

A Secure and Practical Key Management Mechanism for NFC Read-Write Mode

Hsu-Chen Cheng, *Wen-Wei Liao, Tian-Yow Chi, Siao-Yun Wei

Department of Information and Management, Chinese Culture University, Taipei, Taiwan

* Department of Information and Management, Chinese Culture University & Graduate Institute of Information and Computer Education, National Taiwan Normal University, Taipei, Taiwan

hccheng@sce.pccu.edu.tw, lww@staff.pccu.edu.tw, tychi@sce.pccu.edu.tw, hywei@sce.pccu.edu.tw

Abstract— Near Field Communication (NFC) is a short-range communication technology and has been applied around the world. The most common service of NFC will be micropayments. Technically, the card emulation mode of NFC will simulate mobile devices such as cell phones into stored value cards, and then debit from external readers. Apart from simulating cell phones as stored value cards, read-write mode of NFC enables the ability of devices to read and write external cards. In the context of micropayments, cell phone can be as the POS devices to read the balances from external cards and perform the actions of debiting or storing value. In this application condition, it is important to store keys of external cards securely when reading and writing. First, we discuss the security issues of key storage of NFC devices as reading and writing external card, analyze the possible risk of every solution. Last, we propose a secure and practical NFC key management mechanism and apply it on contactless mobile debit. According to the implementing results, the mechanism performs well in the efficacy and the user satisfaction.

Keywords—NFC, Key Management Mechanism, Information Security

I. INTRODUCTION

NFC is a short-range wireless communication technology developed by manufacturers like NXP, Sony, Nokia, and so on. The technology is the extension of Radio Frequency Identification (RFID). NFC chips can be embedded in consumer electronics such as cell phones, PDAs, remote controllers for being IDs of devices and enabling devices to process wireless communication within a short range. RFID tags are normally used as IDs. However, with the properties of frequency checking, hopping, and spread spectrum in a short distance, NFC devices can also exchange information and services wirelessly in a visual range. Short-range information exchanges avoid NFC transactions from attacks of sniffing, man in the middle, and so on. Therefore, NFC technology can be perfectly used in micropayments, etc.

NFC technology processes the following three modes: Card emulation, read/write, and peer to peer. Card emulation mode simulates every NFC device as a contactless card which reads or writes chips through external read-write devices. On read/write mode, NFC devices will change into contactless

read-write devices with reading and writing external passive cards or tags without batteries. As to peer to peer mode, it allows two NFC devices exchanging information within a short distance. NFC enables traditional contact/contactless chip cards to be applied in variety of aspects via the three modes mentioned above and also with the ability of calculation and internet communication (WiFi or 3G communications) of NFC devices (such as cell phones).

In recent years, there are many NFC application services experiments around the world, and the most frequently mentioned service is namely the micropayment service [1]. Technically, micropayments simulate mobile devices such as cell phones into stored value cards via card emulation mode and deduct money from stored value cell phones external card readers. Except for simulating cell phones into stored value cards, read-write mode allows devices to read or write external cards. Therefore, in the context of micropayments, cell phones can be as POS devices. It can read balances of external cards, deduct money, and stored value. As to take NFC cell phones as mobile POS devices, key management is an important issue to be discussed apart from reading and writing NDEF information in cards [2].

This study aims to investigate the security issue of key management as NFC devices read and write external cards, analyze the possible risks in various solutions. Finally, we proposed a securer and more practicable NFC key management mechanism and conduct an application of contactless mobile payments according to the proposed architecture.

II. NFC TECHNOLOGICAL ARCHITECTURE

The NFC application architecture on mobile devices is as shown in Figure 1. Most mobile devices have the setting of Java Virtual Machine; we can install and execute MIDlet of Java ME. MIDlet can communicate with service providing servers by OTA (Over the Air) via wireless communication of cell phones.

The differences between NFC and non-NFC mobile devices are NFC chipsets and the secure element (SE). The functions of three modes mentioned in section I (Card Emulation, Read/Write and Peer to Peer) are provided by NFC chipsets.

MIDlet installed in cellphones can communicate with NFC chipset via JSR257 and JSR177 [3] protocols and enable MIDlet to operation functions of NFC. The secure element is a smart card chipset, an embedded system with unit and memory computation function. Java Card Area can store Applet applications, and Mifare Area can be used for storing

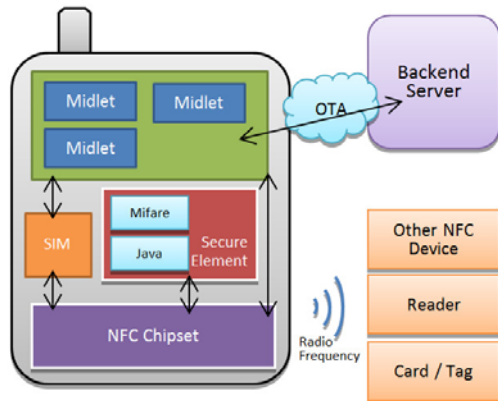


Figure 2. NFC Mobile Handsets Architecture

contents of chip cards. The secure element can be used as SAM (Security and Authentication Module) module necessary for POS operation; it allows NFC mobile POS to assure secure offline RFID access. In addition, the secure element of NFC provides remote interactive authentication and secure connection between NFC mobile POS and the back-end server. It also allows NFC mobile POS to transport transaction information in real time or batches and integrate with enterprise information portal systems. There are three types of secure element implementations: Mobile SIM cards (UICC), embedded SEs in cell phones, and independent SD/MMC cards.

If we want to read-write external cards (Mifare or FeliCa) by NFC devices on read/write mode, the key with read-write authority must be obtained firstly. Without the mechanism of standardizing data encryption and security in ISO 14443, NXP's Mifare and Sony's FeliCa have their own information encryption ways for securing the wireless communication. As the encryption of Mifare Smart Card IC S50 as the example (Figure 2), there are 16 sectors in the storage area of S50 IC, and each sector has four blocks, namely 0,1,2,3. Block 3 is loaded with Key A and Key B of 6 bytes and the access condition of 4 bytes. If external reader devices want to access the sector, they must pass the authentication of Key A or Key

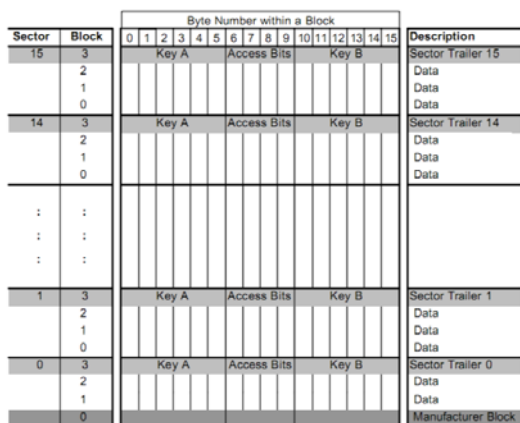


Figure 1. Mifare Smart Card IC S50 Architecture [4]

B. The access condition can decide the actions of Key A or Key B on every block of the sector, such as “can read and write”, “read only”, or “cannot read and write”.

Due to NFC mobile POS can only read and write card correctly and securely under authorizing mechanism with a key, the design of authorization, acquisition, transmission, and cacheable mechanisms in operation can affect the security of operation. In the next section, we will demonstrate and analyze some possible ways of storing keys in the context of NFC and the security threads which will be confronted in the future.

III. SECURITY ANALYSIS

A. Background

The primary NFC development focused on physical interfaces and network communication. However, with the release of related NFC services, many possible attacks on the application level had started to be mentioned [5]. The attacks include URI spoofing, NFC worms, denial-of-service, eavesdropping, data modification, transaction disputes, fake card deception, tag security crack, loss and piracy, etc.

In terms of roles in transaction process, the security threads can occur in servers, MIDlet on cell phones, secure elements, and the communication processes among the mentioned elements. J2ME is the main NFC application environment, and MIDlet might have contained some security risks such as memory monitoring, unauthorized MIDlet application replacement, JAD and RMS data reading, source code sniffing, key stroke sniffing, MIDlet server hacking, etc. In figure 4, we indicates several potential threats, the details are described as follow.

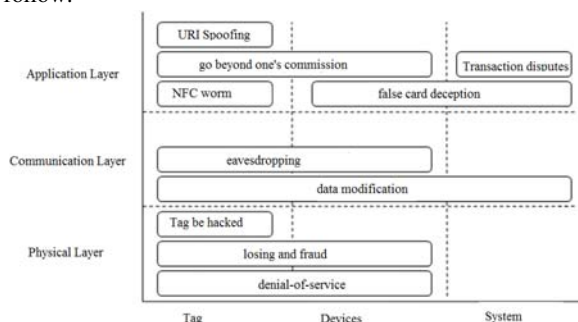


Figure 3. NFC security threat

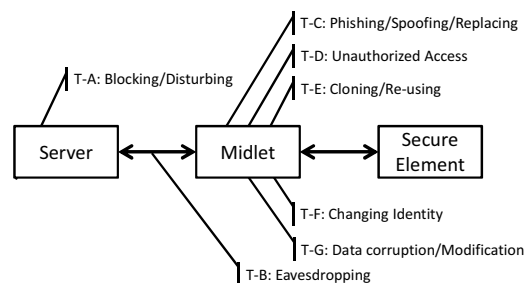


Figure 4. Threats analysis

- T-A. *Blocking/Disturbing*: servers can be attacked by denial of service (DoS) in the process of providing services. It causes the communication failure between servers and MIDlet and makes users feel panic.
- T-B. *Eavesdropping*: The communication between the devices and external servers has been eavesdropped. It causes secret data leakage.
- T-C. *Phishing/Spoofing/Replacing attacks*: MIDlet installed in NFC cell phones might be replaced illegally and the phishing menu will deceive users to transact. Another possible way is to replace the server location connected with MIDlet, users might take it for the original server and transact.
- T-D. *Fake/Unauthorized Access*: if cell phones are lost and the security strength of MIDlet on cell phones is not strong enough, the information might be fake or unauthorized accessed.
- T-E. *Cloning/Re-Using*: MIDlet on cell phones might be cloned, even be reused illegally by getting the system logic with disassemblers. There are some obfuscator tools to garble and protect the control processes of JAVA byte code programs. However, these sorts of attacks cannot be avoided in most of the cases.
- T-F. *Data corruption/modification*: The storage data might be deleted or corrupted and no longer operable. In other case, it might be modified into fake transaction information.
- T-G. *Grift/Changing identity*: The identities on cell phones such as IMEIs, certificates, chipset IDs might be modified via some illegal behaviors (e.g. accessing the engineering models of cell phones).

The risks of security threads mentioned above can be reduced by strengthening servers, MIDlet and secure elements. We systemized the authentication encryption and decryption security tools, the available identical IDs, and the storage area which can store important information for every element.

As for the security tools, Hash (MD5 or SHA), PKI, and encryption (DES, 3DES) are often used as the authentication encryption and decryption tools. The authorization of transaction can be assured by using random numbers as session IDs.

For identity, SSL connection can be used to identify the corrective servers. As for MIDlet, IMEI numbers can be used to identify cell phones. The correctness of MIDlet can be identified by code signing. For secure elements, Chip IDs can be used as the only identification.

For information storage, the information can be stored directly in the database on the server side, or the alertness data (such as keys) can be stored in a special storage area (it can be hardware or software). In MIDlet application on cell phones, the alertness data can be write in the source code and also store in Java Application Descriptor (JAD) and Record Management System (RMS). JAD and RMS store information by plain codes, hence, the encryption of information must be processed separately. On the other hand, the alertness data on the secure element side can store in the applet application directly.

	Server	Midlet	Secure Element
Algorithm	Hash	Hash	Hash
	PKI	PKI	PKI
	Encryption	Encryption	Encryption
	Random	Random	Random
Identity	SSL	IMEI	Chip ID
		Code Sign	
Storage	DB	JAD	Applet
		RMS	

Figure 5. Secure tool, identity and storage

B. Ordinary key management mechanism

According to the analysis of information security threads and the possible methods of strengthening information security mentioned above, we proposed some simple key management mechanisms for NFC devices as reading and writing external cards, and then we analyzed the possible risks of these methods. Simpler mechanisms can reduce both the difficulty of developing services and the needed time in service operation processes. On the other side, more complicated management mechanisms enhance higher security, but the developing difficulty and time are the price we must pay.

1) *Store keys in MIDlet directly*: The simplest way is to write the key on the source codes of MIDlet or JAD and RMS. Due to the plain code storage of JAD and RMS, some encryption methods can be arranged for protecting the key. The users of MIDlet can be requested to enter passwords to unlock the key, and then they can read external cards. The key management mechanism can be easily attacked by Method T-C and T-E, and then the key can be obtained.

2) *Store the key in secure elements, and then obtain the key from secure elements via MIDlet at run time*: The mechanism stores the key in secure elements beforehand and only allows the authorized MIDlet to read. The method avoids the problem of obtaining the key from the source codes by dissemblers. However, after the key in secure element read by the authorized MIDlet at run-time, information of the key will be cached in the memory. There are still possibilities for hackers to obtain the key by monitoring and scanning the memory. Hence, the risks of leakage still exist.

3) *Store the key in the server side, and then obtain the key from the server side by MIDlet at run time*: There are many security strengthening mechanisms to choose in the server side such as the decryption algorithm and firewalls, thus storing the key in the server side is seemly a good choice. However, in the process of obtaining the key by MIDlet from the server, the key must be transmitted through the public network, and the key can be possibly obtained by Method T-B. Similarly, the key still have to be cached in the memory, as MIDlet obtains the key securely at run-time. There are still possibilities for hackers to obtain the key by monitoring and scanning the memory. Besides, hackers might break down the

server by using Method T-A and cause NFC devices unable to obtain the key to finish the transaction.

4) **Store the key in the server and then store the authorized access token in secure elements. MIDlet can obtain the token from secure elements and then obtain the key from the server at run time:** As issuing the secure elements of Applet, store the personalizing token in Applet. We can apply random session IDs as the challenge and generate the pair key of PKI as the safeguard in communication at run time. In this way the security issue of T-B can be prevented. However, if hackers obtain the whole operating logic, they can easily clone a fake transaction and access sensitive data.

The four ordinary key management mechanisms mentioned above have their own weak points, especially to the security thread of T-E.

IV. NFC KEY MANAGEMENT MECHANISM

Based on the analysis and explanations mentioned above, we attempt to propose a secure and available NFC Key Management Mechanism (NKMM), a mechanism that prevents especially from the thread of T-E. We implemented the idea and the implementation detail will be given on the second part of this section.

A. Mechanism

NKMM mechanism divided into two phases: Personalizing time and runtime time. The communication procedure of the phases is shown in Figure 6. The data exchange sequence diagram of the phases is shown in Figure 7.

As to personalizing time, the main step is to install Applet and MIDlet on the NFC cell phone, and then we initialize Applet. Meanwhile, we store the personalizing sensitive data in SE of the NFC cell phone. The circuit of personalizing is as follows.

- 1) The process of personalizing must run in the situation of clean room. The server of NKMM performs the select action on Applet of SE in the NFC device in the secure environment.
- 2) Apple sends back the SE chipset identity ID (SEID) to the server for generating the index of device log.
- 3) The server generates the RSA pair $key(SnPubKey, SnPriKey)$.

- 4) The server writes $SnPubKey$ in Applet.

- 5) SEID from step 2 and $SnPriKey$ are now stored in the server key store.

After finishing the process of personalizing, every applet of SE in every NFC device has its pair key. Figure 6 shows the key obtaining circuit of NKMM at runtime. Explanations will be given separately in each step.

- 1) Users must enter the password to enable MIDlet after device is on, and then MIDlet will select Applet in the SE and perform the initial process of key obtaining at run time.
- 2) Applet will generate a challenge session ID (CID) and a set of communication PKI pair $key(CPubKey, CPriKey)$, these two pieces of information ensure the secure communication of Applet between MIDlet and the server. Before the two pieces of information are transmitted, Applet perform an encryption on $SnPubKey$ stored in Applet at personalizing time and computing its result $ENC_{SnPubKey}(CID, CPubKey)$ marked as R1 in Figure 7. CID and $CPubKey/CPriKey$ are effective only once, so they will be deleted when Applet deselecting.
- 3) Applet sends R1 and SEID to MIDlet.
- 4) MIDlet sent R1 and SEID to the server via mobile network with SSL for reducing the security issue of T-B.
- 5) The server will check whether SEID is the legal issued Applet. If it is legal issued Applet, find out the matching $SnPriKey$ according to SEID for decrypting and computing $DEC_{SnPriKey}(R1)$ to obtain CID and $CPubKey$. The computing result $ENCC_{PubKey}(CID, MK)$ from Mifare Key(MK) encryption will be marked as R2 in Figure 7.
- 6) The server sends R2 to MIDlet.
- 7) MIDlet sends the server response's information R2 into SE Applet.
- 8) SE Applet Decrypts and computes $DECC_{PriKey}(R2)$ to obtain CID and MK, and send MK back if CID matches.
- 9) MIDlet applies MK on external Mifare authentication.
- 10) MIDlet obtains Mifare access authorization and removes MK at the end.

On the circuit mentioned above, MIDlet performed response time control and prevent T-C to the server and SE. However, it cannot deal with T-E. We thus apply Digital Rights Management (DRM) of Open Mobile Alliance (OMA) [6] to encrypt MIDlet. OMA DRM protected MIDlets are super distributed from one terminal to another (via local connectivity, memory card, etc.). it makes hackers unable to perform the dissemblers and obtain the system logic by obtaining MIDlet.

MIDlet after DRM must perform OTA installation through MIDlet server and the address of MIDlet server will be recorded in JAD. Therefore, we only enable MIDlet server when we issue MIDlet at personalizing time and unable MIDlet server when we finish issuing MIDlet for reducing the risk of MIDlet server hacking.

B. Implementation

We implement our idea on Nokia 6212 Classical NFC Phone for implementing NKMM. The hardware and software that we applied are as follows: the operating system of SE of

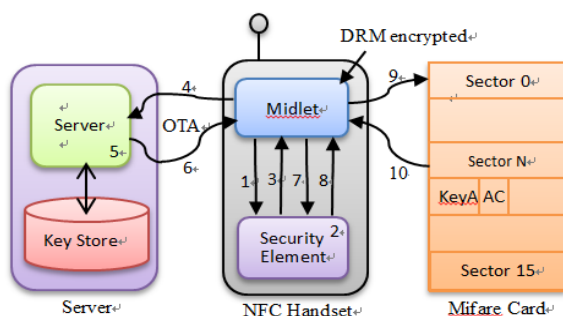


Figure 6. NKMM runtime

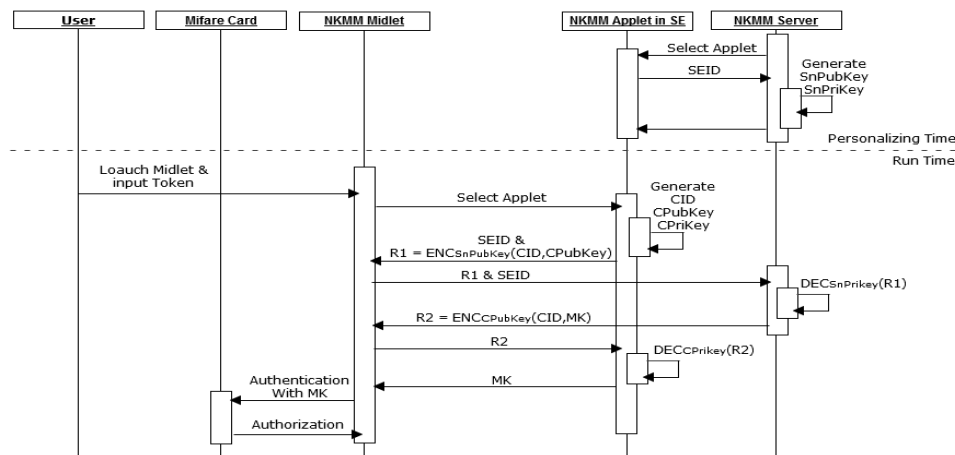


Figure 8. Sequence Diagram

Nokia 6212 is Giesecke & Devrient's (G&D) Sm@rtCafé Expert 3.1. The secure element consists of Java Card area and Mifare 4K area (behaves also as Mifare 1k) for tag emulation. The Java Card area is compliant to Global Platform 2.1.1 and compliant to Java Card 2.2.1. Applets in Java Card area can access the Mifare 4k area with G&D specific libraries (ExtSystem) provided by Sm@rtCafé Professional Toolkit. We use JCOPI Tool and Eclipse to write and install the NKMM Applet in SE. JCOPI Tool provides a set of development tools for the development, testing, and deployment of Applets for Java Card. The NFC phone Nokia 6612 must be unlocked to install Applet in SE. Once it is unlocked, the manufacturer will no longer trust SE and install any Applets in it. The Java Card embedded SE of 6212 doesn't provide API to perform external communication through NFC chipsets. Therefore, the key read from the external cannot be used for performing external communication directly after Applet computation.

We also use Nokia NFC SDK and NetBeans to implement our MIDlet. Mifare in Nokia 6212 must be code-signed to access SE. Besides, Nokia Series 40 phones (like 6212) require that your MIDlet is signed to either the operator or the manufacturer domain to get IEMI. We are not the operator or the manufacturer, so we cannot use IEMI to identify in our implementing system. However, operators or manufacturers of certain devices can use IMEI to identify them.

We performed a half-year trial run of NKMM system on the delivery service to one university provided by one Taiwanese fast food dealer, and implemented Nokia 6212 as the mobile contactless POS to conduct debit transaction on the campus cards issued by the university. The system operated well in the whole trial-run period. The Key obtaining circuit of NKMM in the implementing system would be: After the user enables the token of MIDlet, the key obtaining would be finished in about 2 seconds. We didn't explain the secure issues such as key storage to the end users. However, no users complained about the 2 second initial process. They believed

the operating is faster than the original contactless debit system of fixed POS. It proves the efficacy of our implementing system.

V. CONCLUSION

We expect the NFC key management mechanism can reach the need of the key management of NFC as accessing external tags and provide some practical implementations to the NFC manufacturers. The NKMM we proposed can cope with the architectures of current NFC cell phones and store the key securely in read-write mode. We also analyzed the possible risks and then tried to reduce the risks and trends. Finally, we performed a trial run on the Taiwanese fast food dealer and the steady and efficacy of the system can be assured.

NFC and the related devices are still on developing, we hope the analysis and system developing can provide as a material as developing NFC security. We propose some suggestions to hardware and software as follows.

As to hardware, if the Applet can send the key directly into the NFC controller without through MIDlet to authenticate the external tag, the risk of sniffing the runtime memory can be reduced.

As to software, the http request from MIDlet of J2ME to the server cannot be identified by the server and checked whether the request is sent by certain MIDlet, it causes the inability of interlocking between the server side and the MIDlet side. In the standard of J2ME, there will be a bottom layer mechanism to take the MIDlet identity out from the http head and enhance the security.

Although the Code Signing JVM implementation can check the action, do not prevent the hacker to disassemble part of the binary code. Most hacker will only be reassemble key point, if the program has been fuzzy logic binary code by the JVM to provide more information so that the program logic to identify whether the binary code by being modified, will be better able to enhance the security of NFC architectures.

REFERENCES

- [1] H. C. Cheng, J. W. Chen, T. Y. Chi, P. H. Chen. "A generic model for NFC-based mobile commerce," in *Proc. ICACT'09*, vol. 3. 20, pp. 2009–2014. 2009.
- [2] G. Madlmayr, J. Langer, C. Kantner, J. Scharinger, and I. Schaumüller-Bichl, "Risk Analysis of Over-the-Air Transactions in an NFC Ecosystem," in *Proc. 2009 First International Workshop on Near Field Communication*, pp.87-92, 2009.
- [3] C. E. Ortiz, An Introduction to Near-Field Communication and the Contactless Communication API, Sun Developer Network. [Online]. Available: <http://java.sun.com/developer/technicalArticles/javame/nfc/>, June 2008.
- [4] Standard Card IC MF1 IC S50 Datasheet, [Online], <http://www.nxp.com>.
- [5] C. Mulliner, "Vulnerability Analysis and Attacks on NFC-Enabled Mobile Phones," in *Proc. International Conference on Reliability and Security*, pp. 695-700, 2009.
- [6] Implementation Best Practices For OMA DRM v1.0 Protected MIDlets v1.0. [Online], <http://developer.nokia.com/>, May, 2004.