Security Enhancements in EMV Protocol for NFC Mobile Payment

Nour El Madhoun, Guy Pujolle

Security Enhancements in EMV Protocol for NFC Mobile Payment

Nour El Madhoun, Guy Pujolle
Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris, France
Email: {nour.el-madhoun, guy.pujolle}@lip6.fr

***Abstract*—Today, by integrating Near Field Communication (NFC) technology in smartphones, bank cards and payment terminals, a purchase transaction can be executed immediately without any physical contact, without entering a PIN code or a signature. Europay Mastercard Visa (EMV) is the standard dedicated for securing contactless-NFC payment transactions. However, it does not ensure two main security proprieties: (1) the authentication of the payment terminal to the client's payment device, (2) the confidentiality of personal banking data. In this paper, we first of all detail EMV standard and its security vulnerabilities. Then, we propose a solution that enhances the EMV protocol by adding a new security layer aiming to solve EMV weaknesses. We formally check the correctness of the proposal using a security verification tool called Scyther.**

***Index Terms*—Confidentiality, EMV, mutual authentication, NFC, NFC smartphone, payment, scyther, security.**

I. Introduction

In several countries, NFC technology has been experimented for contactless payment systems by integrating it into smartphones, bank cards and payment terminals. Two types of NFC payment applications exist: (1) the micro payment which is destined for small amounts (maximum amount authorized - 20 Euro in France) without entering a PIN code or a signature, (2) the macro payment which refers to the case when the amount is higher than the authorized limit. It is noted that in the second type, the payment transaction must be confirmed by composing a PIN code or a signature.

EMV is the security protocol implemented for both payment systems [1]: with contact (inserting the card into the terminal) and without contact (NFC). In [2] and [3], authors introduce security vulnerabilities in EMV protocol and show that they represent a much higher risk in the case of NFC payment compared to the classical payment. In this paper, we propose a new protocol that improves the security of EMV standard: we enhance the classical EMV exchanged messages and we add a new security layer allowing to overcome EMV vulnerabilities.

We specify that the proposal is intended to secure NFC mobile payment transactions between NFC smartphones and payment terminals. An NFC smartphone has three wireless connection interfaces: Wi-Fi, 4G and NFC, we consider that it is interesting for the proposal to benefit from using Wi-Fi or 4G to communicate with a trusted party, because the NFC interface can be used only for communication with the payment terminal. The correctness of the proposal is formally analyzed by a security tool called Scyther which describes formal proofs for security protocols.

This paper is organized as follows. Section 2 studies EMV security standard and section 3 describes the proposed protocol solving EMV vulnerabilities. In section 4, we analyze the proposal with respect to informal analyzes whereas in section 5, we formally check the proposal using the Scyther security tool. Section 6 introduces related literature and the last section provides a brief conclusion.

II. EMV Security Standard

EMV is the international standard destined for securing contact and contactless-NFC payment transactions. It has been launched by EMV Consortium (EMVCo). The actors involved to perform an EMV payment transaction (with or without contact) are [1]: • Client's payment device: a user may possess a contact bank card allowing to perform classical payments, or an NFC bank card or an NFC smartphone for NFC purchase transactions. • Issuing bank: it is the bank of the client's payment device. • Payment terminal: it is the merchant's device allowing to make classical and NFC purchases. • Acquiring bank: it is the bank of the payment terminal.

A. EMV Phases

Three EMV security phases are executed sequentially in the same manner for a contact or NFC payment operation [4]:

1) Card authentication: authenticates the client's payment device to the payment terminal in order to provide protection against counterfeit client payment devices. It also guarantees the integrity of banking information.
2) Cardholder verification: allows the verification of the PIN code or signature entered by the client in order to authenticate him to the payment terminal. It allows protection against lost and stolen client payment devices. It is noted that this step is not executed in the case of NFC micro-payment.
3) Transaction authorization: confirms that the issuing bank authorizes the transaction to the payment terminal.

According to EMV specifications [1], each EMV phase has two different implementations "online or offline" depending on Internet connection availability in payment terminals: • Online mode: it requires an Internet connection between: (1) the payment terminal and the acquiring bank, (2) the latter and the issuing bank. Transport Layer Security (TLS) protocol (see [5]) is used to secure these connections. The connection (2) is called inter-bank network and the issuing bank is responsible for the execution of EMV security procedures (see EMV

phases II-A). For reasons of simplicity and visibility, we will not consider the inter-bank communication in this paper: we assume that the payment terminal communicates with the issuing bank directly and securely using TLS protocol. • Offline mode: the payment terminal is responsible for the execution of EMV security procedures. Once the Internet connection is available at the end of the day, the terminal confirms transactions to banks (acquiring and issuing).

B. EMV vulnerabilities

The studies [2] and [3] show two security vulnerabilities in the first phase of EMV protocol "Card authentication" in both modes (online/offline):

(a) The payment terminal is not authenticated to the client's payment device.

(b) The banking data (Primary Account Number (PAN), expiration date) are exchanged in clear without encryption.

In the online EMV card authentication phase, the revocation of banking data are well checked by the issuing bank. However, in the offline mode, the payment terminal cannot verify the validity of banking data because only the issuing bank has a list of revoked banking data [6]. We enumerate this vulnerability: (c). Fig-1 and Fig-2 respectively illustrate the online and offline EMV card authentication phases. We have designed these figures in general and clear diagrams in order to briefly and simply explain and demonstrate EMV vulnerabilities. For more details and clarifications, it is essential to consult EMV specifications in [1] and references [4] [7].

1) EMV Vulnerabilities in the online mode (Fig-1): We can notice in red color that *Banking Data* are sent in clear: vulnerability (b). To illustrate vulnerability (a), we explain the content of each exchanged message:

- (1) P->C: transaction data *TD* (date, amount, currency code, nonce, etc.) are generated by the payment terminal *P*, are unique for each transaction and serve to prevent replay attacks.
- (2) C->P: the client's payment device *C* sends to *P* an *Authorization Request* that mainly contains:
  - *TD* received in the message (1).
  - *Banking Data*: PAN and expiration date. We precise that the PAN is sent in clear because it identifies *C* in the issuing bank *IB*, but we note that this is normally a sensitive and confidential information and must therefore be protected.
  - *Information*: service code, application transaction counter, application interchange profile. It is generated by *C* and is necessary for the transaction (see [1]).
  - *ARQC (Authorization ReQuest Cryptogram)*: guarantees the authentication of *C* and the integrity of '*TD, Banking Data, Information*'. It is a cryptogram generated by *C* which applies a Hash (or MAC) function to '*TD, Banking Data, Information*' and encrypts the Hash/Mac results using *Cryptogram-Key(C,IB)*. The latter is a symmetric key with *IB*, is derived from the issuer master key + PAN, is stored securely in *C* during manufacturing by *IB* and is only known by *IB* and *C*.

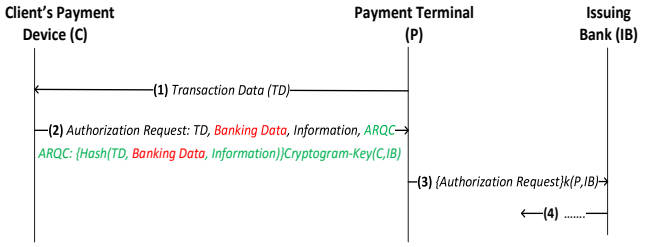


Fig. 1. EMV card authentication phase (Online)

- (3) P->IB: *P* sends the *Authorization Request* to *IB* encrypted with the key *k(P,IB)* of the current TLS session: 1. *IB* verifies that *Banking Data* are not revoked, identifies *C* in the data base with the PAN to obtain the *Cryptogram-Key(C,IB)* and calculates another cryptogram *ARQC1* by hashing '*TD, Banking Data, Information*' and encrypting hash results with *Cryptogram-Key(C,IB)*. 2. *IB* compares *ARQC* received with *ARQC1* calculated: if they are equal then it confirms (message (4)) to *P* the authenticity of *C*, non repudiation for *C* and the integrity of '*TD, Banking Data, Information*'. 3. Otherwise, *IB* rejects the transaction.

Hence, we conclude that there are no security steps provided in the online EMV card authentication phase allowing to authenticate *P* to *C*: vulnerability (a).

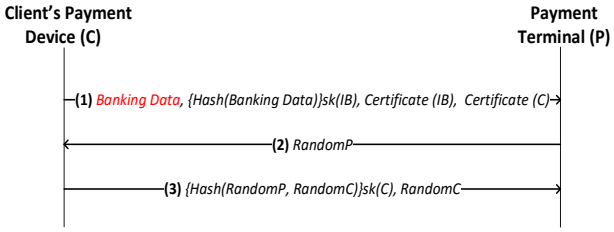


Fig. 2. EMV card authentication phase (Offline)

2) EMV Vulnerabilities in the offline mode (Fig-2): In this mode, we can also notice in red color vulnerability (b): *Banking Data* are sent in clear. In fact, there are three variants of the offline EMV card authentication phase: Static Data Authentication (SDA), Dynamic Data Authentication (DDA) and Combined Data Authentication (CDA). In this paper, we visualize the CDA variant (Fig-2) because it is more prevalent. We will now detail the content of each exchanged message to show vulnerability (a):

- (1) C->P: *C* sends to *P* in clear text: *i. Banking Data*. *ii. {Hash(Banking Data)}sk(IB)*: is the static signature of the hash of *Banking Data*. It is generated by the secret key *sk(IB)* of *IB* and guarantees the integrity of banking data. *iii. Certificate(IB)*: is the certificate of *IB* signed by the secret key of a certification authority *CA*, it will be used by *P* to obtain the public key of *IB* to verify *{Hash(Banking Data)}sk(IB)*. *iv. Certificate(C)*: is the certificate of *C* signed by *sk(IB)*, it will be used by *P* to obtain the public key of *C* to verify the dynamic signature in the last step.
- (2) P->C: *P* in the offline mode is able to execute security operations, so it verifies that: 1. *Certificate(IB)* and

Certificate(C) are during the period of validity and are not revoked. 2. the *CA* issuing *Certificate(IB)* is a trusted certification authority. 3. the public key of *CA* validates the signature of *Certificate(IB)*. 4. the public key of *IB* validates the signature of *Certificate(C)*. 5. the public key of *IB* validates the static signature *{Hash(Banking Data)}sk(IB)*. Then, it generates and sends to *C* a random number *RandomP* for dynamic signature.

- (3) C->P: after receiving (2), *C* generates a random number *RandomC*, signs using its secret key *sk(C)* the hash of *'RandomP, RandomC'* and sends the message (3) to *P*. The latter verifies that the public key of *C* validates the dynamic signature *{Hash(RandomP, RandomC)}sk(C)*. It finally confirms *C* authenticity.

Consequently, we also conclude that there are no security steps provided in the offline EMV card authentication phase confirming the authentication of *P* to *C*: vulnerability (a). Indeed, we showed that *P* did not verify the revocation of *Banking Data*: vulnerability (c). Once the Internet connection is available, *P* contacts *IB* after having confirmed and executed the payment transaction. *IB* can then check the revocation of the *Banking Data* used for the transaction.

C. Discussions

At first, EMVCo has implemented EMV standard for contact payment while assuming that EMV vulnerabilities (section II-B) do not represent significant risks: because the contact communication takes place in a closed environment by inserting the card into the terminal, and under the responsibility of the customer who must normally insert his card into a trusted terminal (for example: in a known store). Then, EMVCo has also implemented EMV protocol for contactless-NFC payment and assumed that within a maximum NFC reading distance of 5-10 centimeters, it is difficult for an attacker to use an unauthenticated NFC reader to steal banking data from an NFC bank card [1] [8].

However, the NFC communication unfolds in an open environment using NFC radio waves and this represents a big risk. Authors in [2] and [3] confirm that the assumption of EMVCo in the case of NFC payment is very weak and show that: the distance of NFC reading can reach up to 1.50 meters if the NFC reader is equipped with a special antenna and an amplifier. Therefore, a skilled attacker in radio-electronics, can remotely steal banking data from a victim's NFC bank card even if the latter is in the bag. The sensitive banking data that can be retrieved are: PAN and expiration date. The visual cryptogram (Card Verification Value (CVV)/Card Verification Code (CVC)) and the cardholder's name are unrecoverable.

D. Risks

Firstly, if a malicious person obtains a revoked bank card, then he can use it to perform unauthorized transactions in the offline mode in an airplane for example [6]. Secondly, if an attacker manages to collect only the PAN and expiration date of an NFC bank card using an unauthenticated NFC reader, several risks are incurred [2] [3]: • Making fraudulent purchase transactions on the Internet without providing the CVV/CVC and the exact name. In fact, there are several websites do not request the CVV/CVC as: "www.amazon.com", "www.zappos.com", "www.cyberguys.com", etc. In addition, the name is not always verified by websites where it is highly possible to write a random name [8]. • User identification (through PAN) and tracking.

Hence, in Fig-3 we were able, using an NFC smartphone (Samsung S5), and a free Android application to read banking data of an NFC bank card: PAN and expiration date. In the paper [8], we also proceeded in showing an example of how an attacker can harm a victim using his stolen bank data to make fraudulent purchase transactions on the Internet.

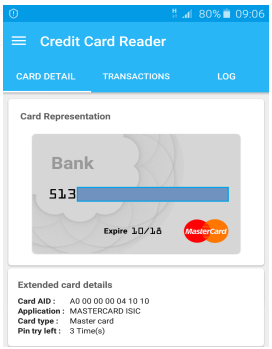


Fig. 3. Data of an NFC bank card read by an NFC smartphone

III. Proposed Protocol

In this paper, we propose a new security protocol improving EMV standard by treating EMV security weaknesses. We have designed the proposal by adding a new security layer to the classical messages of the online EMV card authentication phase (see section II-B1). This proposal includes four actors (see Fig-4) presented in section III-A. It allows securing NFC mobile payment transactions between NFC smartphones and payment terminals. It is executed in the online mode, and we also focus on the advantage of using the 4G (or Wi-Fi) connection interface of an NFC smartphone to remotely communicate with an authentication server. Tab-I presents abbreviations simplifying future descriptions.

A. Actors

1) Authentication Server (AS): it is a trusted entity able to authenticate payment terminals to NFC smartphones by performing security functions. Banks, payment terminals and NFC smartphones trust this server. We specify that *AS*: is connected to banks (acquiring/issuing) in real time, is available from any location at any time, stores a list of trusted certification authorities and contains a security application that enables verification of digital signatures and certificates.

2) NFC Smartphone (S): it includes a cryptographic unit allowing a secure storage for banking data. The client approaches *S* to *P* to make purchases (card emulation mode: simulates an NFC device as an NFC card). *S* is able to communicate securely with *AS* using Wi-Fi or 4G interface through an Android application. They authenticate each other and exchange a session key *k(S,AS)* (TLS protocol [5]). The

proposed protocol allows to offload the verification procedure of *P* authenticity from *S* to *AS* in order to effectively use *S* resources (CPU, memory, etc.)

TABLE I
ABBREVIATIONS

Abbreviation	Description
CA	Certification Authority
IB	Issuing Bank
AB	Acquiring Bank
P	Payment Terminal
S	NFC Smartphone
AS	Authentication Server
TD	Transaction Data generated by *P* (Unique)
Cert(Y)	Certificate of *Y* (*Y*=*P* or *AB* or *CA*)
pK(Y)	Public Key of *Y* (*Y*=*P* or *AB* or *CA*)
sK(Y)	Secret Key of *Y* (*Y*=*P* or *AB* or *CA*)
ReqYi	The *i* (*i*=1,2) authentication request for *Y* (*Y*=*S* or *P*)
H(M)	One way hashing function of *M* = *m1,m2..*
k(S,AS)	Symmetric key of the current TLS session allows to protect information exchanged between *S* and *AS*
k(P,IB)	Symmetric key of the current TLS session allows to protect information exchanged between *P* and *IB*
Cryptogram-Key(S,IB)	Symmetric key stored in *S* during manufacturing by *IB*

3) Payment Terminal (P): it is used to perform contact and contactless-NFC payment transactions. It communicates securely using *k(P,IB)* (TLS protocol [5]) with *IB* (online mode). We assume that *AB* delivers to *P* at the opening of the bank account (merchant): *pk(P)/sk(P)*, *Cert(P)* containing *pk(P)* and signed by *sk(AB)*, *Cert(AB)* containing *pk(AB)* and signed by *sk(CA)*.

4) Issuing Bank (IB): it is the bank of *S* and it communicates with *P* to authenticate *S* classically as in EMV card authentication phase in the online mode (see section II-B1).

B. Targeted Security Properties

The proposed protocol shall guarantee the security properties between *S* and *P*: • Mutual authentication: it is a strong agreement between *S* and *P*. *S* must authenticate *P* (overcoming vulnerability (a), section II-B1) and *P* must authenticate *S*. This agreement excludes potential replay and man-in-the-middle attacks where the attacker could usurp the identity of one of the two parties. • Non-repudiation of origin: *S* and *P* shall provide strong evidence for not being able to deny them in the future. • Confidentiality of banking data: the banking data must be sent in encrypted text (overcoming vulnerability (b)). • Integrity of banking data. • Validity of banking data.

We note that the: authenticity and non repudiation of *S*, integrity and validity of banking data are classically well ensured by the EMV standard (see section II-B1). Thus, the TLS protocol is assumed implemented in our proposal between *S* and *AS*, and between *P* and *IB*. Therefore, the mutual authentication between participants in TLS protocol is ensured by default. Also, all TLS messages are encrypted.

C. Protocol Description

We will now detail the exchanged messages of the proposal (see Fig-4). For all abbreviations you can consult Tab-I.

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

The client presents *S* to *P* thanks to NFC radio waves. *P* sends in clear text to *S*: *TD* as in the online EMV card authentication phase (section II-B1), *ReqS1*, *Cert(P)*, *Cert(AB)* and *SignP*. The latter is an electronic signature of the hash of *'P, S, TD, ReqS1'* generated by *sK(P)*. The *Cert(P)*, *Cert(AB)* and *SignP* allow to confirm the authentication of *P* to *S*, ensure the integrity of *'P, S, TD, ReqS1'* and guarantee that *P* cannot deny having sent *SignP* in the future (non-repudiation of origin). In the next step, *S* will send the received message (1) to *AS* to check *P* authenticity.

(2) *Authentication request for P: S->AS*

- We assume that *S* will not send the *Authorization Request* directly to *P* as in the online EMV card authentication phase (Fig-1), but it will first of all confirm the authenticity of *P* by contacting *AS*. Therefore, after receiving (1), the Android application automatically starts in order to connect to *AS* through a secure communication channel TLS, using the Wi-Fi or 4G interface. Then, *S* sends to *AS* in encrypted text with *k(S,AS)*: (1), a random number *RS1* serves to prevent replay attacks and *ReqP1*.
- Consequently, *AS* deciphers (2) with *k(S,AS)* and checks the validity of *TD* and *RS1*. If they are not valid, then it will not respond to *S*. Otherwise, *AS* proceeds to authenticate *P* (respond to *ReqP1*). It verifies that: 1. *Cert(P)* and *Cert(AB)* are during the period of validity. 2. *Cert(P)* and *Cert(AB)* are not revoked. 3. The issuing *CA* of *Cert(AB)* appears in the list of trusted certification authorities (see section III-A1) and then it is a trusted certification authority. 4. *pk(CA)* (obtained from *Cert(CA)*) validates the electronic signature of *Cert(AB)*. 5. *pk(AB)* (obtained from *Cert(AB)*) validates the electronic signature of *Cert(P)*. 6. *pk(P)* (obtained from *Cert(P)*) validates *SignP*.

(3) *Confirmation of P authenticity: AS->S*

AS sends to *S* in an encrypted text with *k(S,AS)*: *TD*, *RS1*, a random number *RAS1* (prevent replay attacks) and: • Either a message *ConfirmP* if it has obtained results authenticating *P* successfully. This message confirms: authenticity of *P*, non-repudiation for *P* and integrity of the message contained in *SignP*. Also, *ConfirmP* indicates to *S* that it can use *pk(P)* and trust *P*. • Or a message *RejectP* rejecting the authentication of *P*, if the verification of *P* authenticity has failed. *RejectP* tells *S* to finish the communication with *P*.

(4) *Authorization request: S->P*

After receiving (3), *S* checks results: *ConfirmP* or *RejectP*. In the latter case, *S* finishes the transaction with *P* by sending a *Rejected Transaction* message. Otherwise, it checks the validity of *TD*, *RS1* and *RAS1*. If they are:

- Not valid, then it will not respond to *P*.
- Valid, then it prepares the *Authorization Request* as in the message (2) of the online EMV card authentication phase (Fig-1). The *Authorization Request* contains

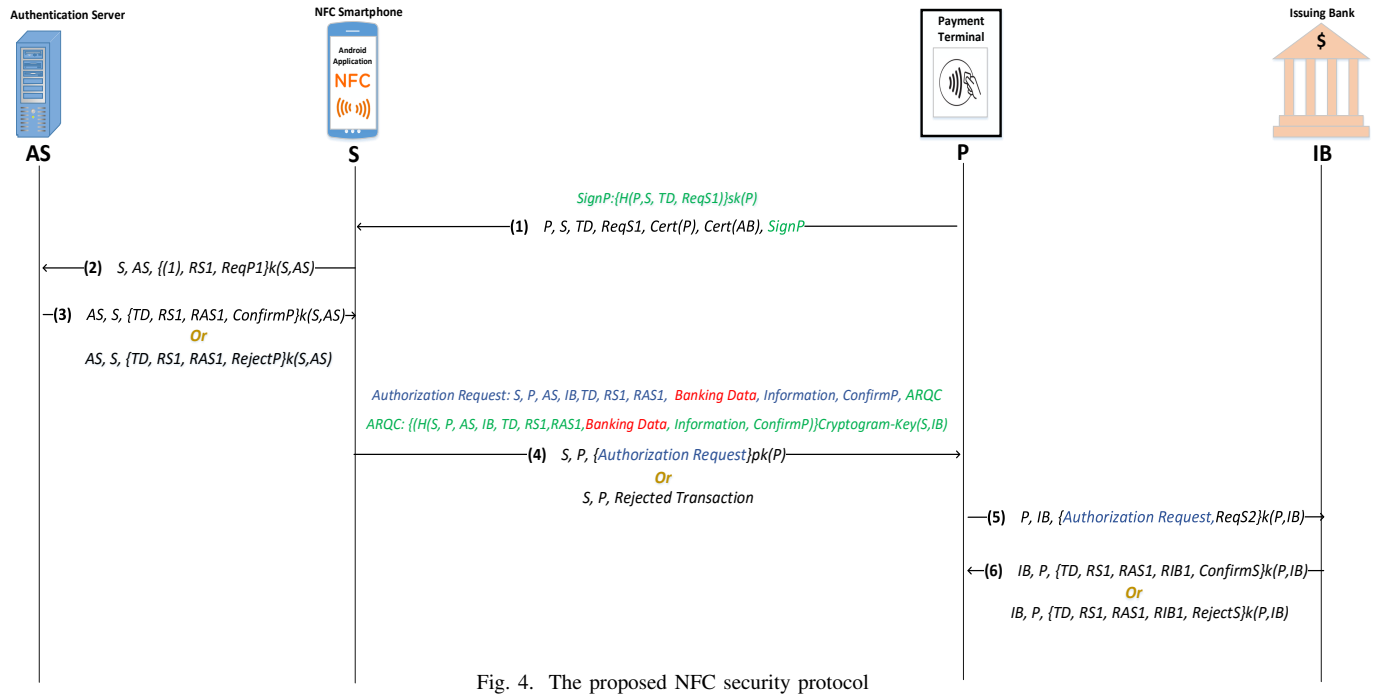


Fig. 4. The proposed NFC security protocol

(see section II-B1): '*TD*, *RS1*, *RAS1*, *Banking Data*, *Information*, *ConfirmP*, *ARQC*' and *S* sends this authorization to *P* encrypted with *pk(P)* (*Banking Data* are encrypted and not sent in clear): 1. *S* responds to *ReqS1* by generating an *ARQC* which is an electronic signature of hash results of *'S, P, AS, IB, TD, RS1, RAS1, Banking Data, Information, ConfirmP'* with *Cryptogram-Key(S,IB)*. 2. *ARQC* allows the authentication of *S* to *P*, ensures the integrity of *'S, P, AS, IB, TD, RS1, RAS1, Banking Data, Information, ConfirmP'* and guarantees that *S* cannot deny having sent *ARQC* in the future (non-repudiation of origin).

In fact, *P* cannot verify *ARQC* because it does not have *Cryptogram-Key(S,IB)* which is only known by *S* and *IB*.

(5) *Authentication and authorization requests for S: P->IB*
In this step, *P* sends to *IB* in an encrypted text with *k(P,IB)* of the current TLS session:

- The *Authorization Request* received from *S*.
- *ReqS2* whose aim is to request the authenticity of *S* once again insofar as *P* could not verify *ARQC*.

IB starts by verifying received nonces *TD*, *RS1*, *RAS1* and it will not respond to *P* if they are not valid. Alternatively:

- *IB* proceeds to authenticate *S* (respond to *ReqS2*) as in section II-B1: 1. It confirms that *Banking Data* are not revoked. 2. It uses the PAN to identify *S* in the data base and obtain the *Cryptogram-Key(S,IB)*. 3. It calculates another *ARQC1* from the received data *'S, P, AS, IB, TD, RS1, RAS1, Banking Data, Information, ConfirmP'* and using the *Cryptogram-Key(S,IB)*. Then, it compares *ARQC* with *ARQC1*.

(6) *Confirmation of S authenticity and authorization: IB->P*
IB sends to *P* in an encrypted text with *k(P,IB)*: *TD*, *RS1*, *RAS1*, a random number *RIB1* allows preventing replay attacks and: • Either a message *ConfirmS* if *ARQC* and *ARQC1* are equal. This confirms: authenticity and authorization of *S*, non-repudiation for *S*, integrity of the message contained in *ARQC* and especially for *Banking Data*. • Or a message *RejectS* if *ARQC* and *ARQC1* are not equal. *RejectS* indicates to *P* to finish the payment transaction with *S*.

IV. Informal Analyzes

A. Results

The proposal in this work meets the security properties discussed in section III-B as follows: •Mutual authentication between *S* and *P*: in step (3), *S* confirms the authenticity of *P* thanks to *Cert(P)*, *Cert(AB)* and *SignP* (overcoming vulnerability (a)). In step (6), *P* confirms the authenticity of *S* through *ARQC*. • Non-repudiation of origin: *P* and *S* cannot deny having respectively sent *SignP* and *ARQC* in the future. • Confidentiality of *Banking Data* using *pK(P)* in step (4) (overcoming vulnerability (b)). • Integrity of *Banking Data* stored on *S* thanks to the cryptogram *ARQC* in step (6). • Validity of *Banking Data* that are not revoked in step (5).

B. Examination of Possible Attacks

We will informally examine some attacks between *P* and *S*. The analyze of attacks on TLS protocol [5] between: *'S* and *AS'*, *'P* and *IB'*, is not the objective of this paper.

1) Malicious Payment Terminal (MP): which wants to pose as *P* to communicate with *S*. We assume that *MP* does not have a valid certificate and is not known by *AS*. Possible attacks:

(a) In step (1), *MP* sends to *S* principally: *TDmp* generated by itself, *Cert(P)*, *Cert(AB)*, and a signature *SignMP* generated by *MP* key. In step (3), *AS* rejects *P* authenticity because it could not verify *SignMP* with *pK(P)*.
(b) If *MP* replays the message (1), then this is can be detected thanks to *TD* validity.
(c) *MP* cannot decipher the message (4) to obtain *Banking Data* because it does not have *sk(P)*.

2) Malicious Smartphone (MS): which wants to pose as *S* to communicate with *P*. We assume that *MS* is not known by *AS* and *IB*. Possible attacks:

(a) In step (4)-(5), *MS* sends *P* an *Authorization RequestMS* encrypted by *pk(P)* and mainly contains: *TD*, random numbers *RMS1* and *RASms1* instead of *RS1* and *RAS1* respectively, falsified *Banking DataMS*, an *ARQCms* generated by a specific key for *MS* instead of *ARQC*. In step (5), *P* sends to *IB* the *Authorization RequestMS* received from *MS* and without knowing that it is a malicious. However, *IB* rejects *S* in step (6) because: there is no good PAN to identify *S* and it could not verify *ARQCms*.
(b) If *MS* replays the message (4), then this is can be detected by the validity of *TD*, *RS1* and *RAS1*.
(c) *MS* cannot decipher the message (4) to obtain *Banking Data* because it does not have *sk(P)*.

V. Formal Analyzes Using Scyther Tool

A. Overview

The verification of the correctness and soundness of a security protocol has proven to this day to be extremely difficult for humans. Hence, we verify the proposed protocol using a security verification tool called Scyther that has previously been successfully used in both research and teaching fields [9]. Scyther allows formal analyzes for security protocols by identifying potential attacks and vulnerabilities. Researchers in [10] show the performance of Scyther compared to other security verification tools. Scyther analyzes protocols using specific Scyther claims (authentication, confidentiality, etc.) with an unbounded number of sessions and guaranteed termination. If it detects an attack corresponding to a mentioned claim, then it produces a graph describing this attack [9].

B. Scyther Claims Results

The Security Protocol Description Language (SPDL) is used to implement protocols in the Scyther tool [11]. Each actor is written as a role in this language. Due to the page limit, we show only the role for *P* in Fig-5. The Scyther claims mentioned in the implementation of our proposal are: (1) *Nisynch, Niagree, Alive, Weakagree* for the mutual authentication and non-repudiation between *S* and *P*, (2) *Secret* for the confidentiality of *Banking Data*. Therefore, they refer to the targeted security properties presented in section III-B. As illustrated in Fig-6, the protocol successfully guarantees all Scyther claims for *P* and *S* and no attacks are found. Indeed, there are no Scyther claims to verify the integrity and validity of *Banking Data*, but we can conclude that they are ensured thanks to the mutual authentication. We provide formal definitions taken from references [11] and [12] for *Nisynch, Niagree, Alive* and *Weakagree* Scyther claims:

```
role P
{
 fresh TD: Nonce;
 var BankData,ConfirmP, ConfirmS, Information: text;
 fresh ReqS1,ReqS2: text;
 var RS1,RAS1,RIB1: Nonce;
 send_1(P,S,P,S, TD, ReqS1, Cert(A),Cert(AB), {H(P,S, TD, ReqS1)}sk(P));
 recv_4(S,P, S,P, {S, P, AS,IB,TD, RS1, RAS1,  BankData, Information, ConfirmP,
 {H(S,P,AS,IB, TD, RS1,RAS1,BankData, Information, ConfirmP)}Cryptogram Key(S,IB)}pk(P));
 send_5(P,IB,P,IB,{S, P, AS,IB,TD, RS1, RAS1,  BankData, Information, ConfirmP,
 {H(S,P,AS,IB, TD, RS1,RAS1,BankData, Information, ConfirmP)}k(S,IB), ReqS2}k(P,IB));
 recv_6(IB,P,IB,P, {TD, RS1,RAS1, RIB1, ConfirmS}k(P,IB));
 claim_p1(P, Nisynch);  claim_p2(P, Niagree);  claim_p3(P,Alive); claim_p4(P,Weakagree);
 claim_p5(P,Secret,BankData);
}
```

Fig. 5. Role *P* in SPDL Language (Scyther)

1) Non-injective synchronization (Nisynch): "ensures that messages are transmitted exactly as prescribed by the protocol. That is to say that whenever A (initiator) completes running the protocol with B (responder), and B has been running the protocol with A, then, all messages are received exactly as they were sent, in the exact order described by the protocol". This authentication form is strictly a stronger property than the other forms: *Niagree, Alive* and *Weakagree*.

2) Non-injective agreement (Niagree): "We say that a protocol guarantees to an initiator A non-injective agreement with a responder B on a set of data items ds (where ds is a set of free variables appearing in the protocol description) if, whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A, and B was acting as responder in his run, and the two agents agreed on the data values corresponding to all the variables in ds".

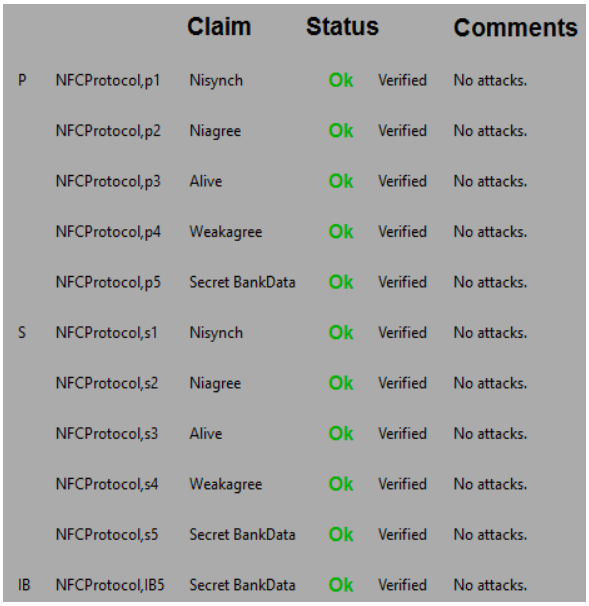
3) Aliveness (Alive): "We say that a protocol guarantees to an initiator A aliveness of an agent B if, whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol".

4) Weak agreement (Weakagree): "We say that a protocol guarantees to an initiator A weak agreement with another agent B if, whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A. Note that B may not necessarily have been acting as responder".

VI. Related Literature

A mutual authentication protocol between NFC smartphones and payment terminals enabling to secure NFC payment transactions is proposed in [13]. The idea of this work is to use a single server that shares, in a static manner, secret symmetric keys with terminals and NFC smartphones. This idea remains difficult to design in a real and global environment. Hence, the proposal aims to resolve EMV security vulnerabilities: it guarantees mutual authentication and banking data confidentiality, but it does not ensure: banking data integrity (which is normally well assured by the original

EMV protocol), origin non-repudiation and the validity of banking data that are not revoked. In fact, authors did not take into consideration the advantage of an NFC smartphone having both Wi-Fi and 4G interfaces that may be useful for direct communication with the server: the NFC smartphone connects firstly using NFC radio waves to the payment terminal, the latter then communicates with the server.



		Claim	Status		Comments
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 BankData	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 BankData	Ok	Verified	No attacks.
IB	NFCProtocol,IB5	Secret BankData	Ok	Verified	No attacks.

Fig. 6. Formal results with Scyther

In our previous studies [8] and [14], we have designed security protocols for NFC payment systems based on asymmetric key cryptography and allowing to overcome EMV security weaknesses in EMV card authentication phase. They ensure: mutual authentication, non-repudiation, confidentiality and integrity of banking data. Thus, we have successfully analyzed the correctness of these proposals using Scyther.

The proposal [8] is destined to secure NFC payment operations between payment terminals and unconnected (without Wi-Fi or 4G) client payment devices: NFC bank cards. It is implemented in the online mode where the payment terminal communicates with an authentication server which is considered a representative of confidence and security of banks (issuing and acquiring). Thus, NFC bank cards trust the server and can communicate with it only via the payment terminal. The server is able to confirm the authenticity of NFC bank cards and payment terminals. This proposal ensures that banking data are not revoked.

The proposal [14] is intended to secure NFC mobile payment transactions between NFC smartphones and payment terminals. It is executed in the offline mode where the payment terminal is not connected to the issuing bank and is responsible for the execution of the security procedure authenticating an NFC smartphone. The latter can interact (using Wi-Fi or 4G) through a secure channel (TLS) with a Cloud platform that offers security services such as: verifying the payment terminal authenticity. This proposal allows an efficient use of smartphone's resources (CPU, memory, etc.), because it enables offloading *"the verification procedure of the payment terminal authenticity"* from the NFC smartphone to the Cloud platform. However, it does not verify the revoking of banking data because it is executed in the offline mode.

The research work [8], [13] and [14], are different from the classical EMV card authentication phase (online or offline): because they implement new security architectures by using new security elements for each actor as: multiple electronics certificates and signatures in [8] and [14], symmetric keys in [13], etc. However, these large differences are too expensive although they theoretically participate in solving EMV vulnerabilities. We can say that these protocols require new implementations in a real environment and this can contribute to remove EMV standard. Our proposal in this paper aims to improve the security of EMV standard while maintaining the same basis of EMV messages and adding a new security layer.

VII. Conclusion

In this paper, we introduced a new protocol destined to secure NFC mobile payment transactions between NFC smartphones and payment terminals. It allows to solve EMV security weaknesses by enhancing the classical EMV exchanged messages and adding a new security layer. It ensures: mutual authentication and non-repudiation (compared to [13]), integrity (compared to [13]) and confidentiality of banking information, the validity of banking data that are not revoked (compared to [13] and [14]). We have successfully analyzed the protocol correctness using the Scyther tool.

References

[1] 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, V. 4.3, EMVCo, http://www.emvco.com/, Nov. 2011.

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

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

[4] S. J. Murdoch, S. Drimer, R. Anderson, and M. Bond, "Chip and pin is broken," *IEEE Symposium on Security and Privacy*, pp. 433–446, 2010.

[5] T. Dierks, "The transport layer security (tls) protocol version 1.2," 2008.

[6] M. Levi, P. Bissell, T. Richardson, and C. P. Unit, "The prevention of cheque and credit card fraud," *Citeseer*, 1991.

[7] S. Bouzefrane, "La norme emv," 2009. [Online]. Available: http://cedric.cnam.fr/~bouzefra/cours/Cartes_Bouzefrane_EMV_nov2009.pdf

[8] N. El Madhoun, F. Guenane, and G. Pujolle, "An online security protocol for nfc payment: Formally analyzed by the scyther tool," *International Conference on Mobile and Secure Services (MobiSec)*, pp. 1–7, 2016.

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

[10] C. Cremers and P. Lafourcade, "Comparing state spaces in automatic protocol verification," *International Workshop on Automated Verification of Critical Systems (AVoCS)*, 2007.

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

[12] G. Lowe, "A hierarchy of authentication specifications," *IEEE 10th Computer Security Foundations Workshop*, pp. 31–43, 1997.

[13] U. B. Ceipidor, C. M. Medaglia, A. Marino, S. Sposato, and A. Moroni, "Kernees: A protocol for mutual authentication between nfc phones and pos terminals for secure payment transactions," *IEEE International ISC Conference on Information Security and Cryptology (ISCISC)*, 2012.

[14] N. El Madhoun, F. Guenane, and G. Pujolle, "A cloud-based secure authentication protocol for contactless-nfc payment," *IEEE 4th International Conference on Cloud Networking (CloudNet)*, pp. 328–330, 2015.