux family. Besides Desktop and Server environments, Linux is used in many embedded devices such as Internet routers and connected consumer electronics. NFC has the potential to greatly enhance the user experience for such devices and so far, several initiatives have been started to integrate NFC into Linux environments:

- Native Android support: Google has integrated NXP's FRI library with a kernel driver for the PN544 chip. The whole SW architecture focuses on NXP NFC hardware and is maintained by NXP and Google. The software supports all main NFC features: reader/writer mode, secure element support for card emulation, basic SWP support, and peer-to-peer mode.
- Open NFC for Android: This is similar to the native approach, but with the focus on Inside Secure hardware. Android phone manufacturer would be responsible for the integration of Open NFC on their own products [8].
- libnfc: libnfc implements NFC functionality completely in the user space and thus depends on the existing drivers. Furthermore, the library is platform independent. It supports reader/writer mode and peer-to-peer functionality [22].

All the above-mentioned initiatives are incomplete and either miss out particular features or are focused on certain hardware. This adds justification to a new approach which is starting to implement NFC support into the linux kernel: Linux NFC Subsystem as of kernel 3.1 [14]. The NFC Linux Subsystem follows the principles of Linux open source projects:

- Vendor independent (drivers are needed for new hardware)
- Portable Operating System Interface (POSIX) compliant
- Sockets and Netlink for data exchange and device control

The first included drivers are for devices based on NXP chips PN533 (via USB) and PN544 (via I2C).

At the moment, the Linux Subsystem provides only limited functionality, but work is ongoing to eventually support all NFC features, including full support for card emulation. Currently, Nokia and OpenBossa [7] are working on this implementation goal.

15.5 NFC Integration in Android

In the following section, the software architecture for the integration of the NFC hardware architecture into an operating system will be described. This is shown for Google's Android operating system and the NXP NFC hardware platform (native Android support). To have a sound understanding on what the NFC chips exactly do, an overview of the components is given first.

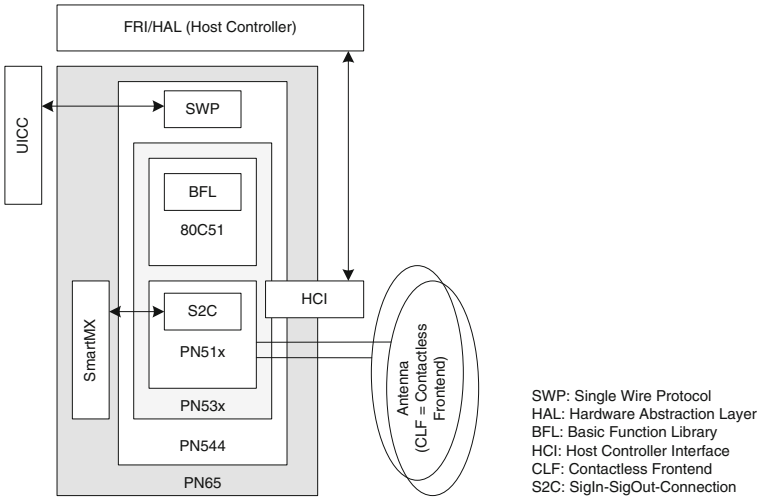

Fig. 15.5 Different chip variants of NXP's PN-family

15.5.1 NFC Chip

The Nexus Galaxy ships with a PN65 chip (see Fig. 15.5) which contains different hardware and software components, such as [20]:

- A PN512 NFC transmission module for contactless communication at 13.56 MHz.
- A micro controller (80C51 core with 32 kbyte of ROM and 1 kbyte of RAM) running the firmware for the PN512 transmission module. The combination of the micro controller and the PN512 is also called PN531.
- An additional interface and software stack to use a SIM card as the secure element. Therefore, the chip needs to implement the so-called SWP.
- A secure smart card chip. In this case, a P5CN072 Secure Dual Interface PKI Smart Card Controller, SmartMX, which can be used as the embedded secure element. This secure element is running a Java Card OS.

The chip used in this phone supports both an embedded secure element as well as a secure element implemented within the SIM card. The software running on the host is thus able to send commands to the NFC chip through the host controller interface in order to user either the embedded or SIM card secure element to emulate a virtual smart card.

From an integration point of view, it does not make any difference if the handset manufacture uses the PN544 or the PN65 as both chips have the same interfaces and use the same pin layout. The only difference is the SmartMX with is included in the PN65 which cannot be directly contacted from outside the chip.

15.5.2 API for the NFC Chip

The NFC software stack running on the host of an Android OS-based device is called the forum reference implementation (FRI) and is already part of Android since Version 2.3 (Gingerbread). The stack is implemented in pure ANSI C and communicates with the */dev/pn544* device of the Android variant of the Linux kernel. On top of the native NFC software stack, there is a Java native interface (JNI) layer that builds the bridge to the Android Java development environment for the Android developer. Finally, the android system development kit (SDK) provides Java APIs which can be used by any app running on the device in order to communicate with the NFC chip in the phone. This API can be used for reader/writer mode, P2P mode, detecting external fields, or targets as well as switching on and off the card emulation mode (Fig. 15.6).

On the J2ME platform, there is already an API standardized for this purpose: Contactless Communications API (JSR257). This JSR was released in 2006 and describes the necessary interfaces in order to allow contactless transactions with a J2ME application running on the handset. Thus, this API makes use of the read-er/writer mode as well as the NFC peer-to-peer mode. The JSR257 already implements the near field communication data exchange format (NDEF) and the basic record type definitions (RTD) published so far by the NFC-Forum [12]. Unfortunately, the Android implementation and the J2ME implementation are not compatible.

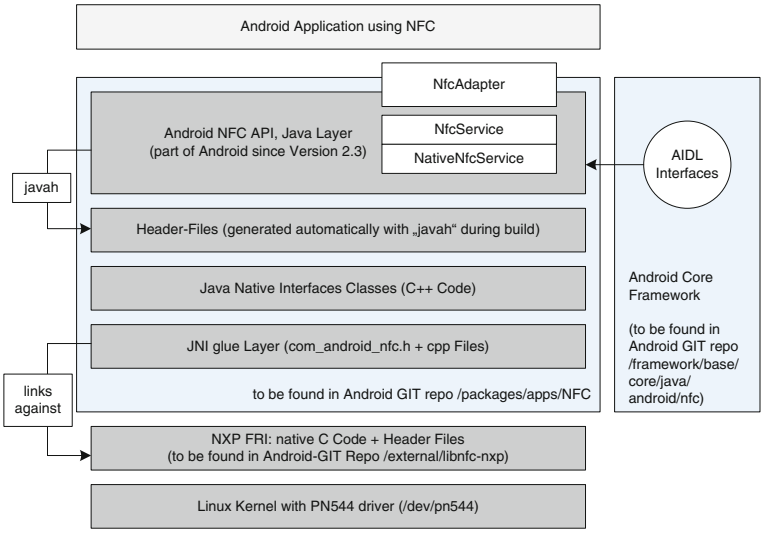

Fig. 15.6 Software stack for the integration of an NFC chip into a handset; by the example of the Google Nexus S/Nexus Galaxy

15.5.3 API for the Secure Element Access

Additionally, an API is required to access the secure elements in the phone. Accessing the embedded secure element within the PN65 can be done through the *SmartConnect* architecture and the FRI. The embedded secure element first needs to be switched into wired mode. Then a communication channel has to be established. After that APDUs can be sent to the smart card chip to read and write information from the secure memory.

Accessing the SIM card involves more software (SW) layers. The host controller of an Android OS-based device cannot directly talk to the SIM card, but must use functions from the radio baseband controller which finally connects to SIM Card. Thus, the host controller needs to send commands to the radio interface layer (RIL) which then talks to the SIM Card. As the RIL is a proprietary implementation and full control of the UICC is not mandatory for Android, it is up to the phone manufacturer to support the necessary RIL functions. The open source project secure element evaluation kit (SEEK) from G&D is for example providing tutorials and an open source stack for accessing the SIM card from an Android application. Phone manufacturers can use this module and integrate it into their Android variant. Google investigates the integration of full UICC access into the official Android code. So far the official implementation is not available.

The GSM association (GSMA) as well as the SIM Alliance agreed on an API for accessing the secure element as well as the security mechanisms using the UICC as the secure element [5, 6, 21]. SEEK already supports the Version 1.01 of the SIM Alliance's API specification as well the authentcation using PKCS#15.

The JSR177 takes over this part on the J2ME platform. The intended goal of this API was to provide the cryptographic functionality of a smart card chip to J2ME applications. The use of a secure storage for Digital Rights Management (DRM) certificates and digital signatures was also a use case during the definition. With the introduction of NFC and the use of a smart card chip for tag emulation, this API received a boost in importance. In 2007, a maintenance release was published [11] (Fig. 15.7).

The Blackberry OS also comes with NFC functionality and therefore provides an API for using contactless functionality. This new API is available since SDK 7 and allows access to the secure element (which can be either embedded or in the UICC) as well as the use of the reader/write and P2P mode.

15.5.4 Security

When looking at security of NFC different aspects are relevant. There are different threat models and attack scenarios for NFC usecases [15]. The most valuable information is stored in the secure element. Hence, this component is implemented as a separate hardware chip in the mobile device. Access to the secure element is possible

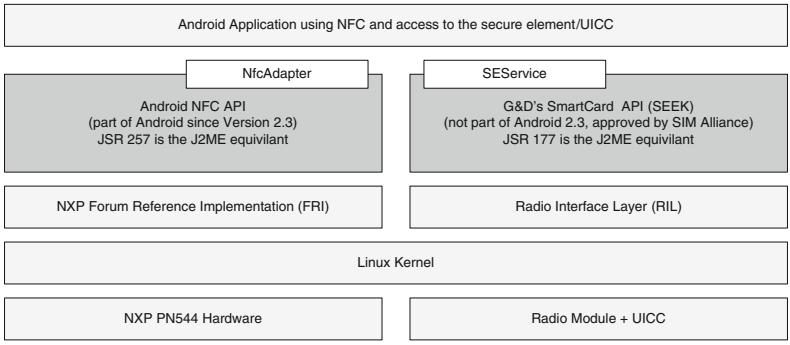

Fig. 15.7 APIs for accessing the different functionalities in an NFC-enabled handset

through the contactless interface of the NFC chip or through an application running on the host controller.

Accessing data in the secure element usually requires the appropriate keys. The most common authentication between an external reader and a secure element is a three pass mutual authentication using symmetric keys. After the authentication, a secure channel is established which allows the two parties to exchange data in a secure way. Although the data stream is routed through the NFC chip, eavesdropping information at this point is useless as the communication is encrypted.

Accessing data in the secure element from an application running on the device is the big advantage of NFC in comparison to usual smart cards. The communication is possible through SEEK (Android) or the JSR177 (J2ME). As these APIs provide access to the secure element of the device special care must be taken in order to restrict the access to those APIs.

All applications using these restricted APIs must be signed with an appropriate certified key. This mechanism is called access condition policys (ACP) enforcement. The ACP is part of the operating system and validates the signature of the application running on the host. In this case, there are certificates (PKCS#15) on either the SIM card or on the phone that are used to validate the signatures of applications that wish to access the secure element.

As the ACP is part of the operating system and therefore implemented in software it can be modified. Especially for systems which are available as open source (e.g., Android), the ACP is easy to disable . Nevertheless, it provides at least an additional barrier to accessing and hacking the UICC from malware and the attacker/customer would also have to root his device and flash a custom ROM into it.

For J2ME, there is an attack method which abuses the fact that there is no byte code verification of an application installed on the device. Thus through modifications in the byte code an application is able to access resources which normally are not available (e.g., accessing the filesystem) [4].

15.6 NFC Tags

In order to allow each NFC device to read and decode the data from NFC Tags, the NFC Forum has defined four different types of NFC Forum compliant tags as well as a data format for storing NFC relevant data structures on such a tag.

15.6.1 Tag-Types

The NFC Forum has agreed on the following four tag types.

Type 1: Type 1 Tag is based on ISO/IEC 14443A. This tag type is read and re-write capable. The memory of the tags can be write protected. Memory size can be between 96 bytes and 2 kbytes. Communication Speed with the tag is 106 kbit/s. Example: Innovision Topaz

Type 2: Type 2 Tag is based on ISO/IEC 14443A. This tag type is read and re-write capable. The memory of the tags can be write protected. Memory size can be between 48 bytes and 2 kbytes. Communication Speed with the tag is 106 kbit/s. Example: NXP Mifare Ultralight, NXP Mifare Ultralight C

Type 3: Type 3 Tag is based on the Japanese Industrial Standard (JIS) X 6319-4. This tag type is pre-configured at manufacture to be either read and re-writable, or read-only. Memory size can be up to 1 Mbyte. Communication Speed with the tag is 212 kbit/s. Example: Sony Felica

Type 4: Type 4 is fully compatible with the ISO/IEC 14443 (A&B) standard series. This tag type is pre-configured at manufacture to be either read and re-writable, or read-only. Memory size can be up to 32 kbytes; For the communication with tags APDUs according to ISO 7816-4 can be used. Communication speed with the tag is 106 kbit/s. Example: NXP DESfire, NXP SmartMX with JCOP.

The specifications for the tag types are available for free from the NFC-Forum website [1]. Note that Mifare Classic is not an NFC forum compliant tag, although reading and writing of the tag is supported by most of the NFC devices as they ship with an NXP chip. Due to its reported security weaknesses, the NXP Mifare Classic should be regarded as obsolete and not recommended for new systems [18].

15.6.2 NFC Data Exchange Format (NDEF)

The NFC forum has defined a structure for writing data to tags or exchanging it between two NFC devices. The format is called NDEF. A so-called NDEF message can contain multiple different NDEF records also referred to as record type definitions (RTD). An NDEF message has to contain at least one RTD. An RTD is an information

set for a single application, as an RTD may only contain isolated information such as text, a uniform resource indicator (URI), Multipurpose Internet Mail Extensions (MIME) media type, a business card or pairing information for other technologies. The different RTD specifications are available from the NFC Forum website. NDEF is a binary data format with a TLV (tag/length/value) structure. The maximum size of a standard NDEF record is 2^{32}-1 Bytes. As lots of NFC applications do not need so much data, the NDEF specification defines a so-called short record with a maximum length of 255 Bytes. Payloads of NDEF records can include nested NDEF messages or chains of linked chunks.

An NDEF record includes three parameters to describe its payload [17]:

- The payload length: The payload length indicates the number of bytes in the payload.
- The payload type: The NDEF payload type identifier indicates the type of the payload. NDEF supports URIs, MIME media type constructs, and an NFC-specific type format as type identifiers. By indicating the type of a payload, it is possible to hand over the payload of the records to the appropriate application on the NFC device.
- The payload identifier: The payload may contain an absolute or relative URI as the payload identifier. The use of an identifier enables payloads that support URI linking technologies to cross-reference other payloads.

The structure of an NDEF record is shown in Fig. 15.8. The header additionally includes the following parameters in the first byte:

- Message begin (MB): Indicates whether this is the first NDEF records of the NDEF message or not.
- Message end (ME): Indicates whether this is the last NDEF records of the NDEF message or not.
- Chunk flag (CF): The chunk flag bit can be set to segment the payload into multiple record with are serialized with one message.

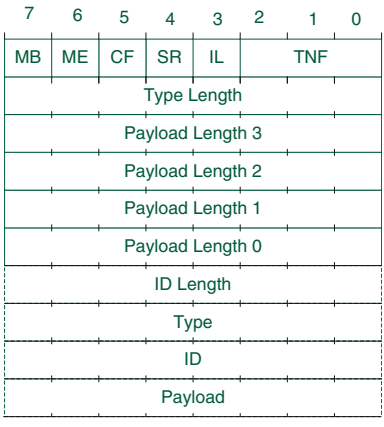

Fig. 15.8 Structure of an NDEF record

- Short record (SR): The short record bit is set to '1' in case the record size is not longer than 255 bytes.
- ID length (IL): This bit is set the record contains the payload identifier and payload identifier length.

15.7 Conclusion

NFC integrates sophisticated RFID and smart card technology into mobile devices. Although the industry has been pushing the technology through the NFC Forum since 2003, it seems to be the integration of NFC into Google's Android platform (in 2010) that has finally pushed the technology into the consumer market. Thus, NFC is on the verge of becoming a ubiquitous technology like bluetooth and WiFi.

The combination of existing contactless applications such as credit card payment and the upcoming NFC capabilities like P2P provides the basis for complete new interaction models between the virtual and physical worlds.

References

1. Nfc-forum. http://www.nfc-forum.org/
2. Bishwajit, C., Juha, R.: Mobile Device Security Element. Mobey Forum, Satamaradankatu 3 B, 3rd floor 00020 Nordea, Helsinki/Finland (2005)
3. Dillinger, O., Langer, J., Madlmayr, G., Muehlberger, A.: Near field communication in embedded systems. In: Proceedings of the Embedded World Conference 2006, vol. 01, p. 7 (2006)
4. Gowdiak, A.: Java 2 micro edition (j2me) security vulnerabilities. In: Proceedings of the Hack in the Box Security Conference (2004)
5. GSM Association: GSMA NFC UICC Requirement Specification Version 2.0. GSMA London Office, 1st Floor, Mid City Place, 71 High Holborn, London WC1V 6EA, United Kingdom, 2.0 edn. (2011). 2st Revision
6. GSM Association: NFC Handset APIs and Requirements v2.0. GSMA London Office, 1st Floor, Mid City Place, 71 High Holborn, London WC1V 6EA, United Kingdom, 2.0 edn. (2011). 2st Revision
7. openBossa Inc.: openbossa website. http://www.openbossa.org/ (2011)
8. InsideSecure: The Open NFC Project, (2011). http://www.open-nfc.org/
9. International Organization for Standardization: ISO/IEC 14443 Part 1–4: Proximity cards (2003)
10. International Organization for Standardization: ISO/IEC 18092: Near Field Communication - Interface and Protocol (NFCIP-1) (2004)
11. Java Community Process (SM) Program: Java Security and Trust Services API (SATSA). http://java.sun.com/products/satsa/ (2004). JSR177 Final Release
12. Java Community Process (SM) Program: Java Contactless Communications API. http://jcp.org/en/jsr/detail?id=257 (2006). JSR257 Final Release
13. Kunkat, H.: NFC und seine Pluspunkte. Electronic Wireless 01, 4–8 (2005)
14. Linux Kernel Organization Inc.: The Linux Kernel Archives, (2011). http://www.kernel.org/
15. Madlmayr, G., Langer, J., Schaffer, C., Scharinger, J.: Nfc devices: Security and privacy. In: S. Jakoubi, S. Tjoa, E.R. Weippl (eds.) Proceedings of the 3rd International Conference on

Availability, Reliability and Security, vol. 03, p. 6. DEXA Society, IEEE Computer Society (2008)
16. Michahelles, F., Thiesse, F., Schmidt, A., Williams, J.R.: Pervasive RFID and Near Field Communication Technology. IEEE Pervasive Computing 6(3), 94–96, c3 (2007). doi:http://doi.ieeecomputersociety.org/10.1109/MPRV.2007.64
17. NFC Forum: Nfc data exchange format (ndef). www.nfc-forum.org/resources/ (2007). Letzter Zugriff am 10.3.2008
18. Nohl, K.: Cryptanalysis of crypto-1. http://www.cs. virginia.edu/ kn5f/Mifare. Cryptanalysis.htm (2008). Letzter Zugriff am 12.12.2008
19. NXP: S2C Interface for NFC (2005). http://www.nxp.com
20. NXP: PN65 — Near Field Communication (NFC) SmartConnect Module in a single package (2006). http://www.nxp.com
21. SIMalliance Limited: Open Mobile API specification V2.02. SIMalliance Limited, 29/30 Fitzroy Square, London W1T 6LQ, 2.02 edn. (2011). 2st Revision
22. Verdult, R., Conty, R.: libnfc.org - Public platform independent Near Field Communication (NFC) library, (2011). http://www.libnfc.org/