A Secure Cloud-Based NFC Payment Architecture for Small Traders

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*—Nowadays, Near Field Communication (NFC) technology is being experimented for payment systems to perform purchase transactions without physical contact, without entering a PIN code or a signature. An NFC purchase transaction is mainly executed between a client's payment device (NFC bank card or NFC smartphone) and a merchant's payment device. The latter may be either an NFC point of sale machine for big traders (in a department store for example), or a simple NFC payment terminal attached to the outside of a smartphone for small traders. In this paper, we propose a new secure cloud-based NFC payment architecture for small traders: which allows them to benefit from their smartphones integrating NFC technology for use directly as NFC merchant payment devices, without needing to buy an external NFC payment terminal. In addition, the proposal introduces a new protocol aiming to secure NFC payment transactions.**

***Index Terms*—Cloud, EMV, merchant, NFC, NFC smartphone, NFC security, small trader.**

I. Introduction

NFC is a short-range (maximum 10 centimeters) wireless technology operating at 13.56 MHz and allowing communication between devices with no further user intervention [1]. It has been experimented in several countries for contactless payment systems: VISA payWave [2] and MasterCard PayPass [3]. Europay Mastercard Visa (EMV) is the standard implemented to secure NFC payment transactions. Four actors (see Fig-1) participate together to execute a secure NFC payment operation according to EMV specifications [4]:

(1) Client's payment device: it securely stores the banking data (Primary Account Number (PAN), expiration date, cardholder's name, visual cryptogram, etc.). It can either be an NFC bank card or an NFC smartphone.

(2) Merchant's payment device: it is the trader's device allowing to accept NFC purchases. It may appear in two different cases:
- For big traders (department supermarkets, shops, etc.): it uses a professional (big) NFC point of sale machine. The latter is considered as a trusted element because, by default, it has hardware and software security capabilities.
- For small traders (small store, private taxi, home schooling, etc.): it includes two elements, the first is a simple NFC payment terminal connected to the outside of the second element which is the trader's smartphone.

(3) Acquiring bank: it is the bank of the merchant's payment device.

(4) Issuing bank: it is the bank of the client's payment device.

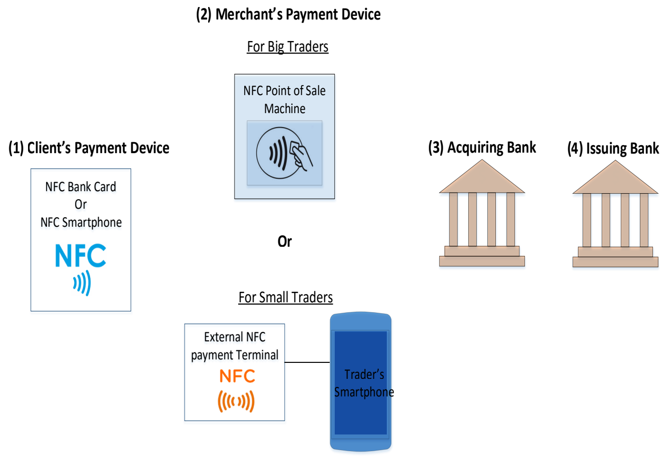

(2) Merchant's Payment Device
For Big Traders
NFC Point of Sale Machine
(1) Client's Payment Device
NFC Bank Card Or NFC Smartphone
NFC
(3) Acquiring Bank
(4) Issuing Bank
Or
For Small Traders
External NFC payment Terminal
NFC
Trader's Smartphone

Fig. 1. Participants involved in an NFC payment transaction

In our previous work [5], we studied EMV security protocol. The latter is implemented with two modes "online and offline" depending on Internet connection availability in merchant payment devices to communicate with the banking network (acquiring and issuing banks). In this paper, we consider only the online mode. So, during the execution of an NFC payment transaction [5] [6] [7]:

- EMV protocol ensures the following security properties:
  - Authentication of the client's payment device to the merchant's payment device.
  - Integrity of banking data exchanged.
  - Validity of banking data that are not revoked.
- Two security vulnerabilities have been detected in EMV protocol:
  - Vulnerability (1): the authentication of the merchant's payment device is not ensured to the client's payment device.
  - Vulnerability (2): the confidentiality of banking data between the client's payment device and the merchant's payment device is not ensured: the PAN and expiration date are exchanged in clear without encryption. The visual cryptogram and the cardholder's name are encrypted.

In fact, EMV standard has been launched by EMV Consortium (EMVCo) for securing NFC payment transactions by assuming that: at a reading distance that does not exceed 10 centimeters with NFC waves, an attacker is not able to use an unauthenticated NFC reader to retrieve banking data from client payment devices [8]. However, the assumption of EMVCo is very weak and the studies [5] [6] [7] demonstrate that if an attacker uses an NFC antenna and an amplifier attached to a simple NFC reader, the distance of NFC reading can reach up to 1.50 meters. The attacker can then remotely steal banking data from client payment devices.

In addition, the banking data of several clients can be retrieved if an attacker listens or a thief steals a merchant's payment device. The latter, in the case of small traders, is less secure than the merchant's payment device in the case of big traders because the smartphone of the small trader [9] [10]:

- Does not have professional hardware and software security capabilities.
- Is not considered as a good trusted element and can be hacked easily.
- Receives from the external NFC payment terminal the banking data without encryption in the Android system, and then they can be retrieved by Android's attackers.

Consequently, EMV vulnerabilities create several risks [5]:

- The attacker can use the retrieved banking data to make fraudulent payments on the Internet without needing to provide the visual cryptogram and the exact cardholder's name: many websites exist today as "www.zappos.com", "www.amazon.com", "www.cyberguys.com" do not request the visual cryptogram and do not check the cardholder's name.
- The PAN is a sensitive information where the attacker can use it to identify and track the client.

Otherwise, in the case of big traders, the studies [5] [8] [11] and [12] have proposed new solutions and protocols aiming to secure NFC payment transactions by solving EMV vulnerabilities. In the case of small traders, in this work we propose to remedy EMV vulnerabilities by designing a new NFC payment architecture based on a cloud infrastructure illustrated in the next section.

II. Proposed Architecture

We propose a secure cloud-based NFC payment architecture for small traders:

- Allowing them to use with confidence their smartphones integrating NFC technology as NFC merchant payment devices.
- Introducing a security protocol that adequately solves EMV weaknesses during the execution of NFC payment transactions.

Therefore, to execute an NFC payment transaction, the NFC smartphone of the small trader communicates thanks to NFC radio waves with the client's payment device: NFC bank card or NFC smartphone. The proposed architecture aims to:

- Minimize the cost: no need to buy an external NFC payment terminal to attach it to the outside of the trader's smartphone.
- Benefit from the NFC interface that exists in the trader's smartphone.
- Simplify and reduce space requirements by eliminating the external component.
- Place the trader's NFC smartphone as a trusted and secure element for clients.
- Simplify and secure the procedure of NFC payment for small traders by solving EMV weaknesses.

TABLE I
Abbreviations

Abbreviation	**Description**
C	Client's Payment Device (NFC bank card or NFC smartphone)
T	NFC Smartphone of the Small Trader
CI	Cloud Infrastructure
Cert(X)	Certificate of *X (X=CI or C)*
TD	Transaction Data generated by *T* (Unique)
H(M)	One way Hashing function of *M = m1,m2..*
ReqXi	*The i (i=1,2)* Authentication Request for *X (X=C or T)*
RandX	Random Number generated by *X (X=CI or C)* (Unique)
pk(X)	Public Key of *X (X=CI or C)*
sk(X)	Secret Key of *X (X=CI or C)*
k(T,CI)	Symmetric key of the current TLS session allows to protect information exchanged between *T* and *CI*

A. Actors

The proposed architecture for small traders includes three actors illustrated in Fig-2. For all abbreviations you can consult Tab-I.

(1) Cloud Infrastructure (*CI*): it is an emerging technology using Hardware and Software resources in order to provide services for the other actors. It represents a set of servers and contains databases. This platform is a trusted entity able to execute security functions. We assume that the banking network (issuing and acquiring) is securely connected in real time to *CI* and trusts it. *CI* is the intermediary between the NFC smartphone of the small trader and the banking network. It owns: *pk(CI)/sk(CI)*, *Cert(CI)* signed by a certification authority. *CI* is characterized by a high level of availability in any time and location around the world.

(2) Trader's NFC Smartphone (*T*): we benefit from the NFC interface of *T* to accept NFC payment transactions from

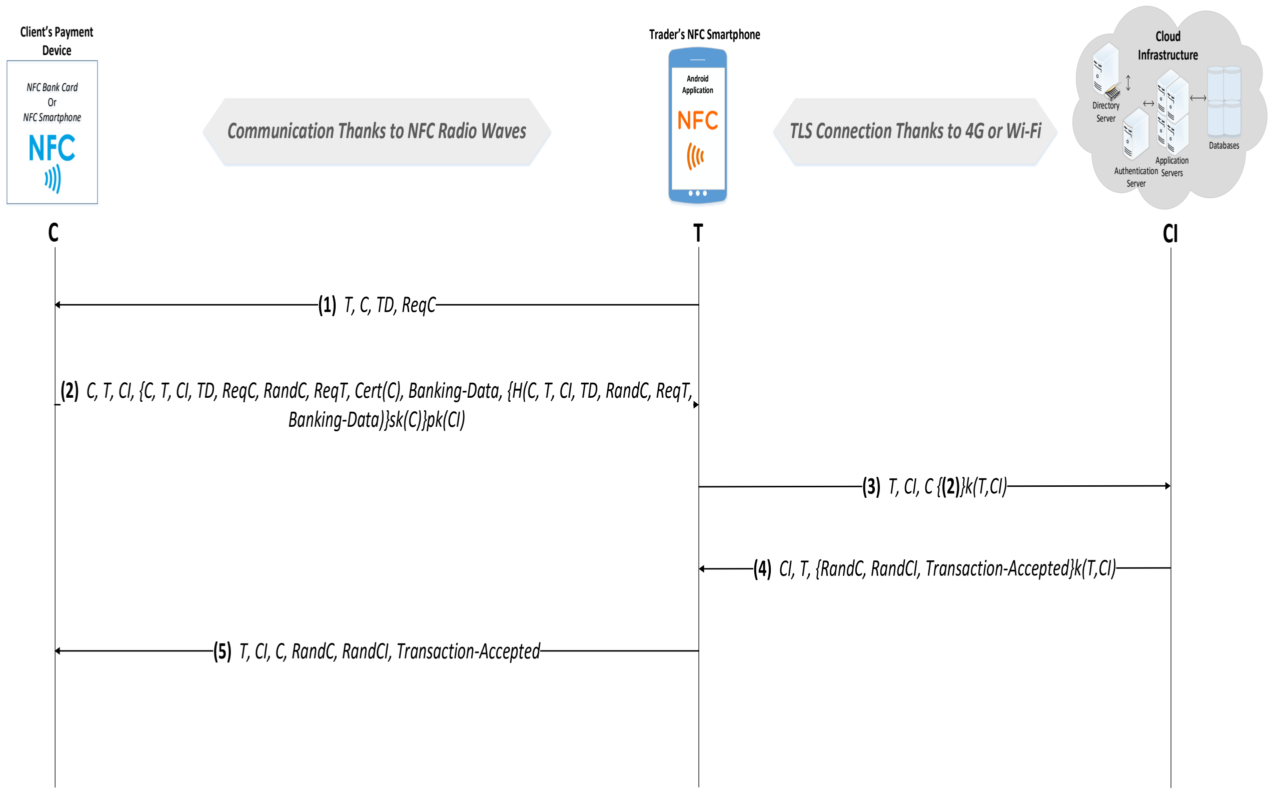

Client's Payment Device
NFC Bank Card Or NFC Smartphone
NFC
Communication Thanks to NFC Radio Waves
Trader's NFC Smartphone
Android Application
NFC
TLS Connection Thanks to 4G or Wi-Fi
Cloud Infrastructure
Directory Server
Authentication Server
Application Servers
Databases
C
T
CI
(1) T, C, TD, ReqC
(2) C, T, CI, {C, T, CI, TD, ReqC, RandC, ReqT, Cert(C), Banking-Data, {H(C, T, CI, TD, RandC, ReqT, Banking-Data)}sk(C)}pk(CI)
(3) T, CI, C {(2)}k(T,CI)
(4) CI, T, {RandC, RandCI, Transaction-Accepted}k(T,CI)
(5) T, CI, C, RandC, RandCI, Transaction-Accepted

Fig. 2. The proposed NFC Payment Architecture for Small Traders

client payment devices. We assume that *T* can connect to *CI* (with Wi-Fi or 4G) via an Android application which implements the Transport Layer Security (TLS) standard [13]: *T* and *CI* communicate securely thanks to the key *k(T,CI)* of the current TLS session.

(3) Client's payment device (*C*): it may be either an NFC bank card or an NFC smartphone. It allows a secure storage for banking data. The user approaches *C* to *T* to make purchases thanks to NFC radio waves. It owns: *pk(C)/sk(C)*, *Cert(C)* signed by the issuing bank, *Cert(CI)*.

B. Targeted Security Properties

The NFC payment architecture proposed in this work allows to simplify and secure NFC payment transactions for small traders by solving EMV vulnerabilities. Therefore, the actors involved in this proposal exchange security messages during the execution of an NFC payment transaction. These messages shall guarantee:

- The security properties that are already well ensured by EMV standard:
  - Authentication of *C* to *T*.
  - Integrity of banking data.
  - Validity of banking data that are not revoked.
- The security properties that are not ensured by EMV standard (see section I):
  - Authentication of *T* to *C*: overcoming vulnerability (1).
  - Confidentiality of banking data: they must be sent in an encrypted text: overcoming vulnerability (2). *T* must be not able to decipher the banking data.

In fact, the mutual authentication between *C* and *T* excludes potential replay attacks where the attacker could usurp the identity of one of them. In addition, the TLS protocol implemented between *T* and *CI* guarantees their mutual authentication and all TLS messages are encrypted thanks to *k(T,CI)*.

C. Description

When a small trader wants to use his NFC smartphone *T* as an NFC merchant payment device to accept NFC payment transactions from client payment devices, two phases are required:

- *Registration phase*: the trader enrolls in the NFC mobile payment service provided by *CI* via an Android application that can be downloaded in *T*: the trader introduces his personal information, his banking details and pays the costs of the NFC mobile payment service allowing to use *T* as an NFC merchant payment device. The Android application allows to *T* to communicate securely with *CI* using the key *k(T,CI)* in order to register *T* as a new NFC merchant payment device in the database of *CI*.
- *NFC payment phase*: thanks to NFC radio waves, the client presents his device *C* to *T* to perform an NFC payment transaction. To authorize and accept the transaction,

the actors involved in this proposal exchange security messages illustrated in Fig-2:

(1) *T->C*: *T* sends to *C* in clear text:
  - Transaction Data *TD* (date, amount, currency code, nonce, etc.). *TD* are generated by *T*, are unique for each transaction and serve to prevent replay attacks.
  - An authentication request for *C ReqC*.

(2) *C->T*: *C* first checks the validity of *TD* and if they are not valid then it will not respond to *T*. Otherwise, it sends to *T* in an encrypted text with *pk(CI)* obtained from *Cert(CI)*:
  - *TD* and *ReqC* received in the message (1).
  - A random number *RandC* to prevent replay attacks.
  - An authentication request for *T ReqT*.
  - Its certificate *Cert(C)*.
  - The banking information: *Banking-Data*.
  - An electronic signature generated by *sk(C)* on the hash of *'C, T, CI, TD, RandC, ReqT, Banking-Data'*. This signature allows to authenticate *C*, guarantees the integrity of the message *'C, T, CI, TD, RandC, ReqT, Banking-Data'* and especially guarantees the integrity of *Banking-Data*.

  In this step, the *Banking-Data* are not sent in clear between *C* and *T*. The latter cannot decipher the message (2) because it does not have the *sk(CI)*.

(3) *T->CI*: *T* sends the message (2) to *CI* in an encrypted text with the current TLS session key *k(T,CI)*. The role of *CI* in this step is to confirm the identity of *T* to *C* and the identify of *C* to *T*. After receiving the message (3), *CI* deciphers it using *k(T,CI)*, deciphers the message (2) using *sk(CI)* and proceeds as following:
  a) Respond to *ReqT*: *CI* has already confirmed the authentication of *T* during the registration phase thanks to TLS standard [13]. Therefore, *T* is well authenticated by *CI*.
  b) Respond to *ReqC*: *CI* verifies the validity *RandC* and if it is not valid, then *CI* will not respond to *T*. Otherwise, it checks the authenticity of *C* by verifying the certificate *Cert(C)* [5] and validating the electronic signature *{H(C, T, CI, TD, RandC, ReqT, Banking-Data)}sk(C)* by *pk(C)*. If *C* is authenticated, then *CI* contacts the issuing bank for authorization, else *CI* finishes the payment transaction with *T*.
  c) Contact the issuing bank: *CI* is connected to the banking network in real time where there is an agreement set between them: if the payment transaction is authorized by the issuing bank, then *CI* accepts the transaction and a bank transfer occurs between the issuing bank and the acquiring bank. Therefore, *CI* connects to the issuing bank to verify the validity of *Banking-Data* that are not revoked and authorize the transaction. If the issuing bank sends to *CI* successful results, then *CI* accepts the payment transaction.

(4) *CI->T*: *CI* sends to *T* in an encrypted text with *k(T,CI)*:
  - *RandC* generated at the beginning by *C*.
  - A random number *RandCI* to prevent replay attacks.
  - A text confirming that the transaction is accepted *Transaction-Accepted*. Also, this text contains the responses to *ReqT* and *ReqC*.

  After receiving (4), *T* deciphers it and verifies the validity of *RandCI* and if it is not valid then *T* finishes the transaction. Otherwise, it confirms from the text *Transaction-Accepted* the authenticity of *C* and that the payment is accepted.

(5) *T->C*: *T* sends to *C* the content of the message (4) in clear text. *C* then confirms:
  - That *T* had contacted *CI* thanks to *RandC* and *RandCI* because: *C* sent *RandC* to *T* in step (2) encrypted by *pk(CI)*, where *T* was not able to obtain it only after decryption by *CI* using *sk(CI)* which is only known by *CI*.
  - From the text *Transaction-Accepted* the authenticity of *T* and that the payment is accepted.

D. Results

During the execution of an NFC payment transaction corresponding to the proposed NFC payment architecture in this work, the security messages exchanged between actors meet the security properties discussed in section II-B as follows:

- Those that are well ensured by EMV standard:
  - Authentication of *C* to *T*: in step (3) of the NFC payment phase, *CI* verifies the authenticity of *C* thanks to *Cert(C)* and the signature generated by *C {H(C, T, CI, TD, RandC, ReqT, Banking-Data)}sk(C)*. In step (4) of the NFC payment phase, *T* confirms the authenticity of *C* thanks to the text *Transaction-Accepted*.
  - Integrity of *Banking-Data*: is guaranteed thanks to the electronic signature generated by *C {H(C, T, CI, TD, RandC, ReqT, Banking-Data)}sk(C)*.
  - Validity of *Banking-Data* that are not revoked is well verified by the issuing bank in step (3) of the NFC payment phase.
- Those that are not ensured by EMV standard:
  - Authentication of *T* to *C* (overcoming vulnerability (1)): during the registration phase, *CI* ensures the authenticity of *T* thanks to TLS protocol. In step (5) of the NFC payment phase, *C* confirms the authenticity of *T* thanks to the text *Transaction accepted*.
  - Confidentiality of *Banking-Data* (overcoming vulnerability (2)): it is ensured thanks to *pk(CI)* in step (2) of NFC payment phase. Also, *T* is not able to obtain the *Banking-Data*.

III. Discussions

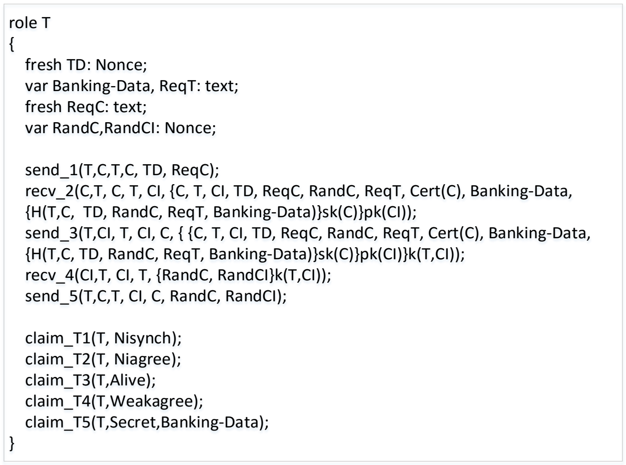
In this section, we examine some attacks between actors of the proposed NFC payment architecture.

1) For the small trader: we assume that an illegal trader wants to use his NFC smartphone *IT* as *T* to communicate using NFC radio waves with *C* and using Wi-Fi or 4G with *CI*. Possible attacks:
   - The registration phase cannot be attacked by *IT* because it does not have a valid certificate to execute TLS protocol with *CI*. Therefore, *IT* cannot obtain a TLS session key [13].
   - In the NFC payment phase:
     - Case 1: In step (1), *IT* sends to *C*: *TDit* generated by itself and *ReqC*. In step (2), *IT* cannot decipher the message (2) because it does not have *sk(CI)* and then it cannot obtain the *Banking-Data*.
     - Case 2: If *IT* replays the message (1) or (3) or (5), then *C* or *CI* can detect these replays during the verification of nonces.
     - Case 3: *IT* cannot decipher the message (3) because it does not have *k(T,CI)*.
2) For the client: we assume that an illegal client wants to use his client's payment device *IC* as *C* to communicate using NFC radio waves with *T*. Possible attacks:
   - In the NFC payment phase:
     - Case 1: In step (2), *IC* sends to *T* a message encrypted by *pk(CI)*: *C* (identify of *C* instead of *IC*), *TD*, *RandIC* generated by *IC*, *ReqT*, *Cert(C)*, *Wrong-Banking-Data*, an electronic signature signed by *sk(IC)* because *IC* does not have *sk(C)*. In step (3), *T* sends the message (2) to *CI* without knowing that it received by an illegal client. Therefore, *CI* rejects the transaction because it could not verify the electronic signature using *pk(C)* and then no confirmation for the authenticity of *C*.
     - Case 2: In step (2), *IC* sends to *T* a message encrypted by *pk(CI)*: *IC* (identify of *IC* instead of *C*), *TD*, *RandIC* generated by *IC*, *ReqT*, *Cert(IC)*, *Wrong-Banking-Data*, an electronic signature signed by *sk(IC)*. In step (3), *T* sends the message (2) to *CI* without knowing that it is received from an illegal client, but *CI* obtains successful results of authenticity of *IC* by verifying the electronic signature using pk(IC). However, the issuing bank will reject the transaction due to the *Wrong-Banking-Data*.
     - Case 3: If *IC* replays the message (2), then *CI* can detect this replay during the verification of nonces.
     - Case 4: *IC* cannot decipher the message (2) because it does not have *pk(CI)*.

IV. Analyzes Using Scyther Tool

A. Overview

We check the correctness of the security exchanged messages of the proposed architecture using a security verification tool called Scyther [14] [15]. The latter allows formal analyzes for security protocols to identify vulnerabilities and attacks. Scyther verifies protocols using Scyther requests/claims: if it detects an attack corresponding to a mentioned claim, then it produces a graph describing this attack.

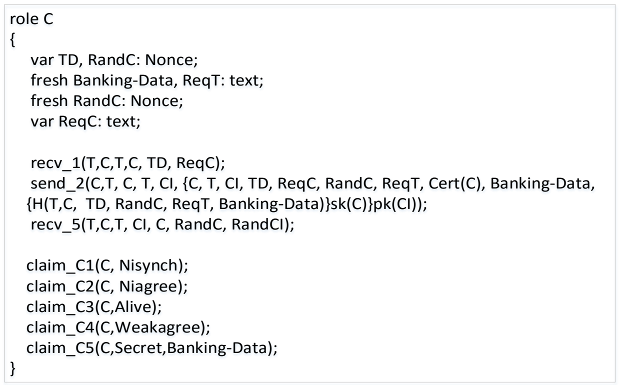


```
role T
{
   fresh TD: Nonce;
   var Banking-Data, ReqT: text;
   fresh ReqC: text;
   var RandC,RandCI: Nonce;

   send_1(T,C,T,C, TD, ReqC);
   recv_2(C,T, C, T, CI, {C, T, CI, TD, ReqC, RandC, ReqT, Cert(C), Banking-Data,
   {H(T,C,  TD, RandC, ReqT, Banking-Data)}sk(C)}pk(CI));
   send_3(T,CI, T, CI, C, { {C, T, CI, TD, ReqC, RandC, ReqT, Cert(C), Banking-Data,
   {H(T,C, TD, RandC, ReqT, Banking-Data)}sk(C)}pk(CI)}k(T,CI));
   recv_4(CI,T, CI, T, {RandC, RandCI}k(T,CI));
   send_5(T,C,T, CI, C, RandC, RandCI);

   claim_T1(T, Nisynch);
   claim_T2(T, Niagree);
   claim_T3(T,Alive);
   claim_T4(T,Weakagree);
   claim_T5(T,Secret,Banking-Data);
}
```

Fig. 3. Role *T* in SPDL Language (Scyther)

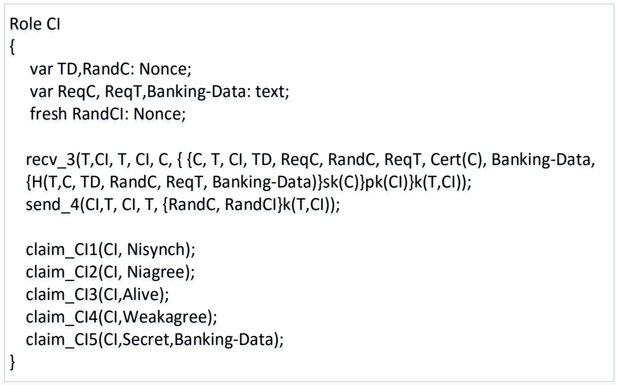


```
role C
{
    var TD, RandC: Nonce;
    fresh Banking-Data, ReqT: text;
    fresh RandC: Nonce;
    var ReqC: text;

    recv_1(T,C,T,C, TD, ReqC);
    send_2(C,T, C, T, CI, {C, T, CI, TD, ReqC, RandC, ReqT, Cert(C), Banking-Data,
   {H(T,C,  TD, RandC, ReqT, Banking-Data)}sk(C)}pk(CI));
    recv_5(T,C,T, CI, C, RandC, RandCI);

   claim_C1(C, Nisynch);
   claim_C2(C, Niagree);
   claim_C3(C,Alive);
   claim_C4(C,Weakagree);
   claim_C5(C,Secret,Banking-Data);
}
```

Fig. 4. Role *C* in SPDL Language (Scyther)



```
Role CI
{
    var TD,RandC: Nonce;
    var ReqC, ReqT,Banking-Data: text;
    fresh RandCI: Nonce;

   recv_3(T,CI, T, CI, C, { {C, T, CI, TD, ReqC, RandC, ReqT, Cert(C), Banking-Data,
   {H(T,C, TD, RandC, ReqT, Banking-Data)}sk(C)}pk(CI)}k(T,CI));
   send_4(CI,T, CI, T, {RandC, RandCI}k(T,CI));

   claim_CI1(CI, Nisynch);
   claim_CI2(CI, Niagree);
   claim_CI3(CI,Alive);
   claim_CI4(CI,Weakagree);
   claim_CI5(CI,Secret,Banking-Data);
}
```

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

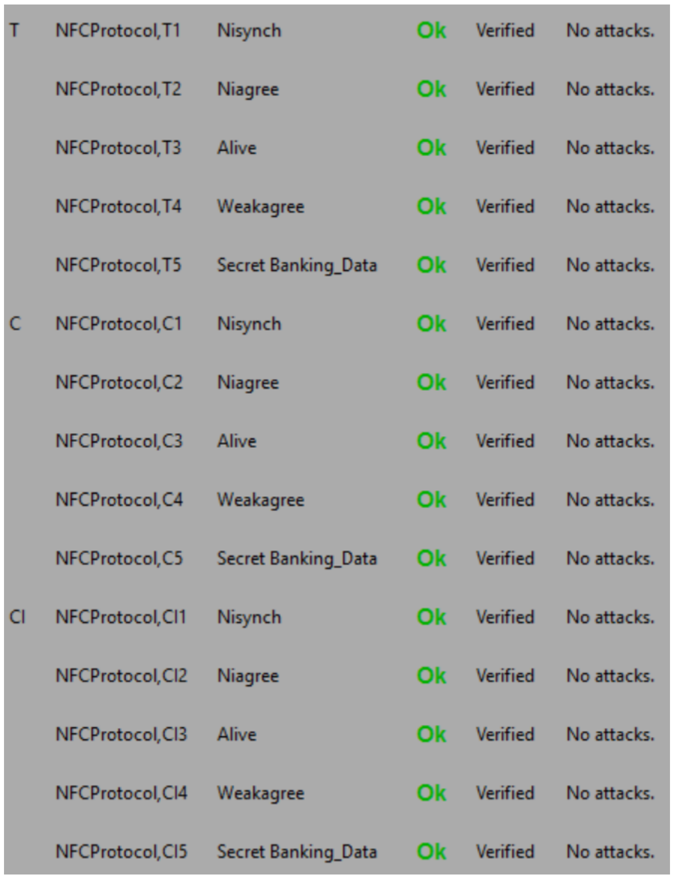
B. Implementation

The Security Protocol Description Language (SPDL) is used to implement protocols in the Scyther tool [16]. Each actor is written as a role in this language. We respectively show the role of *T*, *C* and *CI* in Fig-3, Fig-4, Fig-5. The Scyther claims mentioned in the SPDL implementation refer to the targeted security properties presented in section II-B: (1) *Nisynch,* *Niagree, Alive, Weakagree* for the authentication of *T*, *C* and *CI*, (2) *Secret* for the confidentiality of *Banking-Data*.

C. Scyther Claims Results

The protocol successfully guarantees all Scyther claims for *T*, *C* and *CI* and no attacks are found as illustrated in Fig-6. We provide formal definitions taken from references [16] and [17] for *Nisynch, Niagree, Alive* and *Weakagree* Scyther claims:

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*.



T	NFCProtocol,T1	Nisynch	Ok	Verified	No attacks.
	NFCProtocol,T2	Niagree	Ok	Verified	No attacks.
	NFCProtocol,T3	Alive	Ok	Verified	No attacks.
	NFCProtocol,T4	Weakagree	Ok	Verified	No attacks.
	NFCProtocol,T5	Secret Banking_Data	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	Secret Banking_Data	Ok	Verified	No attacks.
CI	NFCProtocol,CI1	Nisynch	Ok	Verified	No attacks.
	NFCProtocol,CI2	Niagree	Ok	Verified	No attacks.
	NFCProtocol,CI3	Alive	Ok	Verified	No attacks.
	NFCProtocol,CI4	Weakagree	Ok	Verified	No attacks.
	NFCProtocol,CI5	Secret Banking_Data	Ok	Verified	No attacks.

Fig. 6. Results with Scyther Tool

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".

V. Conclusion and Future Work

In this paper, we proposed a new secure NFC payment architecture based on a cloud infrastructure for small traders: where they become able to use their NFC smartphones as NFC merchant payment devices. The proposal adequately solves EMV security vulnerabilities. In our next work, we will implement the proposed architecture in a secure and real environment and will extend this paper to a journal version.

References

[1] H. A. Al-Ofeishat and A. Mohammad, "Near field communication (nfc)," *International Journal of Computer Science and Network Security*, 2012.

[2] VISA payWave, last connection (25/09/2016). [Online]. Available: http://www.visa.ca/fr/personal/visa-paywave/index.jsp

[3] MasterCard PayPass, last connection (25/09/2016). [Online]. Available: http://www.mastercard.com/contactless/index.html

[4] 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.

[5] N. El Madhoun and G. Pujolle, "Security enhancements in emv protocol for nfc mobile payment," *The 15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-16)*, 2016.

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

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

[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] T. Luo, H. Hao, W. Du, Y. Wang, and H. Yin, "Attacks on webview in the android system," *Proceedings of the 27th Annual Computer Security Applications Conference*, pp. 343–352, 2011.

[10] L. Davi, A. Dmitrienko, A.-R. Sadeghi, and M. Winandy, "Privilege escalation attacks on android," *International Conference on Information Security*, pp. 346–360, 2010.

[11] 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.

[12] 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.

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

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

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

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

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