A Cloud-Based Secure Authentication Protocol for Contactless-NFC Payment

Nour El Madhoun*, Fouad Guenane†, Guy Pujolle*
*Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris, France
†Télécom ParisTech, 46 rue Barrault 75013 Paris, France
Email: {nour.el-madhoun, guy.pujolle}@lip6.fr; fouad.guenane@telecom-paristech.fr

***Abstract*—Nowadays, NFC technology is used in contactless payment applications by offering the NFC payment functionality in credit/debit cards, smartphones and payment terminals. Thus, an NFC payment transaction is executed in a simple and practical way. EMV is the security protocol for both contact and contactless payment systems. However, during an EMV payment transaction, this standard does not ensure two main security constraints between a customer payment device and a payment terminal: (1) mutual authentication, (2) confidentiality of sensitive banking data exchanged. These weaknesses represent a major risk in the case of NFC payment because the transaction is performed using NFC radio waves in an open environment. The risk is reduced in the case of contact payment because the transaction is executed in a closed environment by inserting the card into the terminal. In this paper, we propose a new security protocol for NFC payment transactions based on a Cloud infrastructure. We verify the correctness of this proposal using Scyther tool that provides formal proofs for security protocols.**

***Index Terms*—NFC, security, EMV, Cloud, mutual authentication, confidentiality, NFC smartphone.**

I. Introduction

Today, Near Field Communication (NFC) technology is being experimented in several countries for contactless payment operations by equipping smartphones, bank cards and payment terminals with the NFC function. An NFC purchase transaction between an NFC smartphone (or an NFC bank card) and an NFC point of sale terminal is performed instantaneously in a practical way within a short range of communication (around 10 centimetres) without any physical contact [1].

Europay Mastercard Visa (EMV) is the security standard for traditional contact payment transactions. In its specifications: (1) no mutual authentication is ensured between credit/debit cards and payment terminals but only card authentication which is well guaranteed, (2) no encryption mechanism is implemented to encrypt sensitive banking data exchanged during an EMV transaction [2]. However, EMV has been implemented in new NFC payment devices in the same manner as contact payment systems, without any particular attention given to contactless communication security. Authors in [3], [4] show that an attacker equipped with an NFC reader characterized by a special antenna, can steal confidential information stored on NFC bank cards or on NFC smartphones (NFC function is ON) that are in its (NFC reader) operating range.

This work aims to propose a Cloud-based security protocol for mobile NFC payment transactions allowing to overcome vulnerabilities that have been emerged in the EMV standard. This protocol uses asymmetric cryptography to ensure: mutual authentication (with non-repudiation) between an NFC smartphone and an NFC point of sale terminal and the integrity of private banking data. The confidentiality (encryption) of bank data is guaranteed using a symmetric session key calculated during authentication steps. We analyze the proposal by a verification tool called Scyther that provides formal proofs for security protocols.

This paper is organized as follows. Section II introduces the proposed security protocol for NFC payment whereas Section III presents Scyther verification results. The last section provides a brief conclusion and outlines future work.

II. Proposed Protocol

TABLE I
ABBREVIATIONS

Abb.	Description	Abb.	Description
P	NFC Payment terminal	*S*	NFC smartphone
C	Cloud infrastructure	*CAi*	Certification Authority (i=1..2)
H(M)	One way hashing function of M (M = m1,m2,..)	*AB*	Acquiring Bank (for *P*)
IB	Issuing Bank (for *S*)	*X*	Banking data stored on the secure element of *S*
RPi	Random number generated by *P* (i=1..2)	*RSi*	Random number generated by *S* (i=1..2)
TS	Time-stamp generated by *P*	*Cert(Y)*	Certificate of *Y* (*Y*= *P*; *S*; *AB*; *IB*)
pk(Y)	public key of *Y*	*sk(Y)*	secret key of *Y*
KMaster	Master session Key generated by *C*	*k(S,C)*	session Key allows to exchange information between *S* and *C*

A. Actors of the Protocol

Tab-I is used to simplify future descriptions. The proposed protocol includes three actors (Fig-1):

1) Cloud infrastructure (C): it is an emerging technology using Hardware and Software resources in order to provide services for NFC smartphones through a secure connection over the Internet. This platform represents a set of servers and contains databases. We assume that it: is characterized by a high level of availability in any time and location around the world, stores a list of trusted certification authorities, contains a security application allowing to confirm the authenticity of payment terminals to smartphones by proceeding to verify certificates and digital signatures (see section II-B2).

2) NFC smartphone (S): we are interested in NFC mobile payment where an NFC smartphone *S* is a connected object (Wi-Fi, 4G), offering through an NFC Android application a secure communication channel with the Cloud *C* using a new session key *k(S,C)* at each connection. To make purchases with *S*, the user needs only to approach *S* to the NFC payment terminal. Hence, the main goal of the proposed protocol is to offload the verification procedure of the NFC payment terminal authenticity, from the smartphone *S* to the Cloud *C* in order to effectively use the smartphone's resources (see section II-B2). Thus, *S* includes a cryptographic module providing a secure storage, so it stores: the private banking data *X*, *pk(S)/sk(S)*, *Cert(S)* containing *pk(S)* and signed by the issuing bank's secret key *sk(IB)*, *Cert(IB)* containing *pk(IB)* and signed by the *CA1* secret key *sk(CA1)*, an electronic signature *{H(X)}sk(IB)* of *X* generated by *sk(IB)* (after hashing *X*).

3) NFC Payment Terminal (P): it is a device used to perform NFC purchase transactions in shops for example. By default, it is connected to an information system and its acquiring bank in real time. It verifies the authenticity of smartphones locally. We assume that it has: *pk(P)/sk(P)*, *Cert(P)* containing *pk(P)* and signed by the acquiring bank's secret key *sk(AB)*, *Cert(AB)* containing *pk(AB)* and signed by the *CA2* secret key *sk(CA2)*, a list of trusted certification authorities, an application that allows verification of certificates and signatures to authenticate smartphones (see section II-B4).

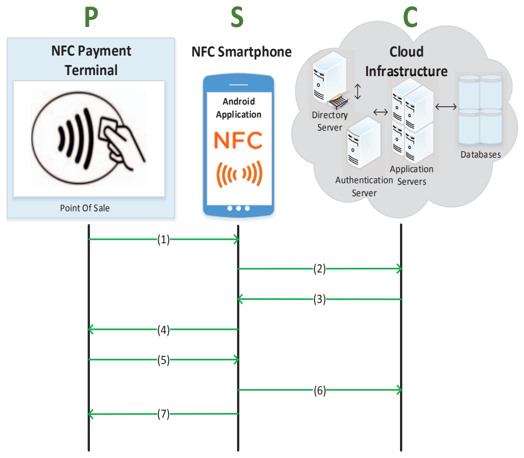


Fig. 1. The proposed NFC security protocol

B. Protocol Description

This section describes the proposed protocol by listing all the steps of exchanged messages between the actors (Fig-1):

1) S Authentication request (P -> S):

$$\textbf{\textit{SignP}} : \{H(P, S, RP1, ReqS, TS)\}\, sk(P)$$

$$P, S, RP1, TS, Cert(P), Cert(AB), ReqS, \textbf{\textit{SignP}} \quad (1)$$

In the first step, *P* sends to *S* in cleartext: a random value *RP1* and a time-stamp *TS* both used to prevent replay attacks, *Cert(P)*, *Cert(AB)*, an *S* authentication request *ReqS* and ***SignP***. The latter is a signature generated by *sk(P)* of *'P, S, RP1, ReqS, TS'* message after its hash. Therefore, *'Cert(P), Cert(AB),* ***SignP****'* allow to authenticate *P*, guarantee the integrity of *'P, S, RP1, ReqS, TS'* message and ensure that *P* cannot deny having sent ***SignP*** in the future (non-repudiation of origin). In this work, we assume that *S* does not have an application allowing to verify the authenticity of *P*, but in the next step, it will send the received message (1) to *C* to execute an authentication verification procedure.

2) P Authentication and session requests (S -> C):

$$\textbf{\textit{A}} : P, S, RP1, TS, Cert(P), Cert(AB), ReqS, \textbf{\textit{SignP}}$$

$$\{\textbf{\textit{A}}, C, RS1, ReqP, ReqSession\}\, k(S, C) \quad (2)$$

A is the message (1) received from *P* requesting *S* authentication and providing proofs authenticating *P*: *'Cert(P), Cert(AB),* ***SignP****'*. We propose that *S* will not send its authentication proofs to *P* before it confirms the authenticity of *P*. However, to confirm or reject the authenticity of *P*, the verification procedure (of *'Cert(P), Cert(AB),* ***SignP****'*) will be done in *C*. Therefore, *S* sends to *C* in an encrypted text with *k(S,C)* (session key of the current connection): ***A***, a random value *RS1* used to prevent replay attacks, an *P* authentication request *ReqP* and a session request *ReqSession*. The latter asks *C* to generate a master session key to begin a payment transaction with *P*. *RS1* acts as a time-stamp and *S* is able to distinguish between sessions by means of *RS1*.

After receiving (2), *C* deciphers the message, checks *TS* validity and if it is not valid, then it will not respond to *S*. Otherwise, *C* proceeds classically in order to authenticate *P*. It checks that: *Cert(P)* and *Cert(AB)* are valid today, the issuing *CA2* of *Cert(AB)* is a trusted certification authority, *pk(CA2)* validates the signature contained in *Cert(AB), pk(AB)* validates the signature contained in *Cert(P), pk(P)* validates ***SignP***.

3) P Authenticity and session confirmations (C -> S):

$$\textbf{\textit{B}} : C, S, P, RP1, RS1, TS, Confirm, Cert(P)$$

$$\{\textbf{\textit{B}}, KMaster\}\, k(S, C) \quad (3)$$

If *C* leads to successful verification results, then it authenticates *P* and sends to *S* in a ciphertext with *k(S,C)*: ***B*** which contains mainly a confirmation authenticity message *'Confirm'* indicating that *S* can trust *P* using *pk(P)* and a master session key *KMaster*. Otherwise, if *P* authentication has failed, *C* sends to *S* a rejection authenticity message *'RejectP'* to finish communication with *P*: *{C,S,P, RP1, RS1, RejectP}k(S,C)*.

4) Authenticity of S (S -> P):

$$\textbf{\textit{D}} : S, P, C, RS1, KMaster, Cert(S), Cert(IB)$$

$$\textbf{\textit{SignS}} : \{H(S, P, C, RS1, RP1, TS, KMaster)\}\, sk(S)$$

$$\textbf{\textit{E}} : \{S, P, RS1, RP1, TS\}\, H(KMaster, RS1, RP1)$$

$$\{\textbf{\textit{D}}, \textbf{\textit{SignS}}, \textbf{\textit{E}}\}\, pk(P) \quad (4)$$

After receiving (3), *S* checks *'RP1, RS1, TS'* and if they are not valid, then it will not respond to *P*. Otherwise, *S* confirms *P* authenticity and prepares its authentication proofs *'Cert(S), Cert(IB),* ***SignS****'*. It sends to *P* in an encrypted text with *pk(P)*: ***D***, ***SignS*** and ***E***. ***SignS*** is a signature generated by *sk(S)* of *'S, P, C, RS1, RP1, TS, KMaster'* message after its hash. *H(KMaster, RS1, RP1)* is a new session key generated by hashing *'KMaster, RS1, RP1'* and is used to start a secure payment transaction session (confidentially) between *S* and *P*. ***E*** is *'S, P, RS1, RP1, TS'* message encrypted with *H(KMaster, RS1, RP1)*.

'Cert(S), Cert(IB), ***SignS****'* allow the authentication of *S*, guarantee the integrity of *'S, P, C, RS1, RP1, TS, KMaster'* message and ensure that *S* cannot deny having sent ***SignS*** in the future (non-repudiation of origin). Therefore, *P* decrypts (4) using *sk(P)* to obtain *'RS1, KMaster, Cert(S), Cert(IB),* ***SignS****'* and proceeds classically to authenticate *S*. It checks that: *Cert(S)* and *Cert(IB)* are valid today, the issuing *CA1* of *Cert(IB)* is a trusted certification authority, *pk(CA1)* validates the signature contained in *Cert(IB)*, *pk(IB)* validates the signature contained in *Cert(S)*, *pk(S)* validates ***SignS***.

5) Confirmation to S (P -> S):

$$\{P, S, RP2\}\, H(KMaster, RS1, RP1) \qquad (5)$$

If *P* leads to successful verification results, then it authenticates *S*, decrypts ***E*** using *H(KMaster, RS1, RP1)* session key, checks *RS1, RP1, TS* and if they are not valid, then it will not respond to *S*. Otherwise, *P* starts a payment transaction session with *S* by sending a new random value *RP2* (serves to prevent replay attacks) in a ciphertext with *H(KMaster, RS1, RP1)*. *S* also can decrypt (5) and obtain *RP2*. If *S* authentication has failed, then *P* sends to *S* a rejection authentication message *'RejectS'* and finishes communication with *S*: *P,S, RejectS*.

6) Confirmation to C (S -> C):

$$\{\{S, C, P, RS2\}\, H(KMaster, RS1, RP1)\}\, k(S, C) \qquad (6)$$

In this step, *S* confirms to *C* that it will start a payment transaction session with *P* by sending in an encrypted text with *k(S,C)*: a new random value *RS2* (serves to prevent replay attacks) encrypted with *H(KMaster, RS1, RP1)*. *C* decrypts (6) using *k(S,C)* and it can also decrypt *{S, C, P, RS2}H(KMaster, RS1, RP1)* because *KMaster* is generated by itself in the step (3) and it already knows *'RS1, RP1'* from the message (2).

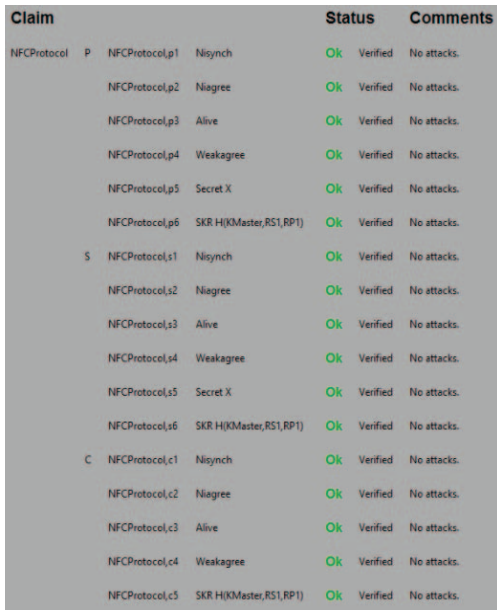
7) Exchange bank data and confirmation to P (S -> P):

$$\{S, P, RP2 - 1, RS2, X, \textbf{\textit{SignX}}\}\, H(KMaster, RS1, RP1) \qquad (7)$$

S starts with *P* the payment session securely using the session key *H(KMaster, RS1, RP1)* by sending: *RS2* (previously sent to *C*), *RP2-1*, the banking data *X* and ***SignX***. The latter is the electronic signature *{H(X)}sk(IB)* of *X* by *sk(IB)*. ***SignX*** ensures to *P* banking data integrity. Therefore, *P* decrypts (7), checks *RP2-1* and obtains *RS2*, *X* and its signature. *P* checks also ***SignX*** using *pk(IB)* obtained in step (4).

III. Scyther Verification

The correctness verification of a security protocol has proven today to be extremely difficult for humans. In this work, we have chosen to verify the proposed security protocol a tool called Scyther that has previously been successfully used and applied in both research and teaching [5]. As illustrated in Fig-2, the protocol guarantees with success all Scyther claims for *P, S, C* and no attacks are found. Authentication claims: Alive, Weakagree, Niagree and Nisynch are used to detect replay, relay and man in the middle attacks. Secret and SKR (Session Key Reveal) are confidentiality claims. Formal definitions for all Scyther claims can be found in [6] and [7].



Claim				Status		Comments
NFCProtocol	P	NFCProtocol,p1	Nisynch	Ok	Verified	No attacks.
		NFCProtocol,p2	Niagree	Ok	Verified	No attacks.
		NFCProtocol,p3	Alive	Ok	Verified	No attacks.
		NFCProtocol,p4	Weakagree	Ok	Verified	No attacks.
		NFCProtocol,p5	Secret X	Ok	Verified	No attacks.
		NFCProtocol,p6	SKR H(KMaster,RS1,RP1)	Ok	Verified	No attacks.
	S	NFCProtocol,s1	Nisynch	Ok	Verified	No attacks.
		NFCProtocol,s2	Niagree	Ok	Verified	No attacks.
		NFCProtocol,s3	Alive	Ok	Verified	No attacks.
		NFCProtocol,s4	Weakagree	Ok	Verified	No attacks.
		NFCProtocol,s5	Secret X	Ok	Verified	No attacks.
		NFCProtocol,s6	SKR H(KMaster,RS1,RP1)	Ok	Verified	No attacks.
	C	NFCProtocol,c1	Nisynch	Ok	Verified	No attacks.
		NFCProtocol,c2	Niagree	Ok	Verified	No attacks.
		NFCProtocol,c3	Alive	Ok	Verified	No attacks.
		NFCProtocol,c4	Weakagree	Ok	Verified	No attacks.
		NFCProtocol,c5	SKR H(KMaster,RS1,RP1)	Ok	Verified	No attacks.

Fig. 2. Verification results

IV. Conclusion and Future Work

In this paper, we introduced a new security protocol for NFC payment transactions allowing to solve security vulnerabilities that have been detected in the EMV protocol. We have based this solution on a Cloud infrastructure to verify payment terminals authenticity and confirm it to smartphones. It ensures: mutual authentication and non-repudiation between an NFC smartphone and an NFC payment terminal, integrity and confidentiality of private banking information. We have successfully analyzed the protocol correctness by using the Scyther verification tool that provides formal proofs for security protocols. As future work, we aim to improve this proposed solution, develop a prototype and show its effectiveness in a real environment.

References

[1] V. Coskun, B. Ozdenizci, and K. Ok, "A survey on near field communication (nfc) technology," *Wireless personal communications*, vol. 71, no. 3, pp. 2259–2294, 2013.

[2] EMV Books - Integrated Circuit Card Specifications for Payment Systems, Book 1: Application Independent ICC to Terminal Interface Requirements, Book 2: Security and Key Management, Book 3: Application Specification, Book 4: Cardholder Attendant and Acquirer Interface Requirements, Version 4.3,, November, http://www.emvco.com/, 2011.

[3] M. Emms and A. van Moorsel, "Practical attack on contactless payment cards," in *HCI2011 Workshop-Heath, Wealth and Identity Theft*, 2011.

[4] R. Lifchitz, "Hacking the nfc credit cards for fun and debit," *Hackito Ergo Sum conference*, April 2012.

[5] C. J. Cremers, "The scyther tool: Verification, falsification, and analysis of security protocols," in *Computer Aided Verification*. Springer, 2008.

[6] G. Lowe, "A hierarchy of authentication specifications," in *In Proc. 10th IEEE Computer Security Foundations Workshop*. IEEE, 1997.

[7] C. Cremers and S. Mauw, *Operational semantics and verification of security protocols*. Springer Science & Business Media, 2012.