FEIPS - a Secure Fair-Exchange Payment System for Internet Transactions

Zoran Djuric[1*], Dragan Gasevic[2]
[1]*Faculty of Electrical Engineering, University of Banjaluka, Patre 5, 78 000 Banjaluka, Bosnia and Herzegovina*
[2]*School of Computing and Information Systems, Athabasca University, Athabasca, Canada*

*Corresponding author: zoran.djuric@etfbl.net

ABSTRACT

In order to be considered secure, a payment system needs to address a number of security issues. Besides fundamental security requirements, like confidentiality, data integrity, authentication and non-repudiation, another important requirement for a secure payment system is fair exchange. Many existing payment protocols require that customers must pay for products before their delivery (in the case of delivery of digital goods) or the delivery of the receipt (in the case of delivery of physical goods). This unfair situation should be eliminated afterwards, that is, it is necessary to rebalance fairness for customers. In order to address these issues, we propose the Fair Exchange Internet Payment Protocol (FEIPS). The FEIPS protocol is designed for the payment of physical goods and falls into the category that uses a trusted third party for assuring fair-exchange. Although FEIPS has a strong emphasis on fair exchange, it still guarantees strong security properties, including confidentiality, data integrity, authentication and non-repudiation. The FEIPS protocol is designed to be simple and practical, unlike other similar protocols designed for the payment of physical goods. To demonstrate that FEIPS satisfies the desired properties, we perform a formal verification using the HLPSL language and the AVISPA tool.

KEY WORDS: payment system, fair exchange, security, cryptography, e-commerce, formal verification

1. INTRODUCTION

E-commerce systems need to address a number of security issues in order to be an effective and secure means of transferring payments across the Internet. In general, an e-commerce transaction can be defined as a process of exchanging goods of value [1]. For example, this can be a process of exchanging physical or digital goods for money.

It is generally accepted that, in order to be considered secure, a payment system must satisfy the following fundamental security requirements [2-7]: *confidentiality* (*secrecy*), *data integrity*, *authentication* and *non-repudiation*. Also, another important security requirement is fair exchange [1, 8, 9]. It is crucial to ensure that e-commerce transactions (e-cash payment, contract signing, etc.) are fair. While ensuring fundamental security requirements across open networks has been studied intensively, designing a fair exchange protocol has become an active research topic in recent years. Fair exchange is a process that guarantees that, at the end of an exchange process, all the involved parties obtain the items they expect or neither one do [1, 10-15]. It means that an e-commerce participant does not have any advantage over the other participants, at the end of an e-commerce protocol. For example, a fair exchange e-payment protocol should make it impossible for a customer to receive the ordered goods (digital goods or a digital receipt for physical goods) while a merchant does not receive the customer's payment, and vice versa.

To better illustrate the need for fair exchange, let us consider a typical e-commerce process (Figure 1). This process is very similar to the e-procurement process described in [16]. This typical e-commerce process can be divided into two phases: ordering and fulfillment. In the ordering phase, the customer and the merchant agree on purchase details. In the second phase (fulfillment phase), the customer makes the agreed payment to the merchant and the merchant delivers the ordered items to the customer. Obviously, exchanges of items during all phases of e-commerce process should be performed fairly. One of the goals that are necessary to accomplish during an e-commerce process is to prevent a situation in which one of the participants does have an advantage over the other. Also, it is necessary to provide all participants with sufficient transaction evidence that can be used in a dispute resolution that may arise afterwards.

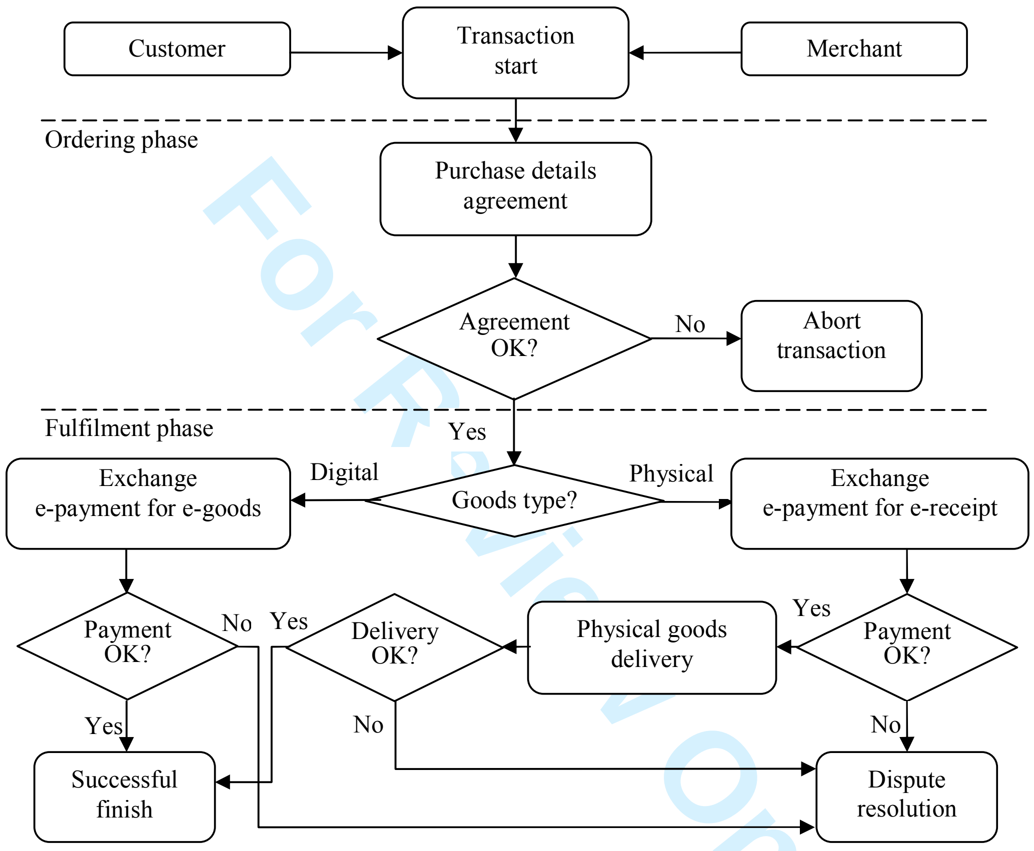

FIGURE 1. A TYPICAL E-COMMERCE PROCESS

Making sure that a protocol addresses all (or some) of the aforementioned security requirements is not a trivial task. Security flaws in many systems have been discovered some time after their proposal [17-19]. In order to make sure that a protocol is secure, one needs to formalize both the protocol and the security requirements that it needs to meet, and then verify that the requirements are met indeed. Fundamental security requirements are security requirements for which the Dolev-Yao intruder is the most powerful intruder [20], under certain assumptions, such as perfect cryptography.

In this paper, we present an extended version of the IPS protocol described in [21]. This extended protocol is called FEIPS (Fair Exchange IPS). As it is not designed with fairness in mind, the unfair merchant in the IPS protocol can violate the fairness for the customer (Section 5.3). This is why the FEIPS protocol tries to address the fair-exchange requirements, besides fundamental security requirements. The fair-exchange requirements are explained in Section 2. In Section 3, we discuss some of the existing fair exchange protocols. Because the FEIPS protocol is designed for the payment of physical goods, an analysis of similar protocols is of special interest for this paper. The original IPS protocol and the IPS system (which implements IPS protocol) are briefly described in Section 4. Also, the fair exchange problem in IPS protocol is described. Building upon the analysis of the related work (Section 3), especially on the analysis of e-payment protocols for physical delivery, and the original IPS protocol (Section 4), we define the FEIPS protocol in detail in Section 5. The specification, analysis and verification of the FEIPS are given in Section 6. As fair-exchange is one of most important properties in e-payment protocol, it is important to verify it through informal and formal analysis approach [9]. This is why discussion of fair-exchange in the FEIPS protocol is given in Section 7. Before concluding the paper, comparison against e-payment protocols that are designed for the payment of physical goods is given in Section 8.

2. FAIR EXCHANGE REQUIREMENTS

According to [1, 10-12, 20, 22, 23], a fair exchange protocol is a protocol satisfying the following requirements: *effectiveness*, *fairness*, and *timeliness*.

Effectiveness means that if participants behave according to the protocol and none of them wants to terminate a protocol execution, at the end of the protocol execution, the participants will obtain the items they expect, without the intervention of a trusted third party (TTP) [24-26]. TTP is introduced in protocols to support their execution and to help protocol participants with the process of exchanging goods.

Fairness means that at the end of the protocol execution either both protocol participants obtain expected goods, or neither of them obtains any useful information about the goods of the other participant [27]. According to this, if at least one participant does not behave according to the protocol, then no honest participant wins or loses anything valuable. Some authors [13, 14, 22] distinguish between strong and weak fairness. Strong fairness is the fairness described previously in this paragraph, while weak fairness means that either strong fairness is achieved or it is possible for a participant to prove to an outside party that an unfair situation has occurred [14, 22].

Timeliness means that the protocol will eventually terminate for all participants (that behave according to the protocol) and that after the termination point the degree of achieved fairness will not change (i.e., after the termination point no party can gain any advantage). A participant who behaves according to the protocol will eventually reach either a "success" or an "abort" termination state. Also, timeliness guarantees that at least one party has the ability to abort the normal protocol execution, and execute a resolution sub-protocol that will be executed in a finite amount of time [28].

3. LITERATURE REVIEW

In general, there are two main categories of fair exchange protocols [1, 12, 14, 29, 30]: protocols with two participants (the exchange of goods is done directly between participants) and TTP-based protocols. Protocols with two participants can be divided into: gradual secret release protocols [31-35] and probabilistic protocols [29, 36-39]. Although two-party protocols offer a potentially attractive solution for fair exchange as they do not require the involvement of a TTP, there is an important drawback of these protocols – it has been formally proven that it is impossible to achieve fair exchange without a TTP [37, 40].

TTP-based protocols are relaying on TTPs to help with the exchange process. TTP-based fair exchange protocols can be divided into [1, 41-44]: fair exchange protocols with inline

TTPs, fair exchange protocols with online TTPs, and fair exchange protocols with offline TTPs. Using a TTP in every step of the protocol (inline TTP) or in almost every step of the protocol (online TTP) is inefficient. These two types of TTP's are susceptible to denial-of-service (DoS) attacks (i.e., these types of TTP's are a single point of failure in terms of performance and security) [14, 15]. Additionally, it is important to notice that inline TTP may give an unfair advantage to one of the parties involved in the protocol execution [14, 15].

To avoid bottlenecks, the optimistic approach to fair exchange is introduced [12, 22]. Optimistic fair exchange protocols act on the assumption that the protocol participants will not misbehave. They are using offline TTPs. Such TTPs are used only in a case of session recovery or abortion (which is assumed to be infrequent). Optimistic fair exchange protocols consist of at least the following two sub-protocols [45-49]: the normal exchange sub-protocol and the recovery sub-protocol. During the first one, participants attempt to exchange desired items by themselves, without any intervention of a TTP. In a case of a dispute, the recovery sub-protocol is invoked. The TTP then recovers the disputed items and restores fairness. In other cases the offline TTP does not take part in a protocol execution. A dispute between protocol participants can be caused by a protocol participant's misbehaviour or a communication link failure. An offline TTP is first time used in the design of a fair e-commerce protocol in 1990 [50].

The NetBill protocol [51] is one of the earliest protocols to provide a complete solution to the problem of selling and delivering electronic goods over the network. The NetBill system uses a TTP called the NetBill server. The NetBill server is an account server which maintains accounts for both customers and merchants. This protocol ensures money and goods atomicity, but it does not ensure true fair exchange, since a merchant who has provided a worthless good is detected only after the transaction completes.

In [30], a true fair exchange protocol based on the theory of cross validation is proposed. In this protocol, each participant can verify that the item they are about to receive is indeed the expected one. It is important to notice that the merchant escrows the encrypted product and a pair of keys with the TTP. Thus, if the merchant disappears after receiving the payment, the TTP can give the customer the key for decrypting the product. However, the private key of the key pairs of both the customer and merchant are available with the bank. Although the authors of this protocol claim that reliance on TTP is minimal, the generation of asymmetric keys for both merchant and customer, as well as encryption of the merchant's goods is all performed by the TTP.

Asokan [22] proposed a generic optimistic fair exchange protocol that is suitable for exchanging signatures, confidential data or payments. They claim that the proposed protocol is a first fair exchange protocol of generic items which are any digital products such as confidential data or payments. In the Asokan protocol [22] parties are able to validate the item only after receiving it, while the Ray et al. protocol [30] allows each party to verify whether the item that they are going to receive is the correct item or not before they receive it.

In [52], the EMH (Enforce Merchant Honesty) e-commerce protocol for exchanging digital products and payments is proposed. The main idea of this protocol is to have one trustworthy party (the customer) and enforce the other party (the merchant) to be honest. This protocol consists of three messages to be exchanged between the customer and the merchant. The customer starts the exchange by sending a description of the product and an encrypted payment. As a response to the first message, the merchant sends an encrypted digital product to the customer. Upon receiving the merchant's product, the customer verifies it. If the customer is satisfied with the product, the customer sends the decryption key to the merchant, so that the merchant can decrypt the payment. Although, the protocol guarantees strong fairness for both the customer and the merchant, it is important to notice that it is based on the assumption of the trustworthiness of the customer, which is unlikely to happen in a real environment.

Because the IPS/FEIPS protocol is designed for the payment of physical goods, an analysis of similar protocols is of special interest for this paper.

Zhang et al. [8] proposed a practical fair exchange e-payment protocol for anonymous purchase and physical delivery which applies the principle of true fair-exchange to the process of purchase and physical delivery via an e-commerce system without the involvement of TTP. This protocol consists of 7 messages. The authors of this protocol claim that this protocol ensures anonymity of both the customer and the merchant, and protects the customer's identity, sensitive information and spending behavior. Additionally, they claim that protocol successfully conceals the identity of either customer or merchant during the transaction even under all possibly potential collusion by various parties. Still, this protocol, with the concept of a delivery cabinet [8] seems complicated and potentially impractical. In this protocol, every customer and merchant must have an asymmetric key pair issued by their bank, which is associated with their bank account. This means that banks must maintain a very large database of customers' and merchants' public keys associated with their bank accounts. Next, the protocol requires the customer to send two payments for one transaction. One payment will be sent to the merchant, while the other payment will be sent to the customer's bank, encrypted with a second key what bears a mathematical relation with the first key received by the merchant. Additionally, according to [23] there is a limitation in the fairness of the protocol which can bring the merchant in an advantage. Although the authors of that protocol mentioned that formal methods of software verification including model checking and theorem proving could be employed [8], security requirements for this protocol (including fairness) are not formally verified.

Li et al. [53] proposed an e-commerce fair exchange protocol for the exchange of payment and physical product without the involvement of TTP. The protocol involves the bank (where all parties have their accounts) and the delivery agent that will deliver the physical product to the customer. The protocol consists of eight messages. The two parties will start the protocol by exchanging their signatures. After that, the delivery agent will deliver the physical product to the customer. If the customer is satisfied with the product then he will release the key to the delivery agent. Then, the physical product will be handed to the customer. The customer will then send a signed receipt to the delivery agent. The delivery agent will forward the key and the signed receipt to the merchant. The merchant will forward the key and the signed receipt to the bank, and the bank will finally transfer the amount from the customer's account to the merchant's account. There are some important issues with this protocol. It is not clear how the protocol will work if the two parties have accounts at different banks, because it is required that the customer and the merchant have their accounts at the same bank. Similar to the Zhang et al. [8] protocol, this protocol, with the concept of a delivery agent, also seems complicated and potentially impractical. Additionally, security requirements for this protocol (including fairness) are not formally verified.

Alaraj [44] proposed an offline TTP-based fair exchange protocol which allows customers to buy physical products from merchants online. The protocol involves the bank and the delivery agent that will deliver the physical product to the customer. It consists of two subprotocols: the exchange subprotocol (which consists of 6 messages) and the resolution subprotocol (which consists of 2 messages). Although the author of this protocol claim that the protocol ensures fairness for both customer and merchant, it is important to notice that this protocol is not formally verified, i.e. it is not formally verified that this protocol satisfies fundamental security requirements, including fairness. Specially, it is not known how the payment is signed by the customer in the first message of the exchange subprotocol because the customer does not have an asymmetric key pair issued [44].

Classical e-payment protocols for credit card payments, like SET [5, 19, 54, 55], Visa 3-D Secure [55, 56], and SSL/TLS [57] establish secure communication over insecure networks in order to ensure that no attacker can obtain sensitive information or impersonate some regular protocol participant. These protocols are protecting protocol participants, who trust one another, from potential attackers. It is obvious that this scenario is inappropriate when one protocol participant does not even know the other protocol participant. Also, it has been well documented [58, 59] that lack of trust between protocol participants can restrict the potential of e-commerce. Preliminary registration is an attempt to strengthen the trust [27]. This is why customers who wish to participate in the protocol must first enroll with an authority. Classical e-payment protocols that employ registration include SET [5, 19, 54, 55] and Visa 3-D Secure [55, 56]. Registration gives the customer some confidence in the merchant, and vice versa, but it does not change the security framework. The protocol participants must still trust each other [27]. This is why these protocols do not achieve the fairness when, for example, the merchant misbehaves. The same is with modified versions of classical e-payment protocols, for example with improved SET protocol [60].

The situation is similar with the IPS protocol that we proposed in [21] and that belongs to the same category as classical e-payment protocols for credit card payments, like SET and VISA 3-D Secure. As it is not designed with fairness in mind, it is possible to violate the fairness for the customer. Before this deficiency is described (Section 4.3), a short description of the original IPS payment system, including description of the original IPS protocol, is given in the following section.

Building on the findings of the analyzed literature above, we set the design goals for the FEIPS protocol in order to address the identified research gaps. The FEIPS design goals include all the original IPS design goals (more information about the IPS design goals can be found in [21]). Regarding security, we demand that the FEIPS protocol must satisfy fundamental security requirements: confidentiality (secrecy), data integrity, authentication, and non-repudiation. Also, there is one additional requirement – the FEIPS protocol must be fair (unlike SET and VISA 3-D Secure), i.e. it must provide fair exchange for both the customer and the merchant, even when one of them is misbehaving (which is likely to happen in a real environment). Besides these requirements, the FEIPS protocol should have minimal reliance on TTP (unlike protocol described in [30]), it should be simple, practical and easy to use for secure credit and debit card transactions (unlike the Zhang et al. e-payment protocol [8] and the Li et al. protocol [53]), while the customer and/or the merchant should not be considered trustworthy participants (unlike the EMH protocol [52]). Contrary to the Zhang et al. [8] and Li et al. [53] e-payment protocols for physical delivery, the FEIPS protocol should not be complicated and impractical. Additionally, all additional design goals from the original IPS protocol are retained – the FEIPS protocol should: use strong cryptography and authenticity checking models, prevent the merchant from seeing payment information, prevent card issuers from seeing order information (i.e., protecting merchants' privacy), and free customers from any form of preliminary registration (unlike the Zhang et al. [8] and Li et al. [53] e-payment protocols). Specifically, this last requirement means that customers are not required installing any additional software for secure payments, and they are not required to have a digital certificate issued from a certificate authority. It is important to mention that the FEIPS requirements are in accordance with the desired security properties of any e-commerce protocol [59]. A detailed description of the FEIPS protocol is given in Section 5.

4. THE ORIGINAL PROTOCOL - IPS

The principal participants in IPS, as well as in FEIPS (as shown in Section 5), are: the customer (the payment Web segment), the merchant, the merchant Web shop, the payment gateway, and the bank (Figure 2). The "merchant" is a piece of software running on the merchant's server. The payment Web segment is a digitally signed piece of software running on the customer's computer, automatically downloaded from the merchant server prior to the protocol execution. The payment Web segment handles all the protocol interaction on the customer's side; for the sake of simplicity we will then use the term "customer" interchangeably with "payment Web segment". It should be clear from the context which one of the two we are referring to. In addition, we will abstract away from the details of the connection between the merchant and the merchant Web shop and view them as being the same.

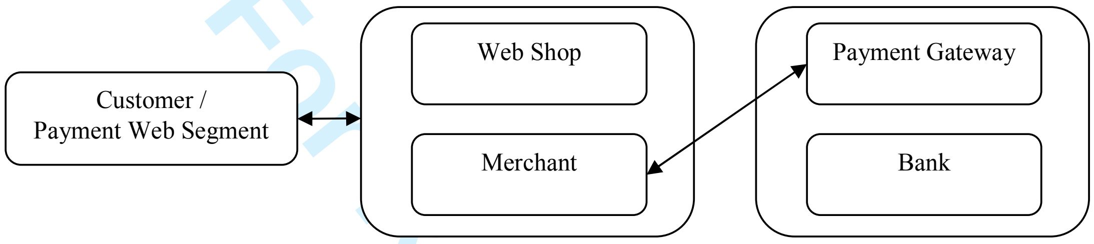

FIGURE 2. THE PRINCIPAL PARTICIPANTS IN IPS AND FEIPS

Because one of the main IPS design goals was to create a protocol in which the customers will not be required to perform any kind of registration before they may engage in IPS transactions (although a prior registration is still an option), there are two scenarios of the IPS protocol [21]. In this section, similarly as in [21], we discuss only the first scenario of the protocol. In this scenario (the more complicated one) of the IPS protocol, customers are freed from preliminary registration, i.e. they are not required to have a digital certificate issued from an IPS Certificate Authority [21].

The communication between the participants goes through two phases: the preparation phase and the IPS protocol phase.

4.1 THE PREPARATION PHASE

The IPS preparation phase consists of two sub-phases, the ordering sub-phase and the payment Web segment download sub-phase. During the ordering sub-phase, the customer searches the merchant's e-commerce website and finds the products they would like to buy, adds them to the shopping cart, and then initiates (e.g., through a check-out button) the payment Web segment download sub-phase. The payment Web segment is digitally signed by a trusted party (the payment gateway) to prevent any possible modifications to it by the merchant.

The payment Web segment asks the customer for their credit card information and a challenge code which will authenticate and authorize the customer to use the credit card in question [21]. The payment Web segment creates a self-signed customer digital certificate, based on a previously created public/private key pair. As this digital certificate is not signed by any CA, it is obvious that at this stage, the protocol does not provide any assurance of the identity of its owner. The ownership of this digital certificate can only be established afterwards, during the IPS protocol execution. After the ownership is established, the certificate can later be used in a dispute resolution in a similar manner to the digital certificates issued by a CA. It is important to mention that this certificate is temporary and it is used for one session only. In addition, this process does not require any kind of customer interaction or intervention, as all of this is done automatically by the payment Web segment.

After this, the IPS protocol starts to execute.

4.2 THE IPS PROTOCOL DESCRIPTION

The IPS protocol consists of six messages, as shown in Figure 3. In the first message of the IPS protocol, a temporary customer certificate is transmitted from the customer to the merchant. The second message represents the merchant's response to the first message. The merchant's response contains a freshly generated unique ID for the session; this ID is also used to identify the transaction. In the next step (i.e., third message), a payment message (containing the relevant payment information based on the user input) and a signature of the order are prepared. The payment message is encrypted using the payment gateway's public key. This prevents the merchant from seeing the payment information. In the fourth message, the payment message is forwarded to the payment gateway, since the merchant cannot decrypt it. The merchant also includes the customer's certificate in the fourth message.

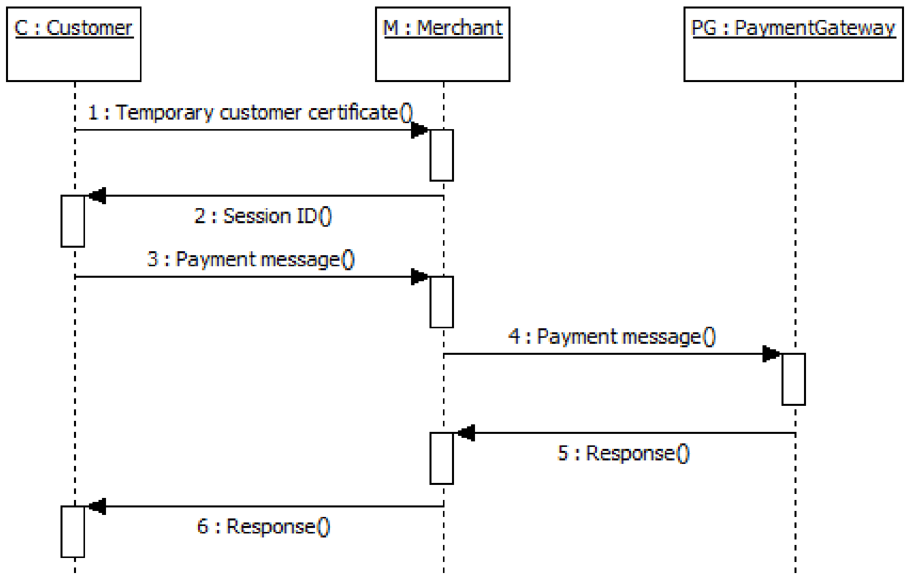

FIGURE 3. MESSAGE EXCHANGE IN THE IPS PROTOCOL

Upon receiving the payment message, the payment gateway decrypts it, checks the signature of the payment information, and checks if the customer is authorized to use the credit card in question. The authorization of the customer is performed based on the combination of the credit card number and the challenge code. If this combination is valid, the customer is authorized to use the credit card in question. This also means that the ownership of the temporary certificate is established (i.e. we can say that the owner of the certificate is the owner of the credit card). The challenge code can be any sort of secret code issued by the bank. For example, the credit card PIN code can be used. In the rest of the paper, we refer to this code simply as PIN. In the fifth message of the IPS protocol, the payment gateway sends a response to the merchant, and the merchant forwards the response to the customer (message 6).

4.3 IPS PROTOCOL AND FAIR-EXCHANGE

As it is already mentioned, the IPS protocol is designed for the payment of physical goods. It is clear that the fair-exchange problem in a case of this protocol (and similar ones) is a problem of exchange of electronic money for a digital receipt. This digital receipt can be used as a proof of a payment for ordered products.

As it can be seen from the IPS protocol description (Section 4.2), fairness can be violated only for the customer. There are two reasons for this. First, the exchange of the messages in the IPS protocol is done in such a way that the merchant is involved in every message exchange, i.e. the customer and the payment gateway do not communicate directly. Second, the digital receipt for the customer (messages 5 and 6) is returned after the payment process is finished.

The design of the IPS protocol, especially the above two reasons, always gives the advantage to the merchant. This advantage should be eliminated afterwards, that is, it is necessary to rebalance fairness for the customer. Similar to IPS, many existing payment protocols require that customers must pay for products before their delivery (in the case of delivery of digital goods) or the delivery of the receipt (in the case of delivery of physical goods) [5, 19, 54-56]. This is why these protocols have the same type of violated fairness problem as IPS has. This is especially true for the SET protocol, because the sequence of messages exchanged between participants is the same as in the case of the IPS protocol. Thus, all observations stated in this section are also valid for the SET protocol.

The most obvious violation of the fairness for the customer occurs in a situation when the merchant does not send message 6 to the customer (Section 4.2, Figure 3). This situation can occur when the merchant does not behave according to the protocol, or in a case of a network connection failure. In such situations, the customer will not receive the response message (message 6). A similar situation can occur if the merchant does not receive message 5. As in the situation when the customer does not receive the response message, the payment process is already finished and the fairness for the customer is unbalanced. As a consequence of the unbalanced fairness in these situations, the customer can initiate a new order of the same goods, or the customer can initiate a new payment of already paid products.

The protocol execution can also be terminated if a network connection failure occurs or if a dishonest merchant does not want to forward message 4 to the payment gateway. In both the situations, the merchant has the payment message and can forward it to the payment gateway afterwards, which will result in a possible money transfer to the merchant's account. As in the previous situations, the customer will not receive the response message and the fairness for the customer is unbalanced again.

Also, it is obvious that the fairness can not be unbalanced for the customer, if a protocol execution is terminated before the merchant receives message 3. The reason for this is that the merchant did not receive the payment message and, obviously, the merchant can not forward it to the payment gateway. This is why the payment gateway can not initiate the payment process and the merchant can not gain the advantage over the customer.

5. THE MODIFIED IPS PROTOCOL - THE FEIPS PROTOCOL

As the FEIPS is based on IPS, it has the same protocol participants: the customer (the payment Web segment), the merchant, the merchant Web shop, the payment gateway, and the bank (Figure 2). The communication between the participants also goes through two phases: the preparation phase and the FEIPS protocol phase. The most important difference is that the FEIPS protocol consists of three subprotocols: the FEIPS setup subprotocol, the FEIPS payment subprotocol and the FEIPS resolution subprotocol. The other assumptions and restrictions defined for the original IPS protocol also hold for the FEIPS protocol (Section 4). Before introducing the FEIPS protocol, the notation used in the description is summarized in Table I.

TABLE I. NOTATION USED IN THE PROTOCOL DESCRIPTION

Notation	**Description**
$\{m\}_k$	Hybrid encryption of the message *m* with the key *k*
PubKA	Party *A*'s public key
DSigA(m)	Digital signature of the message *m* by party *A*
A -> B: m	Message *m* sent from party *A* to party *B*

As it can be seen, the standard Alice-Bob notation is used, in which the sender and the receiver are noted first, followed by the contents of the message. Hybrid encryption of a message *m* with the key *k* means that the message *m* is encrypted using a symmetric session key *s*, which is in turn encrypted using an asymmetric key *k* (the digital envelope). The session key *s* is then used for the remainder of the session between the two parties. For example, the message:

A -> B: ({m}PubKB)

actually means:

A -> B: ({Kab}PubKB, {m}Kab)

and Kab is used for the encryption of all further messages between A and B during the session.

The FEIPS protocol message exchange is shown in Figure 4. The FEIPS setup subprotocol consists of two messages (message 1 and message 2), the FEIPS exchange subprotocol consists of four messages (message 3 to message 6), while messages 7 and 8 represent the FEIPS resolution subprotocol.

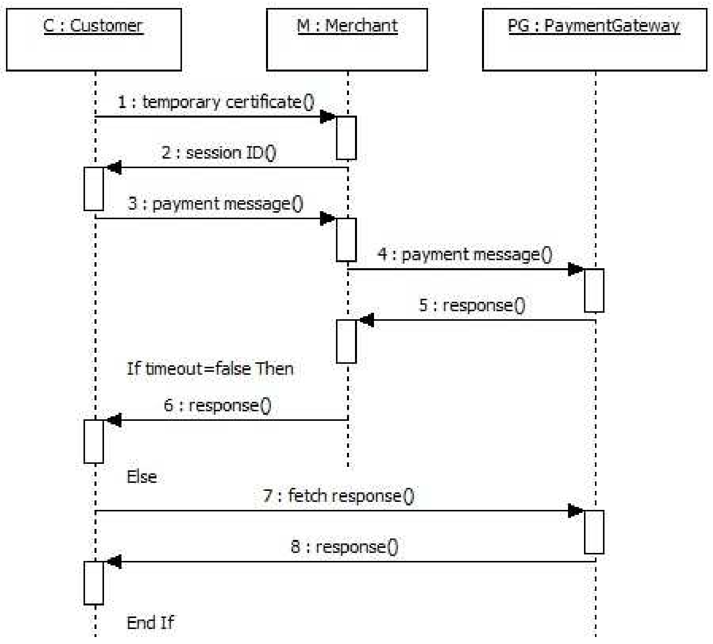

FIGURE 4. MESSAGE EXCHANGE IN THE FEIPS PROTOCOL

Now, each of these eight messages of FEIPS is described in detail, using the Alice-Bob notation.

5.1 FEIPS SETUP SUBPROTOCOL

Message 1: C -> M: ({PubKC}PubKM)

In this message, the temporary customer certificate (denoted as PubKC) is transmitted from the customer to the merchant. The temporary certificate is encrypted with the merchant's public key (denoted as PubKM), using the previously mentioned hybrid encryption.

Message 2: M -> C: ({Sid, DSigM(Sid)}PubKC)

This message represents the merchant's response to the first message. In this message, Sid is a freshly generated random number, which serves as an ID for the session. This session ID is also used as the transaction identifier, and should be unique for each session. Sid and its digital signature (to confirm the merchant's identity) are encrypted using the customer's public key received in message 1.

5.2 FEIPS EXCHANGE SUBPROTOCOL

Message 3: C -> M: {PM, OISig}PubKM

In this step (message 3), based on the user input (CardInf), the payment Web segment generates the relevant payment information PI, including the session ID, the amount (Amount), a fresh nonce for the payment gateway (NonCPG) and the merchant's name M:

PI = (CardInf, Amount, PIN, Sid, NonCPG, M).

Then, the payment message PM is prepared:

PM = {PI, DSigC(PI)}PubKPG.

The payment message (PM) consists of the payment information and the digital signature of the payment information. The whole message is encrypted using the public key of the payment gateway. This prevents the merchant from seeing the credit card information. Additionally, the customer gathers the relevant order information (OI):

OI = (OrderDesc, Amount, Sid).

Then, the customer digitally signs the order information:

OISig = DSigC(OI).

Since the merchant also knows the order description (OrderDesc), the amount (Amount) and session ID (Sid), the merchant can generate OI as well. This is how the merchant can make sure that the customer signed the same order information. If this is the case, the merchant then stores this digital signature along with the order information itself. This can later be used as a proof to verify what the customer has ordered, as is described in Section 6.3. Now, the merchant starts to communicate with the payment gateway, through message 4.

Message 4: M -> PG: ({PM, PubKC, DSigM(Amount, PubKC, Sid)}PubKPG)

In message 4, the payment message PM is forwarded to the payment gateway, since the merchant cannot decrypt it. The merchant also includes the customer's temporary certificate in this message. The customer's temporary certificate, the amount and the session ID are digitally signed to confirm the merchant's identity.

Upon receiving the payment message, the payment gateway decrypts it using its secret key. It checks the signature of the payment information, and checks if the customer is authorized to use the credit card in question. The authorization of the customer is performed based on the combination of the credit card number and the challenge code, as previously explained in Section 4.2. A check is also made to make sure that both the merchant and the customer agree on the amount to be charged. Finally, the payment gateway also checks whether the combination of Sid and NonCPG is fresh, to eliminate possible replay attacks by malicious merchants. Additionally, the payment gateway also checks the freshness of the temporary certificate, to eliminate possible replay attacks by malicious merchants. If everything is correct, the credit card information and the amount are forwarded to the bank. The bank checks whether there is enough money on the customer's account. If so, the transfer is made to the merchant's account and the bank returns a positive response; otherwise a negative response is returned. Additionally, the payment gateway stores the details of the transaction (including the customer's temporary certificate) in its database. Then, the payment gateway forwards the response to the merchant.

Message 5: PG -> M: {Response, DSigPG(Response, Sid, Amount, NonCPG)}PubKM

The response, along with all the other information (Sid, Amount, and NonCPG), is digitally signed, so that the customer can verify if the response has come from the payment gateway and if it is related to the current transaction (i.e., that it is not replayed by the merchant). The merchant forwards the contents of this message to the customer, this time encrypted with the customer's public key.

Message 6: M -> C: {Response, DSigPG(Response, Sid, Amount, NonCPG)}PubKC

5.3 FEIPS RESOLUTION SUBPROTOCOL

In the case when the customer does not receive message 6 in an appropriate amount of time, for any reason, a timeout will happen. In that case, the fairness for the customer is violated in a way it is described in Section 4.3. To rebalance the fairness, the customer initiates the FEIPS resolution subprotocol. This subprotocol consists of messages 7 and 8.

Message 7: C -> PG: {Sid, Amount, NonCPG, PubKC, DSigC(Sid, Amount, NonCPG)}PubKPG

This message represents the customer's request for confirmation of the payment, i.e. for a positive or negative response. Message 7 contains the session ID (Sid), the amount (Amount), a fresh nonce for the payment gateway (NonCPG), the temporary customer certificate (PubKC), and the digital signature of the Sid, Amount and NonCPG values. This content is encrypted with the payment gateway's public key.

Upon receiving this message, the payment gateway decrypts it by using its secret key. After that, the payment gateway checks whether the answer for the combination of Sid, Amount, NonCPG and PubKC is already generated, and for which customer. If that answer is already generated and if the digital signature of message 7 is verified by using the customer's public key (received in message 4), then the payment gateway sends message 8 directly to the customer. If that answer is not previously generated (i.e., if the payment gateway did not received the message containing the session ID (Sid)), the amount (Amount), a fresh nonce for the payment gateway (NonCPG), and the temporary customer certificate (PubKC) in question, then the negative response will be sent to the customer and these values will be saved in the payment gateway's database. If message 4 containing these values arrives afterwards, the payment gateway will know that the negative response is already generated and sent to the customer, so the same negative response will be sent to the merchant, too. This is the same action the payment gateway will take in a case when message 4 is replied by the merchant.

Message 8: PG -> C: {Response, DSigPG(Response, Sid, Amount, NonCPG)}PubKC

Message 8 is identical to message 6. This way the customer can verify if the response came from the payment gateway and if it is related to the current transaction.

It is important to mention that we avoid specifying the concrete amount of time after which the timeout shall occur, that is, it is not specified how long the customer will wait for message 6. This timeout period can be a few seconds, minutes or more, but for achieving the fairness it is necessary to exist, so the customer will not wait forever for message 6 to arrive.

6. FORMAL SPECIFICATION, ANALYSIS AND VERIFICATION OF THE FEIPS PROTOCOL

6.1 FORMAL SPECIFICATION

For the purpose of verifying the security requirements of FEIPS, we used AVISPA [61, 62], a tool for automated validation of complex Internet security protocols, like the protocols described by the Internet Engineering Task Force (IETF). We chose to use AVISPA for two reasons. First, it is easy to use for protocol modellers. This enabled us to focus on the protocol model itself. Second, the same protocol model can be checked with several tools for formal verification which use different approaches.

The FEIPS protocol is specified in the High Level Protocol Specification Language (HLPSL) [63, 64], translated into a state transition system called the intermediate format (IF) and fed to several back-end tools for completely automated analysis. There are four backend analysis engines available with the AVISPA tool: CL-AtSe (a constraint solver [65]), OFMC (a symbolic model checker [66]), SATMC (a SAT-based model checker [67]), and TA4SP (a tree automata based tool [68]). These tools employ the standard Dolev-Yao model, in which the intruder is assumed to have a complete control over the network [20]. Even though the tool has support for the specification of arbitrarily complex properties by the use of LTL (linear temporal logic) formulae, no analysis engine of the AVISPA tool actually uses this power, except of the extended version of the CL-AtSe engine [17, 69]. It is important to mention that only secrecy and authentication properties can be specified directly in HLPSL and analyzed by all AVISPA's backend analysis engines. Data integrity, non-repudiation and fairness properties are modeled indirectly, as described in Section 6.2.

A HLPSL specification is role-based. For each participant of a protocol, a role should be created. Also, protocol steps are modeled as transitions in a role. At the end, the desired security properties of the protocol are also modeled. AVISPA's backend tools search the state space for states that violate the specified security properties.

For the purpose of modeling of the FEIPS protocol, we have made several simplifications, as follows. We have modeled the roles of the customer, the merchant and the payment gateway, and we have abstracted the details of the payment gateway – bank connection, which is considered to be secure. We hypothesize that this is a reasonable assumption, since the compromised payment gateway – bank connection allows an intruder to view all sensitive information. We have also assumed that the shopping process itself already took place, i.e. that both the customer and the merchant know the order details and the amount beforehand. Also, we have assumed that the shopping process (the ordering sub-phase of the preparation phase) took place through HTTPS. The reason for this is explained in Section 6.2.

6.2 HLPSL SPECIFICATION OF ROLES AND SESSIONS

The HLPSL specification of the customer role is given in Figure 5. We will not describe the HLPSL semantics in detail; we will just point out some HLPSL elements important for our model. For more information on HLPSL, we refer to [63, 64]. As shown in Figure 5, the description of the customer role begins with a list of role parameters such as the credit card information (CardInf), the amount (Amount), and the order description (OrderDesc). Next, Figure 5 shows the list of variables local to the role (keyword local) such as the nonces (NonCPG) and the session ID (Sid). The key parts of the role specification are the transitions.

```
role customer(C,M,P: agent,
                CardInf, OrderDesc: text,
                Amount : nat,
                PubKM, PubKPG : public_key,
                PIN : message,
                Hash : hash_func
               ) played_by C def=
 local State : nat,
       NonCPG, Sid, Response : text,
       OI, PI : message,
       Kcm : symmetric_key,
       PubKC : public_key,
       SND, RCV: channel (dy)
 init State := 0
 transition
 1. State = 0 /\ RCV(start)
    =|>
    State' := 2 /\ PubKC' := new()
                /\ Kcm' := new()
                /\ SND({Kcm'}_PubKM.{PubKC'}_Kcm')
                /\ witness(C,P,cp_pubkc,PubKC')
 2. State = 2   /\ RCV({Sid'.{Hash(Sid')}_inv(PubKM)}_Kcm)
    =|>
    State' := 4 /\ NonCPG' := new()
                /\ PI' := CardInf.Amount.Sid'.NonCPG'.PIN.M
                /\ OI' := OrderDesc.Amount.Sid'
                /\ SND({{PI'.{Hash(PI')}_inv(PubKC)}_PubKPG.
                   {Hash(OI')}_inv(PubKC)}_Kcm)
                /\ secret(OrderDesc,order,{C,M})
                /\ secret(Amount,order,{C,M,P})
                /\ secret(CardInf,payment,{C,P})
                /\ witness(C,M,cm_deal,OI')
                /\ witness(C,P,cp_deal,PI')
 3. State = 4   /\ RCV({Response'.{Hash(Response'.Sid.
                   Amount.NonCPG)}_inv(PubKPG)}_Kcm)
    =|>
    State' := 11 /\ request(C,M,mc_deal,OI)
                 /\ request(C,P,pc_deal,PI)
                 /\ request(C,P,pc_pubkc,PubKC)
                 /\ request(C,P,pc_response,Response')
 4. State = 4    /\ RCV(timeout)
    =|>
    State' := 12 /\ SND({Sid.Amount.NonCPG.PubKC.{Sid.
                    Amount.NonCPG}_inv(PubKC)}_PubKPG)
 5. State = 12   /\ RCV({Response'.{Hash(Response'.Sid.
                    Amount.NonCPG)}_inv(PubKPG)}_PubKC)
    =|>
    State' := 15 /\ request(C,M,mc_deal,OI)
                 /\ request(C,P,pc_deal,PI)
                 /\ request(C,P,pc_pubkc,PubKC)
                 /\ request(C,P,pc_response,Response')
end role
```

FIGURE 5. HLPSL MODEL OF THE CUSTOMER

The customer role has five possible transitions. A transition is fired when the role instance is in an appropriate state (as specified by the State variable in HLPSL model) and upon receipt of an adequate message. The first transition corresponds to the first protocol message (Section 5, message 1). In this transition, a fresh session key (Kcm) and a temporary certificate (PubKC) are generated using the HLPSL function *new()*. This message is sent to the merchant over a public channel. Also, one witness event is generated in this transition. Witness events are described later in this section.

The trigger for the second transition is protocol message 2. Note that the digital signatures are represented as encryptions of message hashes by using the corresponding private keys, as this is how digital signatures are represented in AVISPA [63, 64]. The result of this transition is protocol message 3 and a series of secret and witness events. Secret events will also be described later in this section.

The third transition corresponds to protocol message 6, in which the response from the payment gateway is received and the protocol is terminated. In this transition, we generate the request events. These events are also described later in this section.

The trigger for the fourth transition is the timeout. The result of this transition is protocol message 7. The fifth transition corresponds to protocol message 8, in which the response from the payment gateway is received and the protocol is terminated. In this transition, the request events are generated. These request events are identical to the request events generated in the third transition. This is how the same security requirements are modeled, regardless of possibly different protocol execution paths.

The merchant and the payment gateway roles are modeled in a similar way. The merchant role transitions correspond to protocol messages 1-6, while the payment gateway role transitions correspond to protocol messages 4, 5, 7 and 8.

As stated in the beginning of Section 5, we demand that the protocol meets several security goals. These security goals are modeled as events in HLPSL, as follows.

Secrecy goals are achieved through secret events in HLPSL. Secret event *secret(m, id, {a,* *b})* means that message *m* should remain secret between the participants in a set (here *a* and *b*), creating a security goal identified by *id*. If there is a state in the transition system where the intruder learns *m* when he (the intruder) does not belong to the specified set, the goal *id* is violated. As it can be seen from the customer role and its secret events, we demand that the credit card information remains secret between the customer and the payment gateway (a goal identified as payment). Similarly, we demand that the amount remains secret between the customer, the merchant and the payment gateway, and that the order description remains secret between the customer and the merchant (a goal identified as order).

Authentication goals are achieved through witness and their corresponding request events. A witness event *witness(a, b, id, m)* means that the protocol participant *a* delivers a witness that he knows message *m,* to the protocol participant *b*. This goal is identified by *id*. A request event denoted as *request(b, a, id, m)* or *wrequest(b, a, id, m)* means that *b* expects the existence of a witness delivered previously by *a* over message *m*, for the goal *id*. A weak authentication goal *id* is violated whenever a state is reached where a request event (denoted as *wrequest*) is generated, for which there was no previous witness event. Strong authentication additionally requires that request events (denoted as *request*) do not occur more times than the corresponding witness events, meaning that *b* requires *a* to be alive, and thus eliminating replay attacks. Authentication goals in our model are strong authentication goals. For example, in the second transition, the customer acknowledges to wish to communicate with the merchant, using certain values (the order information), for the purpose of *cm_deal*. Later, in the third and fifth transitions, the customer requests that a similar guarantee be produced by the merchant during the protocol execution, for the purpose of *mc_deal*. Note that checking authentication in this way implicitly ensures data integrity, since the values in the *request* events must match those stated in the *witness* events. The *secret*, *witness* and *request* events are automatically fired when the transition is applied.

HLPSL requires that a special role, called session, is specified. This role corresponds to a single session of the protocol. It can be basic (consisting of just one role) or composed (consisting of multiple roles). In the case of FEIPS, it is defined as a composed session, consisting of the *customer*, *merchant*, and *payment* gateway roles (Figure 6). These roles execute together, in parallel (with an interleaving semantics) by means of the operator /\ .

```
role session(C,M,P: agent,
                CardInf : text,
                Amount : nat,
                OrderDesc : text,
                PubKM,
                PubKPG : public_key,
                PIN : message,
                Pin_func : hash_func,
                Hash : hash_func
               )def=
 composition

  customer(C,M,P,CardInf,Amount,OrderDesc,PubKM,PubKPG,PIN,Hash) /\
  merchant(C,M,P,Amount,OrderDesc,PubKM,PubKPG,Hash) /\
  paymentgateway(C,M,P,PubKM,PubKPG,Pin_func,Hash)
end role
```

FIGURE 6. MODEL OF AN FEIPS SESSION IN HLPSL

6.3 ANALYSIS

In the analysis, we modeled three parallel sessions: one with honest participants (normal session), one with a malicious customer, and one with a malicious merchant. In HLPSL, this is achieved through a top-level role called environment. The environment role for the first FEIPS scenario is given in Figure 7. As it can be seen from the figure, we assumed the payment gateway (TTP) to be honest. The payment gateway role is never played by the intruder *i*, while customer and merchant roles can be played by the intruder *i*. We hypothesize that this is a reasonable assumption, since a dishonest payment gateway is allowed to view the sensitive credit card information anyway, and poses a major security problem.

The *intruder_knowledge* set from Figure 7 contains all the constants that the intruder knows prior to the protocol execution. Note that in this model, the constants *amount1* and *orderDesc1* are not contained in this set, that is, we assume that the shopping process takes place through HTTPS. Otherwise, the secrecy of the order description between the customer and the merchant could not be achieved. The payment gateway knows a function called *pin_func* which we use to model credit card authorization based on credit card number/PIN combinations; this function enables the payment gateway to create a map between credit card numbers, PINs and user identities.

```
role environment() def=
       const h: hash_func,
       pin_func : hash_func,
       order,payment,mp_deal,pm_deal,cm_deal,mc_deal : protocol_id,
       cp_deal,pc_deal,cp_pubkc,pc_pubkc : protocol_id,
       pc_response,pm_response : protocol_id,
       c,m,p: agent,
       pubkm,pubkpg,pubk_i: public_key,
       cardInf_c,cardInf_i,orderDesc1,orderDesc2,orderDesc3: text,
       timeout : text,
       amount1,amount2,amount3 : nat

 intruder_knowledge = {c,m,p,pubkm,pubkpg,pubk_i,inv(pubk_i),
                      cardInf_i,amount2,orderDesc2,amount3,
                      orderDesc3,h, pin_func(i.cardInf_i)}
 composition

  session(c,m,p,cardInf_c,amount1,orderDesc1,pubkm,
          pubkpg,pin_func(c.cardInf_c),pin_func,h)
  /\ session(i,m,p,cardInf_i,amount2,orderDesc2,pubkm,
          pubkpg,pin_func(i.cardInf_i),pin_func,h)
  /\ session(c,i,p,cardInf_c,amount3,orderDesc3,pubk_i,
          pubkpg,pin_func(c.cardInf_c),pin_func,h)
end role
```

FIGURE 7. ENVIRONMENT ROLE OF THE FEIPS MODEL IN HLPSL

We also modeled the following security goals (Figure 8): secrecy of the order description between the customer and the merchant; secrecy of the credit card information between the customer and the payment gateway; and (strong) mutual authentication between all pairs of participants (i.e., customer – merchant on the order information, merchant – payment gateway on the amount and the response, and customer – payment gateway on the payment information, the temporary certificate and the response).

```
goal
   authentication_on mp_deal
   authentication_on pm_deal
   authentication_on cm_deal
   authentication_on mc_deal
   authentication_on cp_deal
   authentication_on pc_deal
   authentication_on pc_pubkc
   authentication_on cp_pubkc
   authentication_on pc_response
   authentication_on pm_response
   secrecy_of order
   secrecy_of payment
end goal
```

FIGURE 8. GOAL SECTION OF THE FEIPS MODEL IN HLPSL

Currently, non-repudiation and fairness goals cannot be modeled explicitly in HLPSL and analyzed by all AVISPA's backend analysis engines [70]. However, it is well known that non-repudiation is a form of authentication [71]. As noted before, a strong authentication between two parties on a certain property implicitly ensures the data integrity of such a property between those two parties. Additionally, fairness can be defined as a function of non-repudiation of origin (NRO) and of non-repudiation or receipt (NRR) [70]. The NRO property requires a guarantee that the sender sent some particular message to the recipient, while the NRR property requires a guarantee that the recipient received some particular message. According to the definition of the fairness (Section 2), we can tell that we have fairness if both the properties, NRO and NRR, are ensured or both are flawed at the end of the protocol execution, for a given message M [70].

As already explained in Section 4.3, the fair-exchange problem in a case of the FEIPS protocol is a problem of exchange of electronic money (payment information) for a digital receipt (response). This is why, in the case of the FEIPS protocol, we can leave out the merchant role and define fairness as follows. In the FEIPS protocol, NRO should provide the guarantee that the customer sent a message to the payment gateway containing the payment information. If this message is signed by the customer and successfully received by the payment gateway, it can be used as the evidence that the customer sent this message to the payment gateway. Thus,

NRO_{FEIPS} = PG authenticates C on PI.

In the FEIPS protocol, NRR should provide the guarantee that the payment gateway received (or did not receive) the message containing the payment information. If the message containing the payment information is received from the merchant (Section 5, message 3), then the response will be generated by the payment gateway and eventually delivered to the customer (Section 5, message 6 or message 8). If this response is signed by the payment gateway and successfully received by the customer, it can be used as the evidence that the payment gateway sent this message to the customer. Also, it can be used as the evidence that the payment gateway received (or did not receive) the message containing the payment information both sent by the customer (Section 5, message 3) and transferred by the merchant (Section 5, message 4). Thus,

NRR_{FEIPS} = C authenticates PG on Response.

In other words, the fairness is achieved: if the payment gateway authenticates the customer on the payment information (NRO_{FEIPS}) and the customer authenticates the payment gateway on the response (NRR_{FEIPS}). If both properties, NRO and NRR, are flawed the fairness is also achieved. It is important to mention again that fairness can only be violated for the customer and by the merchant (see Section 4.3).

6.4 VERIFICATION RESULTS

The analysis of our specification of the FEIPS protocol resulted in no attacks. We assumed a strongly-typed model, i.e., a model in which a message field is always interpreted according to its type, and no two fields of different types can be substituted for each other. The strongly-typed model leads to smaller search spaces at the cost of abstracting away possible type-flow attacks on a protocol [66, 67, 72].

Two of the AVISPA backend analysis engines, CL-AtSe and OFMC have verified the protocol to be safe under the assumption of a bounded number of sessions. In other words, they found that the protocol achieves all of the specified goals. Also, it is important to mention that the analysis was too complex for the current versions of the other two tools – SATMC and TA4SP. We were unable to get any output from them in a reasonable amount of time.

Corollary 1. *The FEIPS is a fair exchanging protocol satisfying the fundamental security requirements: confidentiality, authentication, data integrity, nonrepudiation.*

The results obtained using AVISPA confirm the Corollary 1. In summary, the confidentiality (secrecy) requirement of the FEIPS protocol is verified through the secrecy goals in the AVISPA model on all the important data between the participants of the protocol (i.e., credit card information between the customer and the payment gateway; the amount between the customer, the merchant and the payment gateway; and the order description between the customer and the merchant). The authentication requirement of the FEIPS protocol is verified through the authentication goals in the AVISPA model, i.e. strong (mutual) authentication goals are achieved between all the pairs of the participants of the protocol (i.e., customer – merchant on the order information; merchant – payment gateway on the amount and the response; and customer – payment gateway on the payment information, the temporary certificate, and the response). As noted before, checking authentication in this way implicitly ensures data integrity. This means that data integrity of the order information, the payment information, the amount, the temporary certificate and the response is achieved, and that FEIPS protocol satisfies the data integrity requirement.

As already mentioned, we modeled non-repudiation goals implicitly through authentication goals. Since the payment gateway authenticates both the customer (on the payment information and temporary certificate) and the merchant (on the amount), neither of them can repudiate their participation in the transaction. In the case of a dispute, the corresponding entry in the payment gateway database can be used as a proof of nonrepudiation. This way FEIPS protocol satisfies nonrepudiation requirement.

Another dispute that can arise is the one concerning the contents of the customer's order. In this case, it is the merchant's obligation to provide the proof. If the merchant fails, the customer wins the dispute. The digital signature of the order information, *DSigC(OI)*, (Section 5, message 3) combined with the corresponding entry in the payment gateway database (containing the user certificate), provides the merchant with sufficient evidence of the customer's order. It is important to notice that the merchant cannot generate this message by themselves because the merchant does not know the customer's private key. Hence, this message must originate from the network. This is why this can be expressed as an authentication problem on *DSigC(OI)*. Since the authentication on that particular message is reached (as discussed earlier), it can be claimed that the message has been received from the customer and not from someone else.

The FEIPS protocol satisfies the fairness requirement as the payment gateway authenticates the customer on the payment information (NRO_{FEIPS}) and the customer authenticates the payment gateway on the response (NRR_{FEIPS}). Further discussion of the fair-exchange requirements and their fulfillment is given in the following section.

7. DISCUSSION OF FAIR-EXCHANGE

As already mentioned in Section 6.1, the AVISPA tool uses the Dolev-Yao intruder model to automatically verify the specified security goals, including fairness, which is specified indirectly through authentication goals. Although the non-repudiation and fairness goals can be based on authentication problems, this solution does not fully suit the current implementation of the intruder model in the AVISPA tool [69, 70, 73]. When the intruder plays a role of a protocol participant, then the intruder will act as a dishonest protocol participant. In such a situation, the current implementation of the AVISPA's backend engines (except the extended version of the Cl-AtSe engine [17, 69]) will throw away the protocol transitions of the impersonated protocol participant and the knowledge of this protocol participant will be added to the intruder's knowledge [69, 70, 73]. This is not a problem for verification of authentication and secrecy properties, but since the intruder will not fire the transitions that would generate the witness and request predicates of the impersonated protocol participants, non-repudiation can not be verified [70]. This is why specification of a session where the intruder plays a role of the merchant, will not permit to consider all the possible scenarios, and some attacks may be missed with such a restriction. This is why we also explain fair-exchange requirements informally.

As mentioned in Section 2, a fair exchange protocol is a protocol satisfying the following requirements: effectiveness, fairness and timeliness.

7.1 EFFECTIVENESS

According to the definition of the *effectiveness*, all participants are assumed to be honest and none of them want to abort the protocol [24-26]. This condition is necessary because, in asynchronous systems, message delays are unknown (unbounded), but finite. According to this, in a certain period of time it is not possible for a protocol participant, who behaves according to the protocol, to distinguish the following two situations:

1. when the other protocol participant behaves according to the protocol, but the network is slow,
2. when the other protocol participant does not behave according to the protocol.

It is important to notice that there is a risk that the exchange between protocol participants will not succeed because one of them did not wait long enough (timeout occurred) for the message of the other. As messages can be delayed for unbounded (but finite) amount of time, it is obvious that the FEIPS protocol demands that an intruder can not block messages forever between an honest protocol participant and a TTP. Such links are called resilient links. In the case of the FEIPS protocol, we demand two resilient links: one between the merchant and the payment gateway, and the other one between the customer and the payment gateway.

7.2 TIMELINESS

The *timeliness* property ensures that the protocol will eventually terminate for all participants that behave according to the protocol. Also, the degree of achieved fairness will not change after the termination point. In addition, timeliness guarantees that at least one party has the ability both to abort the normal protocol execution and to execute a resolution subprotocol that will be executed in a finite amount of time. In the case of the FEIPS protocol, we introduced the timeout. This is a period of time in which the customer waits for the response from the merchant, after which the FEIPS resolution subprotocol is started. As this subprotocol is executed between the customer and the payment gateway, and as the communication link between them is resilient, it is obvious that the FEIPS protocol will eventually terminate. After the termination point, the degree of achieved fairness will not change, because the FEIPS protocol does not provide a way for that to happen. If the customer, after the termination point, has a digital receipt of successful payment (Section 5, the response message), the money transfer is also made to the merchant's account. If the money transfer is made to the merchant's account, then after the protocol termination, the customer will also have a corresponding digital receipt (Section 5, message 6 or message 8). If one of them has not received expected goods (the customer – digital receipt, the merchant – the money), then the other participant has also not received what he expected.

7.3 FAIRNESS

Now we will analyze the protocol messages in the context of the *fairness*. If the protocol execution is terminated during the exchange of the first two messages, it is obvious that fairness can not be violated neither for the customer nor for the merchant. The reason for this is that the payment message (containing payment information) was not formed by the payment web segment and sent to the merchant. This is why the payment process will not be executed, and the merchant will not gain an unfair advantage over the customer.

If the protocol execution is terminated during the exchange of the third protocol message, the payment message was formed and sent by the customer, but it was not received by the merchant. In this situation the payment process will not be executed, and the merchant will not gain an unfair advantage over the customer. The customer will wait for the response message (message 6) until the timeout occurs. When the timeout occurs, the customer initiates the FEIPS resolution subprotocol (message 7). As the payment gateway did not receive the payment message from the merchant (message 4), it will generate the negative response for the customer (message 8).

As it is mentioned in Section 4.3, the protocol execution can be terminated during the exchange of the fourth protocol message, for the following two situations: a network connection failure has occurred or a dishonest merchant does not want to forward the message 4 to the payment gateway. In both situations the merchant has the payment message and can forward it to the payment gateway afterwards. As in the previous case, the customer will wait for the response message (message 6) until the timeout occurs and initiate the FEIPS resolution subprotocol (message 7). If the payment gateway did not received the payment message from the merchant (message 4), it will generate the negative response for the customer (message 8). Also, the payment gateway will store the following values in its database: session ID (Sid), the amount (Amount), nonce for the payment gateway (NonCPG) and the customer's temporary certificate (PubKC). If the merchant sends message 4 to the payment gateway afterwards, the payment gateway will determine whether it will already have generated the response for the same session. The same negative response will also be returned to the merchant. It is important to notice the following. It is possible to happen (in a case of a "slow" network between the merchant and the payment gateway) that the message 4 transfer takes longer than the timeout, so that the payment process will not be executed. Although this is an implementation issue, it can be seen that the proper selection of the timeout value is very important. If the protocol execution is terminated during the exchange of messages 5 and 6, it is important to note that the payment process is already finished. These are the situations when the fairness for the customer is violated. When the timeout occurs, the customer initiates the FEIPS resolution subprotocol (message 7). The payment gateway will determine if they will already have generated the response for the corresponding merchant and the combination of the session ID (Sid), the amount (Amount) and the nonce for the payment gateway (NonCPG). This answer will also be sent to the customer (message 8).

It is important to notice that the payment gateway acts as a TTP only when the protocol participants have a dispute that they cannot resolve by themselves in a satisfactory manner, i.e. in a situation when the fairness is violated for the customer. This approach is known as the optimistic approach [22]. The response sent to the customer in a case when the FEIPS resolution subprotocol is executed is the same as the response sent to the customer in a normal protocol execution. This is not the case with many existing fair-exchange protocols. Also, it is important to notice that, in the original IPS, the payment gateway communicated only with the certified merchants. In this solution for the FEIPS protocol, the payment gateway must also communicate with the customers. This is why the openness of the payment gateway to the outside world becomes higher, and thus, increasing the possibility of DoS attacks. Still, this possibility is significantly lower than that of the fair-exchange protocols with inline or online TTP's. This increased possibility of DoS attacks can be reduced by distributing the functionality of the TTP service across several entities [14].

7.4 ADDITIONAL DISCUSSION

It is important to explain why the FEIPS protocol must start with the setup subprotocol, i.e. with the first message. Along with this explanation, a brief discussion about a possible modification of the protocol is given.

If the Customer tries to start the protocol by sending message 3, then this message will be discarded by the Merchant and the protocol will be terminated. The reason for this is that message 3 must contain (among other information) a digital signature of the relevant order information DSigC(OrderDesc, Amount, Sid), where Sid (session ID) is previously generated by the Merchant and sent to the Customer in message 2. This unique freshly generated random number serves as an ID for the session and as a transaction identifier. If the Customer does not have the right Sid then the digital signature of the order information cannot be verified by the Merchant. This is why message 3 will be discarded by the Merchant and the protocol will be terminated.

The Customer can also try to start the protocol by sending message 7 directly to the Payment Gateway. To construct a valid message 7, the Customer also needs a valid session ID (Sid) generated by the Merchant and sent to the Customer in message 2. Upon receiving message 7, the Payment Gateway checks whether the response for the combination of Sid, Amount, NonCPG and PubKC is already generated, and for which customer. In this case, when the Customer tried to start the protocol by sending message 7, the response for this combination of Sid, Amount, NonCPG and PubKC is not previously generated, and this is why the negative response will be generated by the Payment Gateway sent to the Customer.

If the protocol was modified in order to send payment information directly from the Customer to the Payment Gateway, not through the Merchant (like it is now), then we would basically have a two-party payment protocol. In this case, the protocol would not be a fair exchange protocol. As it is already mentioned, it has been formally proven that it is impossible to achieve fair exchange without a TTP [37, 40]. If we neglect this fact, we would still have at least one important issue to resolve: we should have one/some additional message(s) between the Merchant and the Payment Gateway. This/these message/messages is/are necessary because the Merchant must get information about the payment. This is why the modified protocol would contain at least five messages (in best case): 1. C->M; 2. M->C; 3. C-PG; 4. PG->M, and 5. PG->C (i.e., the Payment Gateway must send two messages about a successful payment – one to the Merchant and the other one to the Customer). Again, this modified protocol would not be a fair exchange protocol. Therefore, one additional message in the FEIPS protocol (or three in the case of a dispute resolution, which is the worst case), is not so high price to pay for a fair exchange protocol.

8. COMPARISON TO THE RELATED WORK

In this section the FEIPS protocol is compared against e-payment protocols that are designed for the payment of physical goods. Therefore, the FEIPS protocol is compared against the Zhang et al. [8], Li et al. [53], and Alaraj [44] protocols and the classical e-payment protocols for credit card payments SET, Visa 3-D Secure and SSL/TLS.

Comparing FEIPS with them, we find that the main advantages of FEIPS are:

– prior registration of the customer is not required (unlike other protocols).
– The customer has an asymmetric key pair (temporary certificate) which is not generated by the bank or TTP, and which is not associated with the customer's bank account (unlike the Zhang et al. protocol, the Li et al. protocol and SET). This way, banks do not have to maintain a very large database of customers' public keys associated with their bank accounts. Also, the customer is not required to install any additional software for secure payments (unlike SET), even if the customer has a digital certificate.
– the protocol is simple and practical. The Zhang et al. protocol uses the concept of a delivery cabinet, while the Li et al. and Alaraj protocols are using the concept of a

delivery agent. Both concepts appear to be complicated and potentially impractical.

– the protocol is a fair-exchange protocol (Section 6 and Section 7) similar to the Zhang et al., Li et al., and Alaraj protocols. FEIPS and the Alaraj protocol use offline TTP, while the Zhang et al. and Li et al. protocols do not require involvement of a TTP. Still, FEIPS and the Alaraj protocol consist of 6 messages to be exchanged between participants, while the Zhang et al. and Li et al. protocols consist of 7 and 8 messages to be exchanged between participants, respectively. The resolution subprotocols of both FEIPS and the Alaraj protocol consist of 2 messages. The Li et al. protocol does not discuss the dispute resolution phase and therefore it is not clear how the disputes are resolved if one of the participants acts dishonestly. Classical e-payment protocols for credit card payments, like SET, Visa 3-D Secure, SSL/TLS and their improved variants (e.g., the improved SET protocol) are designed to protect protocol participants, who trust one another, from potential attackers. This is the main reason why they are not fair-exchange protocols.
– security requirements for FEIPS (including fairness) are formally verified (unlike the Zhang et al., Li et al., and Alaraj protocols). According to [23], there is a limitation in the fairness of the Zhang et al. protocol, which can bring the merchant in an advantage.
– the protocol does not require the customer to send two payments for one transaction (unlike Zhang et al. protocol).

The main drawback of the FEIPS is that the payment gateway must also communicate with the customers (when the FEIPS resolution subprotocol is initiated). This increases the possibility of DoS attacks which can be reduced by distributing the functionality of the TTP service across several entities [14].

9. CONCLUSION

Fair exchange is an important property that needs to be addressed by all e-commerce protocols. Keeping this in mind, we proposed FEIPS, an e-commerce protocol that ensures fairness within the scope of the protocol itself. This protocol is an extended version of the IPS protocol [21]. The FEIPS protocol is designed for the payment of physical goods and falls into the category that uses a trusted third party for assuring fair-exchange. Specially, the TTP role is played by the trusted protocol participant – the payment gateway. The FEIPS protocol is designed to be simple and practical, unlike other similar protocols designed for the payment of physical goods.

Most of the times, none of participants of an e-commerce protocol (in all e-commerce protocols) misbehaves. Keeping this in mind, we have taken an optimistic approach in the FEIPS protocol – the payment gateway does not play the TTP role unless necessary. This is in contrast to some other similar protocols that actively use the TTP to ensure fair exchange. This approach reduces the bottleneck at the TTP and also allows for accomplishing requested security objectives by using an off-line TTP that need not be active for the entire duration of the transaction.

In summary, the FEIPS protocol is a fair-exchange e-payment protocol that satisfies fundamental security requirements. Today, there is a common agreement that payment protocols need a proof of security to be acceptable. To prove the security of the FEIPS protocol, we modeled the protocol using HLPSL. We have run the analysis using the AVISPA tool. The results of the analysis were positive; in other words the protocol achieves the desired properties: confidentiality, integrity, authentication, non-repudiation and fairness. Currently, we are working on a modification of the protocol, trying to optimize it for purchase and delivery of electronic goods, while retaining the achieved properties of FEIPS.

REFERENCES

1. Nenadic, A. (2005) *A Security Solution for Fair-Exchange and Non-Repudiation in E-Commerce*. Ph.D. thesis, University of Manchester, UK.
2. Lawrence, E., Corbitt, B., Tidwell, A., Fisher, J. (1998) *Internet Commerce: Digital Models for Business*. John Wiley & Sons, New York.
3. Ford, W., Baum, M. (1997) *Secure Electronic Commerce.* Prentice Hall, New Jersey.
4. Stallings, W. (1995) *Network and Internetwork Security, Principles and Practice*. Prentice Hall, New Jersey.

5. Merkow, S., Breithhaupt, J., Wheeler, K. (1998) *Building SET Applications for Secure Transactions*. John Wiley & Sons, New York.
6. Bhimani, A. (1996) Securing the Commercial Internet. *Communications of the ACM*, 39, 29-35;
7. Tsiakis, T., and Stephanides, G. (2005) The concept of security and trust in electronic payments. *Computers & Security*, 24, 10-15.
8. Zhang, O., Markantonakis, K., Mayes, K. (2006) A Practical Fair Exchange E-Payment Protocol for Anonymous Purchase and Physical Delivery. *Proceedings of 4th ACS/IEEE International Conference on Computer Systems and Applications*, Sharjah/Dubai, United Arab Emirates, 8-11 March, pp. 851-858. IEEE Computer Society Press.
9. Wang, B., Ma, H., Chen, J. (2009) Formal Analysis of Fairness in E-Payment Protocol Based on Strand Space. *Lecture Notes on Computer Science*, 5854, 469-478.
10. Franklin, M. K., and Reiter, Mi. K. (1997) Fair Exchange with a Semi-Trusted Third Party (extended abstract). *Proceedings of the ACM Conference on Computer and Communications Security*, Zurich, Switzerland, 2-4 April, pp. 1-5. ACM New York, NY, USA.
11. Ray, I., Ray, I., Narasimhamurthi, N. (2000) A Fair Exchange E-commerce Protocol with Automated Dispute Resolution. *Proceedings of the IFIP Workshop on Database Security*, Schoorl, the Netherlands, 21-23 Aug, pp. 27–38. Springer-Verlag Berlin, Heidelberg.
12. Asokan, N., Schunter, M., Waidner, M. (1997) Optimistic Protocols for Fair Exchange. *Proceedings of the ACM Conference on Computer and Communications Security*, Zurich, Switzerland, April, pp. 6–17. ACM New York, NY, USA.
13. Pagnia, H., Vogt, H., Gartner, F.C. (2003) Fair Exchange. *The Computer Journal*, 46, 55–75.
14. Ray, I., Ray, I., Natarajan, N. (2005) An anonymous and failure resilient fair-exchange e-commerce protocol. *Decision Support Systems*, 39, 267-292.
15. Luo, D., and Zhang, J. (2011) Efficient Self-fair Exchange Anonymous E-payment Protocol. *Journal of Computational Information Systems*, 7, 1302-1309.
16. Nenadic, A., and Zhang, N. (2003) Non-repudiation and Fairness in Electronic Data Exchange. *Proceeding of the 5th International Conference on Enterprise Information Systems - ICEIS '03*, Angers, France, pp. 55–62.
17. Pfitzmann, B., Schunter, M., Waidner, M. (1995) How to break another "provably secure" payment system. *Advances in Cryptology: EUROCRYPT '95*, Saint-Malo, France, 21–25 May, pp.121–132. Springer-Verlag Berlin, Heidelberg.
18. Kailar, R. (1995) Reasoning About Accountability in Protocols for Electronic Commerce. *Proceedings of the 14th IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 8-10 May, pp. 236-250. IEEE Computer Society Washington, DC, USA.
19. Bella, G., Massacci, F., Paulson, L.C. (2005) An overview of the verification of SET. *International Journal of Information Security*, 4, 17-28.
20. Cederquist, J., Corin, R., Torabi Dashti, M. (2005) On the quest for impartiality: Design and analysis of a fair non-repudiation protocol. *Lecture Notes in Computer Science*, 3783, 27-39.
21. Djuric, Z., Maric, O., Gasevic, D. (2007) Internet Payment System: A New Payment System for Internet Transactions. *Journal of Universal Computer Science*, 13(4), 479-503.
22. Asokan, N. (1998) *Fairness in electronic commerce*. Ph.D. thesis, University of Waterloo, Canada.
23. Alotaibi, A., and Aldabbas, H. (2012) A review of Fair Exchange Protocols. *International Journal of Computer Networks & Communications*, 4(4), 307-319.
24. Draper-Gil, G., Ferrer-Gomila, J., Hinarejos, M.F., Zhou, J. (2013) An Asynchronous Optimistic Protocol for Atomic Multi-Two-Party Contract Signing. *The Computer Journal*, doi:10.1093/comjnl/bxs175.
25. Zhao, Y., and Qin, Z. (2012) An Optimistic Protocol for Distributed Fair Exchange. *Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, Palermo, Italy, 4-6 July, pp. 395-399. IEEE Computer Society Washington, DC, USA.
26. Liu, Z., Pang, J., Zhang, C. (2011) Verification of A Key Chain Based TTP Transparent CEM Protocol. *Electr. Notes Theor. Comput. Sci. (ENTCS)*, 274, 51-65.
27. Bella, G., and Paulson, L.C. (2006) Accountability Protocols: Formalized and Verified. *ACM Transactions on Information and System Security*, 9(2), 138-161.
28. Ateniese, G., Medeiros, B., Goodrich, M.T. (2001) TRICERT: A distributed certified E-mail scheme. *Symposium on Network and Distributed Systems Security (NDSS 2001)*, San Diego, CA, USA, February, pp. 47-58. Internet Society.
29. Ben-Or, M., Goldreich, O., Micali, S., Rivest, R. (1990) A Fair Protocol for Contract Signing. *IEEE Transactions on Information Theory*, 36(1), 40–46.
30. Ray, I., and Ray, I. (2003) An Optimistic Fair Exchange E-commerce Protocol with Automated Dispute Resolution. *Lecture Notes in Computer Science*, 1875, 84-93.
31. Blum, M. (1983) How to Exchange (Secret) Keys. *ACM Transactions on Computer Systems*, 1(2), 175–193.

32. Damgard, I. B. (1994) Practical and Provably Secure Release of a Secret and Exchange of Signatures. *Proceedings of Advances in Cryptology - EUROCRYPT '93*, Lofthus, Norway, 23–27 May, pp. 200–217. Springer-Verlag Berlin, Heidelberg..
33. Even, S., Goldreich, O., Lempel, A. (1985) A Randomized Protocol for Signing Contracts. *Communications of the ACM*, 28(6), 637–647.
34. Goldreich, O. (1984) A Simple Protocol for Signing Contracts. *Proceedings of Advances in Cryptology - CRYPTO '83*, Santa Barbara, California, USA, 21-24 August, pp. 133–136. Springer-Verlag, US.
35. Okamoto, T., and Ohta, K. (1994) How to Simultaneously Exchange Secrets by General Assumptions. *Proceedings of the ACM Conference on Computer and Communication Security*, Fairfax, Virginia, USA, 2-4 November, pp. 184–192. ACM New York, USA.
36. Bahreman, A., and Tygar, J.D. (1994) Certified Electronic Mail. *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, San Diego, CA, USA, 3 February, pp. 3–19. Internet Society.
37. Markowitch, O., and Roggeman, Y. (1999) Probabilistic Non-repudiation without Trusted Third Party. *Proceedings of the Conference on Security in Communication Networks - SCN '99*. Amalfi, Italy, 16-17 September.
38. Mitsianis, J. (2001) A New Approach to Enforcing Non-Repudiation of Receipt, Manuscript.
39. Rabin, M.O. (1983) Transaction Protection by Beacons. *Journal of Computer and System Science*, 27, 256–267.
40. Pagnia, H., and Gartner, F.C. (1999) On the impossibility of fair exchange without a trusted third party. Technical Report TUD-BS-1999-02, Darmstadt University.
41. Okada, Y., Manabe, Y., Okamoto, T. (2006) Optimistic Fair Exchange Protocol for E-Commerce. *The 2006 Symposium on Cryptography and Information Security*. Hiroshima, Japan, 17-20 January.
42. Li, X., Wang, Q., Chen, L. (2008) Analysis on Cyclic Multi-party Fair Exchange Protocols. *Proceedings of International Conference on Computer Science and Software Engineering*, Wuhan, China, 12-14 December, pp. 601-604. IEEE Computer Society Washington, DC, USA.
43. Shao, J., Feng, M., Zhu, B., Cao, Z. (2007) An efficient certified email protocol. *Lecture Notes in Computer Science*, 4779, 145-157.
44. Alaraj, A.M. (2012) Purchase of physical products online. *International Conference on Multimedia Computing and Systems*, Tangier, Morocco,10-12 May, pp. 937-940. IEEE Computer Society Washington, DC, USA.
45. Dashti, M.T., Nair, S.K., Jonker, H.L. (2008) Nuovo DRM Paradiso: Designing a Secure, Verified Fair DRM Scheme. *Fundamentae Informatica (FI)*, 89, 1-25.
46. Kremer, S., and Raskin, J.-F. (2001) A game-based veri cation of non-repudiation and fair exchange protocols. *Lecture Notes in Computer Science*, 2154, 551-565.
47. Zhou, J., Deng, R., Bao, F. (2000) Some Remarks on a Fair Exchange Protocol. *Lecture Notes in Computer Science*, 1751, 46-57.
48. Gil, G.D., Zhou, J., Ferrer-Gomila, J.L. (2010) An Agent-Mediated Fair Exchange Protocol. *Lecture Notes in Computer Science*, 6476, 235-250.
49. Küpçü, A., Lysyanskaya, A. (2012) Usable optimistic fair exchange. *Computer Networks*, 56, 50–63.
50. Burk, H., and Pfitzmann, A. (1990) Value Exchange Systems Enabling Security and Unobservability. *Computers and Security*, 9(9), 715–712.
51. Cox, B., Tygar, J.D., Sirbu, M. (1995) NetBill Security and Transaction Protocol. *Proceedings of the First Usenix Workshop on Electronic Commerce*, New York, New York, July, pp. 77-88. USENIX Association Berkeley, CA, USA.
52. Alaraj, A., and Munro, M. (2007) An e-commerce Fair Exchange Protocol for exchanging Digital Products and Payments. *Proceedings of IEEE/ACM International Conference on Digital Information Management*, Lyon, France, 28-31 October, pp. 248-253. Springer-Verlag Berlin, Heidelberg.
53. Li, H., Kou, W., Du, X. (2006) Fair E-Commerce Protocols without a Third Party. *Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC'06)*, Cagliari, Italy, 26-29 June, pp. 324-327. IEEE Computer Society Washington, DC, USA.
54. Bella, G., Massacci, F., Paulson, L.C., Tramontano, P. (2000) Formal verification of cardholder registration in SET, in: Cuppens, F., Deswarte, Y., Gollman, D., Waidner, M. (Eds.), *Computer Security ESORICS 2000*, *Lecture Notes in Computer Science*, 1895, 159-174.
55. Jarupunphol, P., and Mitchell, C.J. (2003) Measuring 3-D Secure and 3D SET against e-commerce end-user requirements. *8th Collaborative Electronic Commerce Technology and Research Conference*, Galway, Ireland, June, pp. 51–64.
56. Wrona, K., Schuba, M., Zavagli, G. (2001) Mobile payment — state of the art and open problems, in: Fiege, L., Mühl, G., Wilhelm, U. G. (Eds.). Proceedings of 2nd International Workshop WELCOM, *Lecture Notes in Computer Science*, 2232, 88-100.

57. Rescorla, E. (2001) *SSL and TLS — Designing and Building Secure Systems*. Addison-Wesley, Massachusetts.
58. Shu, Y., and Kanliang, W. (2009) The influence of information sensitivity compensation on privacy concern and behavioral intention. *ACM SIGMIS Database*, 40, 38–51.
59. Antoniou, G., and Batten, L. (2011) E-commerce: protecting purchaser privacy to enforce trust. *Electronic Commerce Research*, 11(4), 421-456.
60. Boping, Z., and Shiyu, S. (2009) An Improved SET Protocol. *Proceedings of the International Symposium on Information Processing*, Huangshan, China, 21-23 August, pp. 267-272. Academy Publisher.
61. Armando, A. et al.. The AVISPA Tool for the automated validation of internet security protocols and applications, *Proceedings of the 17th international conference on Computer Aided Verification*, Edinburgh, Scotland, pp. 281-285. Springer–Verlag Berlin, Heidelberg.
62. Viganò, L. (2006) Automated security protocol analysis with the AVISPA tool. *Electronic Notes in Theoretical Computer Science*, 155, 61–86.
63. Von Oheimb, D. (2005) The High-Level Protocol Specification Language HLPSL developed in the EU project AVISPA. *Proceedings of APPSEM 2005 Workshop*, Frauenchiemsee, Germany, 13 September.
64. Chevalier, Y., Compagna, L., Cuellar, J., Hankes Drielsma, P., Mantovani, J., Modersheim, S., Vigneron, L. (2004) A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols. *Journal of Automated Software Engineering*, 180, 193–205.
65. Turuani, M. (2006) The CL-Atse Protocol Analyser, *Proceedings of the 17th international conference on Term Rewriting and Applications*, Seattle, WA, USA, 12-24 August, pp. 277–286. Springer-Verlag Berlin, Heidelberg.
66. Basin D., Modersheim S., Vigano L. (2005) OFMC: A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3), 181–208.
67. Armando, A., Compagna, L. (2004) SATMC: a SAT-based Model Checker for Security Protocols. *Proceedings of the 9th European Conference on Logics in Artificial Intelligence*, Lisbon, Portugal, pp. 617–627. Springer-Verlag Berlin, Heidelberg.
68. Boichut, Y., Heam, P-C, Kouchnarenko, O. (2005) Automatic Verification of Security Protocols Using Approximations. Research Report RR-5727, INRIA-Lorraine – CASSIS Project.
69. Santiago, J., and Vigneron, L. (2007) Optimistic non-repudiation protocol analysis. *Proceedings of the 1st IFIP TC6 /WG8.8 /WG11.2 international conference on Information security theory and practices: smart cards, mobile and ubiquitous computing systems*, Heraklion, Crete, Greece, 9-11 May, pp. 90-101. Springer-Verlag Berlin, Heidelberg.
70. Santiago, J., and Vigneron, L. (2005) Study for Automatically Analysing Non-repudiation. *Actes du 1er Colloque sur les Risques et la Sécurité d'Internet et des Systèmes*, CRiSIS, Bourges, France, October, pp. 157-171.
71. Ryan, P., Schneider, S., Goldsmith, M., Lowe, G., Roscoe, B. (2000) *Modelling & Analysis of Security Protocols*, Addison Wesley, Massachusetts.
72. Heather, J., Lowe, G., Schneider, S. (2000) How to prevent type flaw attacks on security protocols. *Proceedings of The 13th Computer Security Foundations Workshop (CSFW'00)*, Cambridge, England, UK, 3-5 July, pp.105-119. IEEE Computer Society Press.
73. Klay, F., Santiago, J., Vigneron, L. (2007) Automatic Methods for Analyzing Non-Repudiation Protocols with an Active Intruder. *Lecture Notes in Computer Science*, 5491, 192-209.