New Security Protocols for Offline Point-of-Sale Machines

Nour El Madhoun, Emmanuel Bertin, Mohamad Badra, Guy Pujolle

New Security Protocols for Offline Point-of-Sale Machines

Nour El Madhoun[1], Emmanuel Bertin[2], Mohamad Badra[3], and Guy Pujolle[4]

[1] Security and System Laboratory, EPITA, 14-16 Rue Voltaire 94270 Le Kremlin-Bicêtre, France
nour.el-madhoun@epita.fr
[2] Orange Labs, 42 rue des Coutures BP 6243 14066 Caen, France
emmanuel.bertin@orange.com
[3] College of Technological Innovation, Zayed University, P.O. Box 19282 Dubai, U.A.E
mohamad.badra@zu.ac.ae
[4] Sorbonne Université, CNRS, LIP6, 4 place Jussieu 75005 Paris, France
guy.pujolle@lip6.fr

Abstract. EMV (Europay MasterCard Visa) is the protocol implemented to secure the communication between a client's payment device and a Point-of-Sale machine during a contact or an NFC (Near Field Communication) purchase transaction. In several studies, researchers have analyzed the operation of this protocol in order to verify its safety: unfortunately, they have identified two security vulnerabilities that lead to multiple attacks and dangerous risks threatening both clients and merchants. In this paper, we are interested in proposing new security solutions that aim to overcome the two dangerous EMV vulnerabilities. Our solutions address the case of Point-of-Sale machines that do not have access to the banking network and are therefore in the "offline" connectivity mode. We verify the accuracy of our proposals by using the Scyther security verification tool.

Keywords: EMV protocol · EMV vulnerabilities · NFC · offline · payment · security.

1 Introduction

EMV is the international protocol implemented to secure contact and contactless-NFC purchase transactions. In order to execute a secure EMV purchase transaction, five EMV actors (see Fig-1) exchange with each other a sequence of security messages [1]. Indeed, the description of the EMV protocol is non-trivial, due to the high complexity of the EMV specifications in more than 1000 pages in [2–7]. Therefore, in our previous work [8], we introduced a synthetic overview of this protocol: it is essential to refer to this work in order to clarify the roles of the EMV actors and the description of the EMV exchanged messages (EMV phases).

Indeed, in various studies, the operation of EMV protocol has been analyzed in order to verify its reliability: several security vulnerabilities have been

identified in this protocol and they represent major risks for our day to day safety [9] [10] [11] [12] [13] [14]. In our work [1], we presented a survey of these security vulnerabilities as well as we discussed the possible attacks due to them. Consequently, there are *two EMV security vulnerabilities* that have raised our attention to address them in this paper among the other EMV vulnerabilities [13]:

* ***Vulnerability (1)***: the confidentiality of the *Banking-Data* is not ensured: the *PAN* (Primary Account Number) and *ExpDate* (Expiration Date) are sent in clear from C (Client's Payment Device) to P (Point of Sale). The latter can also store them.
* ***Vulnerability (2)***: with the detection of the *Vulnerability (1)*, also the authentication of P to C is not ensured. C can answer any device without authenticating it, by sending the *Banking-Data* (*PAN* and *ExpDate*) in clear. Indeed, since the authentication of P to C is not ensured, then the non-repudiation for P is also not ensured.

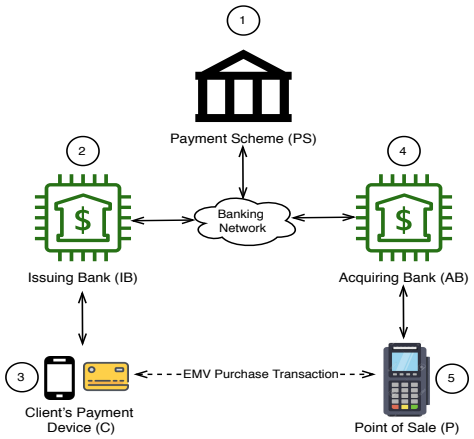

Fig. 1: EMV payment system [8]

Accordingly, in our previous work [8], we proposed a new security solution that solves these *two EMV weaknesses* in the case where P can communicate with the banking network and then it is in the "online" connectivity mode. In this paper, we are interested in solving the *two EMV weaknesses* in the case where P is in the "offline" connectivity mode and then it cannot communicate with the banking network. More specifically, in this paper, we propose two new security solutions:

– The first solution is designed for the case where P and C are both in the "offline" connectivity mode.
– The second solution is designed for the case where P is in the "offline" connectivity mode and C is in the "online" connectivity mode.

The difference between the two solutions lies in the dependence on the availability of the connection at C. This dependence makes it possible to take advantage of the availability of the Internet connection to communicate with a third party and execute security procedures.

In this paper, we use several acronyms that have been defined by default in existing literature, and in addition, we propose new abbreviations. Therefore, in order to simplify the reading of this paper, we describe in Table 1 the different abbreviations that are hereby used. In Table 2, we present the elements and procedures of security for each EMV actor. It is essential to consult these two tables while reading the paper.

The organization of this paper is as follows. In section 2, we give an essential background of our proposals. In section 3, we describe our proposals that aim to improve the cryptographic layer of the EMV protocol, where P is in the "offline" connectivity mode, by solving *the two EMV vulnerabilities* (discussed above). In section 4, we discuss and evaluate the results of our proposals. In section 5, we analyze our proposals in a formal way using the Scyther tool. The last section provides a brief conclusion.

Table 1: Abbreviations Used in the Paper

Abb.	**Description**
AB	Acquiring Bank
$ARQC$	Authorization ReQuest Cryptogram
$AuthorizReq$	Authorization transaction Request
$AuthorizResp$	Authorization transaction Response
C	Client's payment device
CA	Certificate Authority
CAp	Certificate Authority for P
$CertX$	Certificate of X
$CK\text{-}AC$	Symmetric Card Key derived from ($IMK\text{-}AC$ + PAN). It is only shared between IB and C. It is used to generate cryptograms as $ARQC$.
$CK\text{-}SMC$	Symmetric Card Key derived from ($IMK\text{-}SMC$ + PAN). It is only shared between IB and C. It is used for secure messaging as 'PIN change'.
$ConfC$	Confirmation of the authenticity of C, the non-repudiation for C and the integrity of $Banking\text{-}Data$. In our proposal, $ConfC$ indicates also that IB authorizes the transaction
$ConfP$	Confirmation of the authenticity of P, the non-repudiation for P and the integrity of the signed message.
$ConfigInfo$	Configuration Information: results of phases 1 and 2 and other data.
EMV	Europay Mastercard Visa
$ExpDate$	Expiration Date
$H(M)$	One way Hashing function of M = $m1$,$m2$..
IB	Issuing Bank
IdC	Identifier of C in the database of IB. This identifier is not a sensitive information
$IMK\text{-}AC$	Symmetric Issuer Master Key for Application Cryptogram Computation. It is only shared between IB and C.
$IMK\text{-}SMC$	Symmetric Issuer Master Key for Secure Messaging for Confidentiality. It is only shared between IB and C.
$k(X,Y)$	Symmetric key of the current TLS session allows to protect information exchanged between X and Y
MAC	Message Authentication Code
NFC	Near Field Communication
OnX	Specify that X is able to connect to the internet
$OffX$	Specify that X is not able to connect to the internet
P	Point of Sale
PAN	Primary Account Number
PIN	Personal Identification Number
$pk(X)$	RSA Public Key of X
PS	Payment Scheme (Visa, MasterCard, etc.)
$ReqX$	Request for authentication of X and non-repudiation for X.
RSA	Rivest–Shamir–Adleman
RX	Unpredictable Random Number generated by X(Unique)
$SignX$	Digital Signature generated by X using its $sk(X)$ on the hash of a message. It allows to ensure the authentication of X, the non-repudiation for X and the integrity of the message (thanks to the hash)
$sk(X)$	RSA Secret private Key of X
TD	Transaction Data generated by P (amount, country code, nonce, currency,..). TD are unique for each transaction
TLS	Transport Layer Security
X	It represents: C, P, AB, PS or IB.
Y	It represents: C, P, AB, PS or IB.

2 Background of our Proposals

As discussed in the previous section, in this paper, we propose two new security solutions that mainly aim to overcome the *two EMV weaknesses*. Our solutions deal with the case of P machines that do not have access to the banking network

and that are therefore in the "offline" connectivity mode. We summarize in Table 3 the characteristics of our security solutions and we list them as follows:

- 1st Security Solution *(OffC/OffP)*: C and P are offline. It allows to secure contact and NFC purchase transactions. C may be either a bank card or an NFC smartphone.
- 2nd Security Solution *(OnC/OffP)*: C is online and P is offline. It allows to secure only NFC purchase transactions because it supports an online C which can be only an NFC smartphone and cannot be a bank card. The latter does not have a Wi-Fi or 4G interface.

Table 2: Elements & Procedures of Security for EMV Actors [8] [14]

Actor	Description
PS	• *PS* acts as a CA to certify *IB*. • *PS* has *pk(PS)/sk(PS)*: RSA root keys self-generated. • *PS* has *CertPS*: self-signed and contains *pk(PS)*.
IB	• *IB* generates two symmetric keys *IMK-AC* and *IMK-SMC*. • For each *PS* contracted with *IB*: – *IB* stores the *CertPS* as a trusted anchor *CA*. – *IB* securely generates an RSA key pairs *pk(IB)/sk(IB)*. – *PS* signs *pk(IB)* with *sk(PS)* where the *CertIB* is generated.
C	• *C* can belong to a single contracted *PS* at once. • *C* stores in its secure element the security information: – Client's name. – *Banking-Data* (*PAN*, *ExpDate*): which are generated by *IB*. They are very sensitive information because they allow to perform purchase transactions. Indeed, the *PAN* allows to identify *C* in the database of *IB*. – Static signature generated by *IB* using *sk(IB)*: *{H(Banking-Data)}sk(IB)*. – *pk(C)/sk(C)*: they are generated by *IB* for *C*. *IB* also signs *pk(C)* with its *sk(IB)* where the *CertC* is produced. – *CK-AC* generated by *IB* and used by *C* to generate an *ARQC*. – *CK-SMC* generated by *IB* and used by *C* for secure messaging. – *CertPS*, *CertIB* and *CertC*.
AB	• For each *PS* contracted with *AB*: *AB* stores *CertPS* as a trusted anchor *CA*.
P	• For each *P*, *AB* stores the *CertPS* of each contracted *PS*.

Table 3: Characteristics of the Proposed Security Solutions

Characteristics Solution	1st - *OffC/OffP*	2nd - *OnC/OffP*
C Connectivity mode	Offline	Online
P Connectivity mode	Offline	Offline
C may be a bank card	√	-
C may be an NFC smartphone	√	√
Secure contact purchases	√	-
Secure NFC purchases	√	√

2.1 Actors

The two main actors in our security solutions are C and P:

- For the 1st solution *(OffC/OffP)*, C and P cannot communicate with a third actor.
- For the 2nd solution *(OnC/OffP)*, C can communicate with the banking network "*AB, PS, IB*" (see Table 1 and see Fig-1).

In addition, we suggest to add a new Certification Authority for P: *CAp*. The latter has a pair of RSA root keys *pk(CAp)* and *sk(CAp)* and a certificate *CertCAp* signed by itself. The role of *CAp* in this paper is to certify P by generating it an RSA key pair *pk(P)/sk(P)* and signing *pk(P)* using *sk(CAp)* to produce *CertP*. We also assume that *CAp* is considered by the banking network (*AB, PS, IB*) as a trusted anchor.

2.2 Authentication Procedure for *P*

We suggest that the actor P needs to generate an electronic signature *SignP*, using its *sk(P)*, in order to authenticate itself to C. The verification of the authenticity of P, in order to respond to *ReqP*, will be performed by another actor as follows [8]:

- Verification if the issuer of *CertP* is a trusted certification authority: *CAp*.
- Verification if *CertP* is valid (validity period).
- Verification if *pk(CAp)* validates the signature of *CertP*.
- Verification if *pk(P)* (obtained from *CertP*) validates *SignP*.

Table 4: Summary of Targeted Security Properties

	Targeted Security Property	**Is it ensured by the EMV protocol ?**
a)	Integrity of *Banking-Data*	✓
b)	Validity of *Banking-Data*	✓
c.a)	Confidentiality of *Banking-Data*: They must be sent encrypted from C to IB	*Vulnerability (1)* (see section 1)
c.b)	Confidentiality of *Banking-Data*: P must not be able to obtain or store them	*Vulnerability (1)* (see section 1)
d.a)	Authentication of C to P	✓
d.b)	Authentication of P to C	*Vulnerability (2)* (see section 1)
e.a)	Non-repudiation for C	✓
e.b)	Non-repudiation for P	*Vulnerability (2)* (see section 1)

2.3 Targeted Security Properties

During a purchase transaction, each proposed solution is represented in a set of security messages that are exchanged between the involved actors. Indeed, we aim that these messages guarantee the following security properties:

a) Integrity of *Banking-Data*: they must not be modified.

b) Validity of *Banking-Data*: they must not be revoked.

c) Confidentiality of *Banking-Data*:

- *c.a)* They must be sent encrypted from C to IB.
- *c.b)* P must not be able to obtain or store them.

d) Mutual authentication: it is a strong agreement excluding man-in-the-middle and potential replay attacks where an attacker could usurp the identity of one of them.

- *d.a)* Authentication of C to P.
- *d.b)* Authentication of P to C.

e) Non-repudiation of origin: C and P must not be able to deny in the future: strong evidence sent by themselves or their participation in the purchase transaction.

- *e.a)* Non-repudiation of origin for C.
- *e.b)* Non-repudiation of origin for P.

In Table 4, we illustrate each targeted security property to be insured by our proposals and we show whether or not it is guaranteed in the EMV protocol.

2.4 Objectives of our Proposals

Each Security Solution proposed in this paper has several objectives [8]:

- **Objective 1**: to improve the security of the classical EMV protocol by:
  - *Challenge* 1: ensuring the following security properties (see sections 1, 2.3 and 4.1):
    - $\{\underline{c.a)}, \underline{c.b)}\}$ in order to overcome the EMV *Vulnerability (1)*.
    - $\{\underline{d.b)}, \underline{e.b)}\}$ in order to overcome the EMV *Vulnerability (2)*.
  - *Challenge* 2: ensuring the security properties $\{\underline{a)}, \underline{b)}, \underline{d.a)}, \underline{e.a)}\}$ that are typically well ensured by the EMV standard (see sections 1, 2.3 and 4.1).
  - *Challenge* 3: putting P as a trusted element, where all communications are secured/ transparent for P: property $\underline{c.b)}$.
- **Objective 2**: to respect the same EMV principle in order to be doable in the real-world in the future by:
  - *Challenge* 4: using the main security keys and certificates of the EMV payment system (see Table 2) and adding a new security layer if judged necessary (new keys, new certificates, etc.).
- **Objective 3**: to effectively use the resources of C, which is poor in the speed of calculation, in order to guarantee a fast run by:
  - *Challenge* 5: avoiding if it is possible that C performs asymmetric cryptographic functions (except for the EMV usual functions). This is because the asymmetric encryption/decryption tends to be 1000 times slower than the symmetric encryption/decryption [15] [16].
  - *Challenge* 6: offloading the execution of the asymmetric functions, that has been avoided to execute in C, to another actor that has more powerful resources than C.

2.5 Assumptions of our Proposals

- For the two solutions proposed in this paper, we assume that:
  - C is identified by *IdC* in the database of *IB* and not by the *PAN* as in the EMV protocol [8]. *IdC* is not a sensitive information as the *PAN*, it does not present a risk if it is sent in clear because it does not allow to make purchases.
  - C produces *SignC* to authenticate itself.
  - P produces *SignP* to authenticate itself.
  - *SignP* allows to ensure the properties $\{\underline{d.b)}, \underline{e.b)}\}$ (overcoming *Vulnerability (2)*), and to guarantee the integrity of the signed message (see Tables 1 and 4).
- For the 1st Security Solution *(OffC/OffP)*, we assume that:
  - C stores *CertCAp* as a trusted certificate of a trusted certification authority.

- *SignC* does not include the *Banking-Data* and allows to ensure the properties $\{\underline{d.a}), \underline{e.a})\}$ and to guarantee the integrity of the signed message.
- *C* proceeds to verify the authenticity of *P*.
- *P* proceeds to verify the authenticity of *C*.
- *P* periodically receives, when it is connected to the Internet, the list of revoked certificates of clients..
- If *CertC* is valid, then the *PAN*/*ExpDate* are valid.
- If *CertC* is revoked or not valid, then the *PAN*/*ExpDate* are also revoked or not valid.
- If *CertP* is revoked or not valid, then *P* cannot produce *SignP*.

– For the 2nd Security Solution *(OnC/OffP)*, we assume that:
- *C* can securely communicate with *IB* thanks to the TLS protocol using a TLS session key: *k(C,IB)*.
- *SignC* includes the *Banking-Data* and allows to ensure the properties $\{\underline{a}), \underline{d.a}), \underline{e.a})\}$ and to guarantee the integrity of the signed message (see Tables 1 and 4).
- *AB* stores *CertCAp* as a trusted certificate of a trusted certification authority.
- *AB* proceeds to verify the authenticity of *P*.
- *IB* proceeds to verify the authenticity of *C*.
- *C* will not use the *CK-AC* to generate an *ARQC* but it will use the *CK-SMC* to generate an enciphered message (see Tables 1 and 2).
- *P* will not proceed to verify the authenticity of *C* but it will wait, from *IB*, for the confirmation of the authenticity of *C* (thanks to *ConfC*, see Table 1).

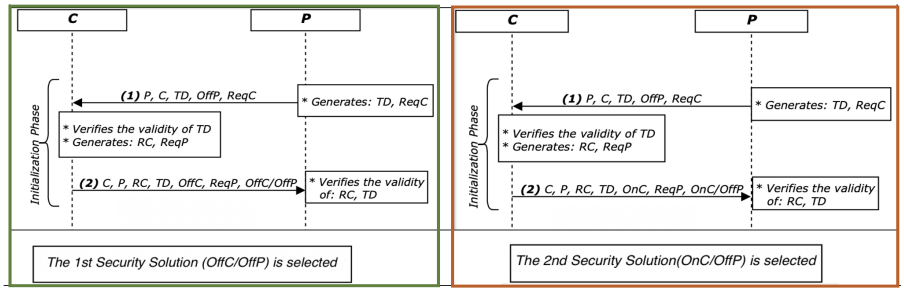

Fig. 2: Initialization (Phase 1)

3 Description of our Proposals

During a purchase transaction, each proposed solution is represented in a set of security messages that are exchanged between the involved actors. For each of our proposals, we suggest dividing these security messages into three phases (*Initialization (Phase 1)*, *Authentication of the Client (Phase 2)*, *Authentication of C and P (Phase 3)*) instead of four phases as in the EMV security protocol: as specified in section 1, it is essential to refer to our work [8] in order to clarify the roles of the EMV actors and the description of the EMV phases.

3.1 Phase 1 and Phase 2

These two phases are more or less the same for the two proposed security solutions.

Initialization (Phase 1) this phase is illustrated in Fig. 2. It allows, firstly, P to inform its connectivity mode to C, and secondly, C to inform its connectivity mode to P. The response from C gives the decision for the **Security Solution** that will be chosen subsequently to execute in phase 3.

(1) ***P*->*C***:
- 1.1 P generates *TD* that aims to prevent replay attacks and an authentication request for C *ReqC*. Then, it sends to C: *TD*, P connectivity mode (*OnP* or *OffP*) and *ReqC*.
- 1.2 C receives the message **(1)**, verifies the validity of *TD*, generates *RC* to prevent replay attacks and and an authentication request for P *ReqP*.

(2) ***C*->*P***:
- 2.1 C responds to P by sending: *RC*, *TD*, C connectivity mode (*OnC* or *OffC*), *ReqP* and the decision "C connectivity mode"/"P connectivity mode".
- 2.2 P receives **(2)** and verifies the validity of *RC* and *TD*. It will respond to *ReqP* in phase 3.
- 2.3 P also takes into consideration the decision of C, and then, according to this decision, one of the security solutions is selected for the execution in phase 3:
  - *OffC/OffP*: the 1st Security Solution is selected if C and P are offline.
  - *OnC/OffP*: the 2nd Security Solution is selected if C is online and P is offline.

Authentication of the Client (Phase 2) this phase is executed in the same manner as in the original EMV protocol thanks to a PIN code or a signature or nothing if the payment is contactless-NFC [8].

3.2 Description of Phase 3 for the 1st Security Solution *(OffC/OffP)*

After the execution of phases 1 and 2, if the 1st Security Solution *(OffC/OffP)* is selected, phase 3 is executed as follows (see Fig. 3):

(1) ***P*->*C***:
- 1.1 P generates *SignP* with its *sk(P)* principally on the hash of {*TD, RC, ReqC, OffC/OffP,...*} and sends to C: *TD, RC, CertP* and *SignP*.
- 1.2 After receiving **(1)**, C verifies the validity of nonces and the authenticity of P as presented in section 2.2.
- 1.3 *ConfP* is a formal message generated by C to indicate that P is well authenticated (properties {*d.b*), *e.b*)}, overcoming *Vulnerability (2)*).
- 1.4 C produces *SignC* with its *sk(C)* mainly on the hash of {*..., IdC, RC, TD, ConfP, OffC/OffP, ConfigInfo,...*} and without including the *Banking-Data*: *PAN, ExpDate*.

(2) ***C->P***:

2.1 *C* sends to *P* in clear: *ConfigInfo, ConfP, CertC, CertIB* and *SignC*. *C* does not send the *Banking-Data* in clear.

2.2 Since the *Banking-Data* are not sent from *C* to *P*, then they cannot be modified or changed (property $\underline{a}$)). Additionally, in an implicit way, they will remain confidential to *P* and *IB* (properties {$\underline{c.a}$), $\underline{c.b}$)}, overcoming *Vulnerability (1)*).

2.3 *P* receives **(2)**, confirms that it was well authenticated by *C* through *ConfP* and proceeds to authenticate *C* by verifying that:
- *PS* is the issuer of *CertIB*.
- *CertIB* and *CertC* are valid today.
- *pk(PS)* validates the signature contained in *CertIB*.
- *pk(IB)* validates the signature contained in *CertC*.
- *pk(P)* validates *SignC*.

2.4 *P* confirms the authenticity of *C* by generating the formal message *ConfC* (properties {$\underline{d.a}$), $\underline{e.a}$)}). *P* also confirms that the *PAN* and *ExpDate* are valid because *CertC* is valid (property $\underline{b}$)).

(3) ***P->C***:

3.1 *P* sends to *C*: *TD, RC, ConfC*. Then, *C* verifies the validity of nonces and confirms that it was well authenticated by *P* thanks to *ConfC*.

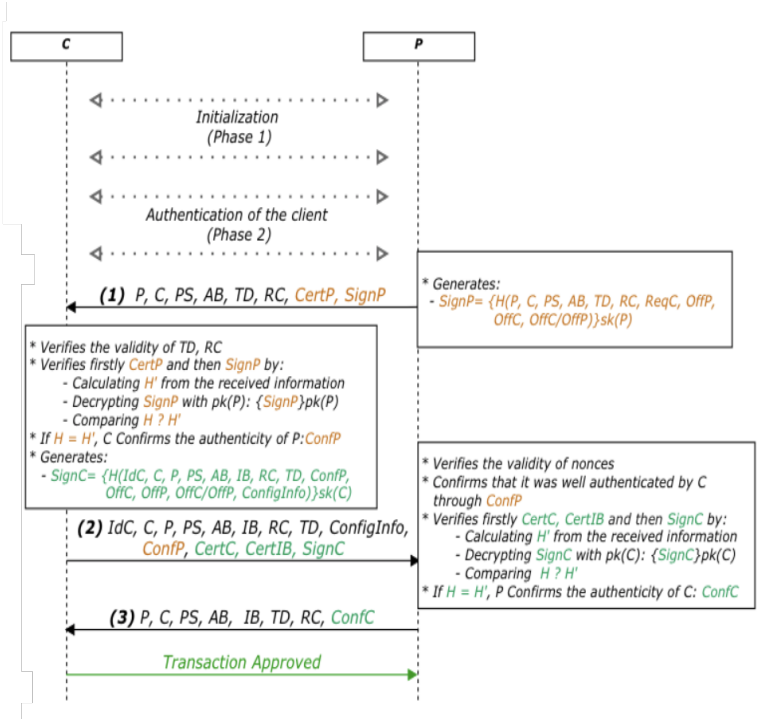

Fig. 3: The 1st Security Solution *(OffC/OffP)* (Phase 3)

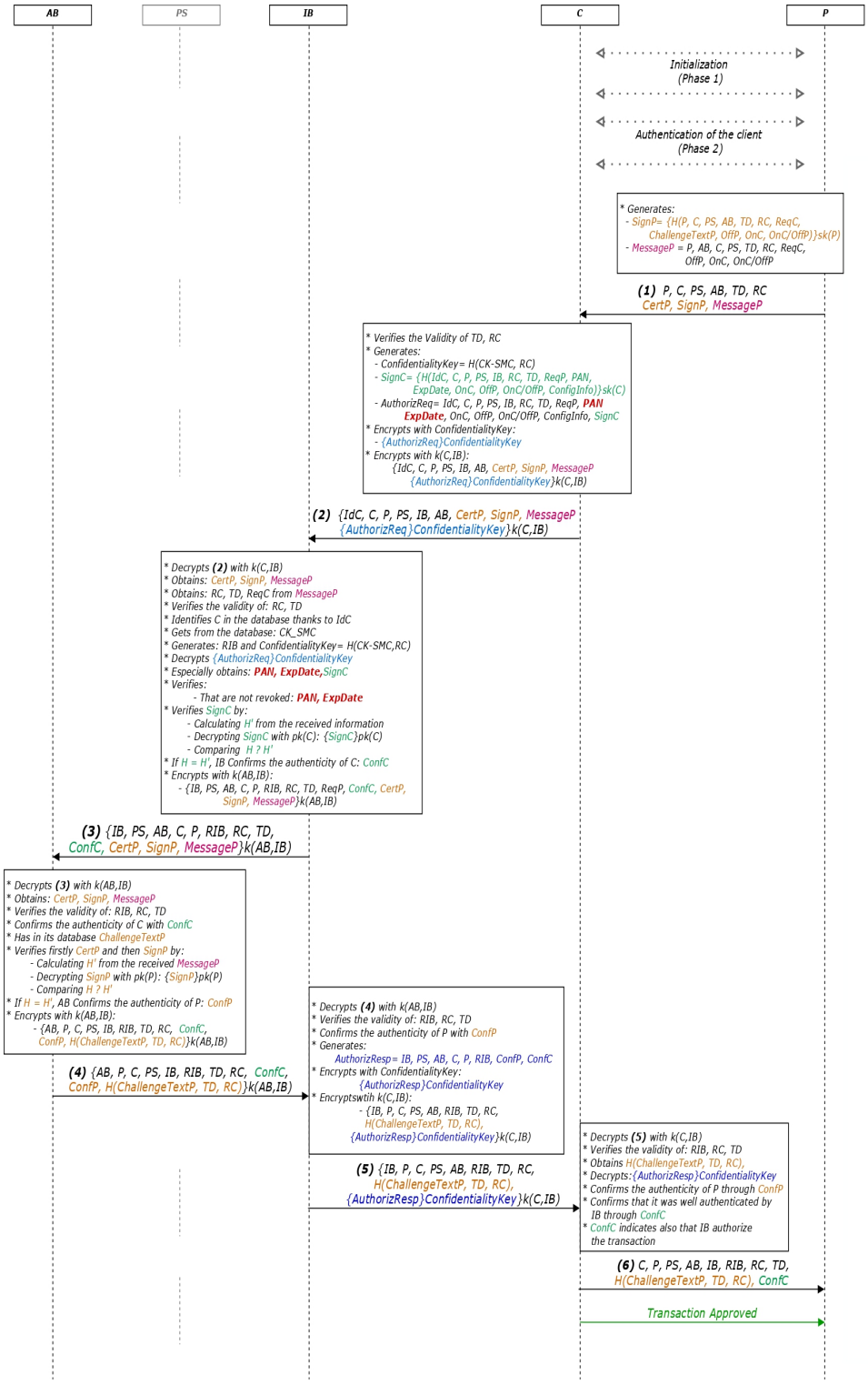

Fig. 4: The 2nd Security Solution *(OnC/OffP)* (Phase 3)

3.3 Description of Phase 3 for the 2nd Security Solution *(OnC/OffP)*

After the execution of phases 1 and 2, if the 2nd Security Solution *(OnC/OffP)* is selected, phase 3 is executed as follows (see Fig. 4):

(1) ***P->C***:

1.1 *P* generates *SignP* with its *sk(P)* mainly on the hash of {*TD, RC, ChallengeTextP, OnC/OffP,...*}. *ChallengeTextP* is a confidential text known only by *P* and *AB*. So, *P* will wait to receive *ChallengeTextP* from *C* to confirm that *C* has well contacted *AB*.

1.2 *P* also prepares the *MessageP* that mainly contains {*TD, RC, ReqC, OnC/OffP ...*}. Afterwards, it sends to *C*: *CertP*, *SignP* and the *MessageP*.

1.3 After receiving **(1)**, *C* will not verify the authenticity of *P*, but it calculates the *ConfidentialityKey* as a symmetric session key by hashing both the *CK-SMC* and *RC*.

1.4 *C* produces *SignC* with its *sk(C)* mainly on the hash of {*IdC, RC, TD, PAN, ExpDate, OnC/OffP, ConfigInfo,...*}.

1.5 *C* also creates an authorization transaction request for *IB* '*AuthorizReq*', that mainly contains: {*IdC, RC, TD, PAN, ExpDate, OnC/OffP, ConfigInfo, SignC,...*}. *C* encrypts *AuthorizReq* using the *ConfidentialityKey*.

(2) ***C->IB***:

2.1 *C* sends to *IB* in an encrypted text with the current TLS session *k(C,IB)*: *CertP, SignP, MessageP* and the *{AuthorizReq}ConfidentialityKey*.

2.2 *{AuthorizReq}ConfidentialityKey* asks if *IB* authorizes the transaction or not and allows to ensure the properties {$\underline{c.a)}$, $\underline{c.b)}$} (overcoming *Vulnerability (1)*). It can be deciphered only by $\overline{IB}$. In addition, the *AuthorizReq* contains *SignC* which allows to ensure the properties {$\underline{a)}$, $\underline{d.a)}$, $\underline{e.a)}$}.

2.3 *IB* receives the message **(2)**, decrypts it using *k(C,IB)*, obtains *CertP, SignP, MessageP*, verifies the validity of *RC, TD* and generates a random number *RIB* that serves to prevent replay attacks.

2.4 *IB* then identifies *C* in its database through *IdC* and gets *CK-SMC*. The latter is required to calculate the *ConfidentialityKey* as illustrated in Fig. 4.

2.5 *IB* deciphers *{AuthorizReq}ConfidentialityKey* and especially obtains *PAN*, *ExpDate, SignC* (property $\underline{c.a)}$, overcoming *Vulnerability (1)*).

2.6 *IB* verifies that the *PAN*, *ExpDate* are not revoked (property $\overline{b)}$) and it will respond to *ReqC* by verifying *SignC* using *pk(C)* obtained from its database.

2.7 *ConfC* is a formal message generated by *IB* to indicate that *C* is well authenticated (properties $\underline{a)}$, $\underline{d.a)}$, $\underline{e.a)}$) and that the transaction is well authorized by *IB*.

(3) ***IB->AB***:

3.1 *IB* sends to *AB* in encrypted text with the current TLS session key *k(AB,IB)*: *RIB, RC, TD, ReqP, ConfC, CertP, SignP, MessageP*.

3.2 After receiving **(3)**, *AB* deciphers it using *k(AB,IB)*, verifies the validity of *RIB, RC, TD*, confirms the authenticity of *C* through *ConfC* and obtains *CertP, SignP, MessageP*.

3.3 *AB* is the bank of *P* and also allows to verify the authenticity of *P*. Then, it will retrieve *ChallengeTextP* from its database and respond to *ReqP* as presented in section 2.2.

3.4 *ConfP* is a formal message generated by *AB* to indicate that *P* is well authenticated (properties $\{\overline{d.b}), \overline{e.b)}\}$, overcoming *Vulnerability (2)*). Also, *H(ChallengeTextP, TD,* $\overline{RC)}$ is a dynamic hash message which is destined to *C*.

(4) ***AB->IB***:

4.1 *AB* sends to *IB* in an encrypted text with the current TLS session key *k(AB,IB)*: *TD, RC, ConfC, ConfP, H(ChallengeTextP, TD, RC)*.

4.2 *IB* receives the message **(4)**, decrypts it using *k(AB,IB)*, verifies the validity of nonces and confirms the authentication of *P* thanks to *ConfP*.

4.3 *IB* creates an authorization response *AuthorizResp* for *C* containing mainly: *{RIB, ConfP, ConfC}* and encrypts it using the *ConfidentialityKey*.

(5) ***IB->C***:

5.1 *IB* sends to *C* in an encrypted text with the current TLS session key *k(C,IB)*: *RIB, RC, TD, H(ChallengeTextP, TD, RC), {AuthorizResp}ConfidentialityKey*. *C* receives the message **(4)**, deciphers it, verifies nonces, obtains *H(ChallengeTextP, TD, RC)* and the *{AuthorizResp}ConfidentialityKey*.

5.2 *C* decrypts *{AuthorizResp}ConfidentialityKey*, confirms the authenticity of *P* thanks to *ConfP*, confirms thanks to *ConfC* that it was well authenticated by *IB* and that the transaction is authorized by *IB*.

(6) ***C->P***:

6.1 *C* sends to *P*: *RIB, RC, TD, H(ChallengeTextP, TD, RC), ConfC*. Then, *P* verifies the validity of nonces, calculates another hash *H'* of *ChallengeTextP, TD, RC*, compares the calculated hash *H'(ChallengeTextP, TD, RC)* with the received hash *H(ChallengeTextP, TD, RC)* and confirms that *C* has well contacted *AB*. *P* also confirms thanks to *ConfC* the authenticity of *C* and that the transaction is well authorized by *IB*.

4 Results of the Proposals

4.1 Discussions

Our two security solutions achieve the *Objectives* presented in section 2.4:

– **Objective 1**: improvement of the security of the EMV classical protocol as follows :
  - In our two proposals, as illustarted in Table 5, the security messages exchanged between the actors, during a payment transaction, meet the targeted security properties {*a*), *b*), *c.a*), *c.b*) *d.a*), *d.b*), *e.a*), *e.b*)} (see Table 4 and section 2.3). However, the EMV protocol meets only the security properties {*a*), *b*), *d.a*), *e.a*)} (see Table 4).
  - In our two proposals, *P* does not receive the *Banking-Data* either in clear nor in an encrypted manner because it is in the offline mode, and then, it is not able also to hack these data: property *c.b*).
  - In our 2nd solution, *P* does not verify the authenticity of *C* by itself because *SignC* includes the *Banking-Data*. However, in our 1st solution, *P* verifies the authenticity of *C* by itself but *SignC* does not include the *Banking-Data*. Hence, in our two proposals, we place *P* as an element of trust after its authentication by *AB* or by *C*.

Table 5: Achieved Security Properties in our Proposals

Properties	**Steps in which they have been achieved (Phase 3)**
——————**The 1st Security Solution *OffC/OffP***——————	
a)	- In message **(2)**: 2.1, 2.2. The *Banking-Data* are not sent by *C* to *P* and then they cannot be modified.
c.a), *c.b*)	- In message **(2)**: 2.2. The *Banking-Data* are not sent by *C* to *P*, and then in an implicit way, they remain confidential to *P* and *IB*.
d.a), *e.a*)	- In message **(2)**: 2.3, 2.4. Thanks to *SignC* that is verified by *P*.
b)	- In message **(2)**: 2.4.
d.b), *e.b*)	- In message **(1)**: 1.1, 1.2, 1.3. Thanks to *SignP* that is verified by *C*.
——————**The 2nd Security Solution *OnC/OffP***——————	
a), *d.a*), *e.a*)	- In message **(2)**: 2.5, 2.6, 2.7. Thanks to *SignC* that is verified by *IB*. - In message **(6)**: 6.1. Thanks to *ConfC* that confirms to *P* the authenticity of *C*.
b)	- In message **(2)**: 2.6.
c.a), *c.b*)	- In message **(2)**: 2.1, 2.2, 2.5. Thanks to the *AuthorizReq* containing the *Banking-Data* and which is sent, from *C* to *IB*, enciphered by the *ConfidentialityKey*. Only *IB* can obtain the *Banking-Data* by deciphering *{AuthorizReq}ConfidentialityKey*. - In message **(2)**: The *Banking-Data* are not sent by *C* to *P*, and then in an implicit way, they remain confidential to *P*.
d.b), *e.b*)	- In message **(3)**: 3.3, 3.4. Thanks to *SignP* that is verified by *AB*. - In message **(5)**: 5.2. Thanks to *ConfP* that confirms to *C* the authenticity of *P*.

– **Objective 2**: respect of the same EMV principle as follows:
  - As illustrated in Table 6, our proposals globally use the same security elements provided by the EMV payment system (see Table 2). The 2nd Security Solution *OnC/OffP* uses *CK-SMC*, that is classically provided by EMV protocol, instead of *CK-AC* (please refer to [8])). The 1st Security Solution *OffC/OffP* uses neither *CK-SMC* nor *CK-AC* because these keys are specific for the communication with *IB*. Additionally, our proposals add new *CAp*, *pk(P)/sk(P)* and *CertP* that allow to certify *P* (see sections 2.1 and 2.2).
  - As illustrated in Table 7, our proposals globally perform the same security procedures as in the EMV security protocol. In the latter, each of the *ARQC* and *SignC* allows to authenticate *C* [8]. For our proposals, we have seen that *SignC* is sufficient to authenticate *C*. For the 2nd Security Solution *OnC/OffP*, the *{AuthorizReq}ConfidentialityKey* is needed to encrypt the *Banking-Data*. Then, we replaced the *ARQC* calculation with the *{AuthorizReq}ConfidentialityKey* calculation. In addition, the *ARPC*, in the EMV protocol, allows to respond to *C* and to authenticate *IB*. Consequently, for the 2nd Security Solution *OnC/OffP*, since

we used the *ConfidentialityKey*, then we replaced the *ARPC* calculation with the *{AuthorizResp}ConfidentialityKey* calculation.
- In fact, both the *{AuthorizReq}ConfidentialityKey* and *{AuthorizResp}ConfidentialityKey* are unnecessary in the 1st Security Solution *OffC/OffP*, because the *ConfidentialityKey* is derived from *CK-SMC* which is specific for the communication with *IB*.

– **Objective 3**: effective use for C resources:
- As illustrated in Table 8, there is in the EMV protocol only '1' asymmetric function: *calculating SignC*. For the 2nd Security Solution, we successfully succeeded in avoiding performing in C other asymmetric functions, with the exception of that which is provided by default by EMV. This success is achieved because we have taken advantage of the availability of the internet connection to offload the asymmetric functions, that are supposed to be executed by C, to another actor, such as AB/IB.
- For the 1st Security Solution, since C and P are both in the offline mode, and C needs to authenticate P, then, C has to additionally execute to the old asymmetric function, a new one in order to verify *SignP*. Therefore, in this solution, it was not possible to avoid the execution of new asymmetric cryptographic functions.

Table 6: Security Elements in the EMV Protocol and Our Proposals

	EMV Protocol [8]	1st - *OffC/OffP*	2nd - *OnC/OffP*
Elements Provided by EMV & Used	- *CertIB*. - *CertC*. - *pk(C)/sk(C)*. - *CK-AC*.	- *CertIB*. - *CertC*. - *pk(C)/sk(C)*.	- *CertIB*. - *CertC*. - *pk(C)/sk(C)*. - *CK-SMC*.
Elements Provided by EMV & Not Used	- *CK-SMC*.	- *CK-AC*. - *CK-SMC*.	- *CK-AC*.
Elements Not Provided by EMV: Our Proposed Elements		- New *CAp*. - *CertP*. - *pk(P)/sk(P)*.	- New *CAp*. - *CertP*. - *pk(P)/sk(P)*.

Table 7: Security Procedures in the EMV Protocol and Our Proposals

	EMV Protocol [8]	1st - *OffC/OffP*	2nd - *OnC/OffP*
Used EMV Procedures	- *SignC*. - *AuthorizReq*. - *AuthorizResp*. - *ARQC*. - *ARPC*.	- *SignC*.	- *SignC*. - *AuthorizReq*. - *AuthorizResp*.
Not Used EMV Procedures		- *ARQC*. - *ARPC*.	- *ARQC*. - *ARPC*.
New Procedures: Our Proposed Procedures		- *SignP*.	- *SignP*. - Encipher *AuthorizReq* by *ConfidentialityKey*. - Encipher *AuthorizResp* by *ConfidentialityKey*.

4.2 Performance Evaluation

To the best of our knowledge, our proposals have not been previously proposed with the same ideas and objectives. This make us the first to give better results than the related works as we will discuss in this section. In Table 9, we compare the cost of the cryptographic functions computation, between the existing solutions, EMV protocol and our proposals. In Table 10, we show a robustness comparison. The questions asked in this table are: **Q1**: Is it feasible to actually implement it?, **Q2**: It does not change EMV principle?, **Q3**: Is there an efficient

use for C resources?, **Q4**: Does it resist against replays attacks?, **Q5**: Does it resist against impersonation attacks?, **Q6**: Does it resist against man-in-the-middle attacks?, **Q7**: Does it resist if P was stolen ?

The comparison shows the effectiveness of our proposals as follows: from Table 9, the existing solutions [17], [18], [19] and [20] are faster than our proposals because the use of the symmetric cryptography takes less time compared to the use of the asymmetric cryptography [16]. From Table 10, our proposals satisfy all properties and are totally robust (answers to questions). In addition, as illustrated in Table 10, the solutions [17], [18], [19] and [20] do not satisfy all the security requirements and are not totally robust. The solutions [14] [21], compared to the solutions [17], [18], [19] and [20], are less rapid (see Table 9), satisfy more security requirements and are more robust (see Table 10). Consequently, we can conclude that our proposals are more interesting than the solutions proposed in literature, and especially when we compare the execution time of our proposals to the EMV execution time: they need, in addition to EMV, two asymmetric operations and two hashing functions to authenticate P (see section 4.1).

Table 8: Cryptographic Cost In C

	Symm. Encryption	Symm. Decryption	Asymm. Encryption	Asymm. Decryption	Hash Functions
EMV [8]	2	-	1	-	3
1st OffC/OffP	-	-	1	1	2
2nd OnC/OffP	2	2	1	-	2

Table 9: Cryptographic Cost Comparison

	Symm. Encryption	Symm. Decryption	Asymm. Encryption	Asymm. Decryption	Hash Functions	Exchanged Messages
[14]	6	6	2	2	6	11
[17]	6	6	-	-	9	8
[18]	7	7	-	-	-	7
[21]	10	10	2	2	4	11
[19]	4	4	1	1	3	6
[20]	7	7	-	-	2	8
EMV [8]	*6*	*6*	*1*	*1*	*4*	*9*
1st *OffC/OffP*	-	-	**2**	**2**	**4**	**3**
2nd *OnC/OffP*	**6**	**6**	**2**	**2**	**6**	**6**

Table 10: Robustness Comparison

	[14]	**[17]**	**[18]**	**[21]**	**[19]**	**[20]**	**EMV [8]**	**1st** *OffC/OffP*	**2nd** *OnC/OffP*
a)	√	√	-	√	√	√	√	√	√
b)	√	-	-	-	-	√	√	√	√
c.a)	√	√	√	√	√	√	-	√	√
c.b)	-	-	-	-	-	-	-	√	√
d.a)	√	√	√	√	-	√	√	√	√
d.b)	√	√	√	√	√	√	-	√	√
e.a)	√	-	-	√	-	-	√	√	√
e.b)	√	-	-	√	-	-	-	√	√
Q1	-	-	-	-	-	-	√	√	√
Q2	√	-	-	-	-	-	√	√	√
Q3	-	-	-	√	√	√	√	√	√
Q4	√	√	√	√	√	√	√	√	√
Q5	√	√	√	√	-	√	-	√	√
Q6	√	√	√	√	-	√	-	√	√
Q7	-	-	-	-	-	-	-	√	√

5 Formal Security Analysis using Scyther Tool

The verification of the correctness and soundness of a security protocol has proven to this day to be extremely difficult for humans [22]. Hence, we have chosen to verify our solutions by a verification tool called Scyther that allows formal analysis for security protocols by identifying potential attacks and vulnerabilities (man-in-the-middle, data modification, data insertion, etc.) [23]. To

the best of our knowledge, Scyther is the most efficient tool in terms of simplicity of implementation and verification of security properties. Furthermore, there are many documents in literature to learn how it works [24]. Additionally, Scyther has been successfully used in both research and teaching fields, and in [25] [26], authors prove the performance of Scyther compared to other tools.

			Status		Comments
C	SecondSecurityMechanismOffOff,c1	Nisynch	Ok	Verified	No attacks.
	SecondSecurityMechanismOffOff,c2	Niagree	Ok	Verified	No attacks.
	SecondSecurityMechanismOffOff,c3	Alive	Ok	Verified	No attacks.
	SecondSecurityMechanismOffOff,c4	Weakagree	Ok	Verified	No attacks.
P	SecondSecurityMechanismOffOff,p1	Nisynch	Ok	Verified	No attacks.
	SecondSecurityMechanismOffOff,p2	Niagree	Ok	Verified	No attacks.
	SecondSecurityMechanismOffOff,p3	Alive	Ok	Verified	No attacks.
	SecondSecurityMechanismOffOff,p4	Weakagree	Ok	Verified	No attacks.

Fig. 5: Scyther Results For the 1st Security Solution

			Status		Comments
C	ThirdSecurityMechanismOnOff,c1	Nisynch	Ok	Verified	No attacks.
	ThirdSecurityMechanismOnOff,c2	Niagree	Ok	Verified	No attacks.
	ThirdSecurityMechanismOnOff,c3	Alive	Ok	Verified	No attacks.
	ThirdSecurityMechanismOnOff,c4	Weakagree	Ok	Verified	No attacks.
	ThirdSecurityMechanismOnOff,c5	Secret PAN	Ok	Verified	No attacks.
	ThirdSecurityMechanismOnOff,c6	Secret ExpDate	Ok	Verified	No attacks.
	ThirdSecurityMechanismOnOff,c7	SKR H(k(C,IB),PAN,RC)	Ok	Verified	No attacks.
P	ThirdSecurityMechanismOnOff,p1	Nisynch	Ok	Verified	No attacks.
	ThirdSecurityMechanismOnOff,p2	Niagree	Ok	Verified	No attacks.
	ThirdSecurityMechanismOnOff,p3	Alive	Ok	Verified	No attacks.
	ThirdSecurityMechanismOnOff,p4	Weakagree	Ok	Verified	No attacks.
IB	ThirdSecurityMechanismOnOff,ib1	Nisynch	Ok	Verified	No attacks.
	ThirdSecurityMechanismOnOff,ib2	Niagree	Ok	Verified	No attacks.
	ThirdSecurityMechanismOnOff,ib3	Alive	Ok	Verified	No attacks.
	ThirdSecurityMechanismOnOff,ib4	Weakagree	Ok	Verified	No attacks.
	ThirdSecurityMechanismOnOff,ib5	Secret PAN	Ok	Verified	No attacks.
	ThirdSecurityMechanismOnOff,ib6	Secret ExpDate	Ok	Verified	No attacks.
	ThirdSecurityMechanismOnOff,ib7	SKR H(k(C,IB),PAN,RC)	Ok	Verified	No attacks.

Fig. 6: Scyther Results For the 2nd Security Solution

Indeed, Scyther allows to analyze security protocols thanks to 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. Consequently, in order to implement our proposals in Scyther, the Security Protocol De-scription Language (SPDL) is used [26]. In this language, each actor is either written in a Scyther role or is declared as a Scyther Agent. In our work [8], we have given definitions of the Scyther claims/roles/agents. Therefore, we have used Scyther in our solutions as follows:

– For the two solutions: we have used the following Scyther claims to refer to the targeted security properties discussed in section 2.3:

  - *Nisynch, Niagree, Alive, Weakagree*: for the integrity of the *Banking-Data* (*PAN*/*ExpDate*) and the authentication/non-repudiation of *C*, *P*: {$\underline{a}$), $\underline{d.a}$), $\underline{d.b}$), $\underline{e.a}$), $\underline{e.b}$)}.

- *Secret*: for the confidentiality of the *Banking-Data* (*PAN*/*ExpDate*): {*c.a*), *c.b*)}.

– Only for the 2nd Solution: we have used the following Scyther claim:

- *SKR*: for the confidentiality of the *ConfidentialityKey* that is encoded in scyther by *H(k(C,IB), PAN, RC)* where *k(C,IB)* represents *CK-SMC*.

– For the 1st Solution:

- Scyther Roles/Agents: we have implemented Scyther roles for *{C, P}*, and for {*AB, IB, PS, CAp*}, we have used Scyther agents.
- Scyther Results: as illustrated in Fig. 5, the 1st Security Solution successfully guarantees all the Scyther claims for {*C, P*} and no attacks are found.

– For the 2nd Solution:

- Scyther Roles/Agents: we have implemented the Scyther roles for *{C, P, IB}*, and for {*AB, PS, CAp*}, we have used the Scyther agents.
- Scyther Results: as illustrated in Fig. 6, the 2nd Security Solution successfully guarantees all the Scyther claims for {*C, P, IB*} and no attacks are found.

6 Conclusion

In this paper, we proposed new security solutions aiming to overcome *two security vulnerabilities* that have been detected in the classical EMV payment protocol. According to our previous study in [1], these two EMV vulnerabilities represent major risks for our day to day safety. The idea of our solutions is to improve the security of the classical EMV protocol by solving these two vulnerabilities where *P* machines are in the "offline" connectivity mode. Consequently, our proposals allow ensuring all the targeted security properties presented in Table 4 and are totally robust compared to the other solutions proposed in literature. They are also verified correctly thanks to the Scyther security tool.

References

1. El Madhoun, N., Bertin, E., Pujolle, G.: The emv payment system: Is it reliable? 2019 3rd Cyber Security in Networking Conference (CSNet) pp. 80–85 (2019)
2. EMV - Integrated Circuit Card Specifications for Payment Systems,: Book 1: Application Independent ICC to Terminal Interface Requirements, Version 4.3, EMVCo, November (2011)
3. EMV - Integrated Circuit Card Specifications for Payment Systems,: Book 2: Security and Key Management, Version 4.3, EMVCo, November (2011)
4. EMV - Integrated Circuit Card Specifications for Payment Systems,: Book 3: Application Specification, Version 4.3, EMVCo, November (2011)
5. EMV - Integrated Circuit Card Specifications for Payment Systems,: Book 4: Cardholder, Attendant, and Acquirer Interface Requirements, Version 4.3, EMVCo, November (2011)

6. De Ruiter, J., Poll, E.: Formal analysis of the emv protocol suite. Springer Theory of Security and Applications pp. 113–129 (2012)
7. van den Breekel, J., Ortiz-Yepes, D.A., Poll, E., de Ruiter, J.: Emv in a nutshell. Technical Report (2016)
8. El Madhoun, N., Bertin, E., Badra, M., Pujolle, G.: Towards more secure emv purchase transactions. Annals of Telecommunications **76**(3), 203–222 (2021)
9. Basin, D., Sasse, R., Toro-Pozo, J.: The emv standard: Break, fix, verify. 2021 IEEE Symposium on Security and Privacy (SP) pp. 1766–1781 (2021)
10. Emms, M., van Moorsel, A.: Practical attack on contactless payment cards. HCI2011 Workshop Heath, Wealth and Identity Theft (2011)
11. Lifchitz, R.: Hacking the nfc credit cards for fun and debit. Hackito Ergo Sum conference (April 2012)
12. Benjamin Cohen: Millions of barclays card users exposed to fraud (2012), last connection (24 December 2021)
13. Gerard Tubb: Contactless cards: App reveals security risk. https://news.sky.com/story/contactless-cards-app-reveals-security-risk-10443980 (2013), last connection (24 December 2021)
14. Emms, M.J.: Contactless payments: usability at the cost of security? Ph.D.Thesis, Newcastle University (2016)
15. Elminaam, D.A., Kader, H.A., Hadhoud, M.M.: Performance evaluation of symmetric encryption algorithms. Communications of the IBIMA **8**, 58–64 (2009)
16. Badra, M., Badra, R.B.: A lightweight security protocol for nfc-based mobile payments. Elsevier, Procedia Computer Science **83**, 705–711 (2016)
17. Thammarat, C., Kurutach, W., Phoomvuthisarn, S.: A secure lightweight and fair exchange protocol for nfc mobile payment based on limited-use of session keys. 17th International Symposium on, Communications and Information Technologies (ISCIT), IEEE pp. 1–6 (2017)
18. Ceipidor, U.B., Medaglia, C.M., Marino, A., Sposato, S., Moroni, A.: Kernees: A protocol for mutual authentication between nfc phones and pos terminals for secure payment transactions. International ISC Conference on Information Security and Cryptology (ISCISC), IEEE pp. 115–120 (2012)
19. Lee, Y.S., Kim, E., Jung, M.S.: A nfc based authentication method for defense of the man in the middle attack. Proceedings of the 3rd International Conference on Computer Science and Information Technology pp. 10–14 (2013)
20. Al-Tamimi, M., Al-Haj, A.: Online security protocol for nfc mobile payment applications. 8th International Conference on Information Technology (ICIT), IEEE pp. 827–832 (2017)
21. Pourghomi, P., Ghinea, G., et al.: A proposed nfc payment application. International Journal of Advanced Computer Science and Applications **12**, 173–181 (2013)
22. Basin, D., Cremers, C., Meadows, C.: Model checking security protocols. Handbook of Model Checking, Springer pp. 727–762 (2018)
23. Cremers, C.J.: The scyther tool: Verification, falsification, and analysis of security protocols. Springer Computer Aided Verification (2008)
24. Kahya, N., Ghoualmi, N., Lafourcade, P.: Formal analysis of pkm using scyther tool. International Conference on Information Technology and e-Services, IEEE pp. 1–6 (2012)
25. Cremers, C., Lafourcade, P.: Comparing state spaces in automatic protocol verification. International Workshop on Automated Verification of Critical Systems (AVoCS) (2007)
26. Cremers, C., Mauw, S.: Operational semantics and verification of security protocols. Springer Science & Business Media (2012)