An Optimistic Fair Exchange E-commerce Protocol for Complex Transactions

Cătălin V. Bîrjoveanu[1] and Mirela Bîrjoveanu[2]
[1]*Department of Computer Science, "Al.I.Cuza" University of Iaşi, Iaşi, Romania*
[2]*Continental Automotive, Iaşi, Romania*
cbirjoveanu@info.uaic.ro, mbirjoveanu@gmail.com

Keywords: Electronic Commerce Security, Complex Transactions, Fair Exchange, Security Protocols.

Abstract: In this paper, we define the concept of complex transaction as a combination in any form of aggregate and optional transactions. Even if there are many multi-party fair exchange protocols with applications in buying digital goods, digital signature of contracts and certified e-mail, no one can be used to solve our problem: complex transactions where a customer wants to buy several physical products from different merchants, providing fair exchange while preserving atomicity. In this paper, we propose the first fair exchange e-commerce protocol for complex transactions in that the customer wants to buy several different physical products from different merchants. Our protocol uses as building block the fair exchange internet payment protocol (FEIPS) for physical products that considers only one customer and one merchant. Also, our protocol provides effectiveness, timeliness, non-repudiation, integrity and confidentiality of data exchanged between the parties.

1 INTRODUCTION

In order to assure security, e-commerce protocols must satisfy some fundamental requirements: *confidentiality*, *authentication* and *non-repudiation*. Nowdays, in the electronic commerce, it is usually that a customer wishes to buy a pack of products/services composed of several products (physical or digital)/services from different merchants. In this type of e-commerce transactions, the customer is interested in buying all products from the pack or no product at all, namely *aggregate/atomic transactions*. For flexibility, in the *optional transactions*, the customer wants to buy exactly one product from many merchants, and for this, he specifies in his request more possible products according to his preferences but from this options only one will be committed. We will refer to the combination in any form of aggregate and optional transactions as *complex transactions*.

In e-commerce protocols, *fair exchange* is another essential property. For complex transactions, an e-commerce protocol in that a customer wishes to buy many products from different merchants assures fair exchange if:

- for any optional transaction from the complex transaction, the customer obtains exactly one digital receipt for the product's payment, and
- for any aggregate transaction from the complex transaction, the customer obtains the digital receipts for the payments of all products,

and each merchant obtains the corresponding payment for the product, or none of them obtains nothing. In a complex transaction, the issues that can appear are when in a aggregate transaction some products can be successfully acquired, but the others not, or when in an optional transaction more than one product is successfully acquired. In this cases, even if each corresponding merchant gets the payment for his product, fair exchange is not ensured.

To achieve fair exchange, the proposed protocols are based on a Trusted Third Party (TTP) that can be inline, online or offline. The protocols that are based on inline TTP (that is used in every message) or on online TTP (that is used in every protocol instance) are not efficient because TTP becomes a bottleneck. Using offline TTP (that is involved in a protocol instance only when an exception appears) removes the disadvantage mentioned above. One goal of our protocol is to obtain the fair-exchange requirement in complex transactions using an offline TTP.

In the literature there are many e-commerce protocols that consider only one customer and one merchant, most of them for buying digital products and only few for buying physical products.

Even if there are many multi-party fair exchange

V. Bîrjoveanu C. and Bîrjoveanu M.
An Optimistic Fair Exchange E-commerce Protocol for Complex Transactions.
DOI: 10.5220/0006853501110122
In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications (SECRYPT 2018)*, pages 111-122
ISBN: 978-989-758-319-3

protocols with applications in buying digital goods, digital signature of contracts and certified e-mail, no one can be used to solve our problem: complex transactions where a customer wants to buy several physical products from different merchants, providing fair exchange.

Our Contribution. In this paper, we propose the first fair exchange e-commerce protocol for complex transactions in that the customer wants to buy several different physical products from different merchants. Our protocol uses as building block the fair exchange internet payment protocol (FEIPS) for physical products proposed in (Djuric and Gasevic, 2015). The FEIPS protocol considers only one customer and one merchant. In our scenario, a complex transaction is a combination in any form of aggregate and optional transactions. Also, our protocol provides effectiveness, timeliness, non-repudiation, integrity and confidentiality of data exchanged between the parties.

The paper is structured as follows: section 2 gives application examples of our protocol, section 3 defines security requirements. Section 4 discusses related work, section 5 briefly presents the FEIPS protocol. Our protocol is presented in section 6. Section 7 contains the security analysis of the proposed protocol. A comparative analysis is provided in section 8 and section 9 contains the conclusion.

2 APPLICATIONS TO B2B/B2C SCENARIOS

Our protocol has use cases in Business to Consumer (B2C) and Business to Business (B2B) scenarios. For a B2B scenario, the customer is the Electron company that manufactures electronic boards for different purposes, on request from his clients. To plan its business, Electron uses an online catalog from where it can buy several electronic components from different merchants denoted by $M1, M2, M3$, e.t.c. From the online catalog, Electron can select products like: resistors (R), capacitors (C), integrated circuits (IC), cables, connectors, printed circuit boards (PCB) and so on. Electron wants to start the production of a new electronic board and therefore wants to prepare its order in form of an e-commerce complex transaction as follows: (100R of $10k\Omega$ from $M1$ or 70R of $20k\Omega$ from $M2$) and (50C of 100mF from $M3$ or 100C of 70mF from $M4$) and 70 connectors $DB35$ type from $M5$ and 30PCB from the $M6$. The complex transaction is composed from an aggregate transaction and two optional transactions. For the first optional transaction if Electron can not acquire 100R of $10k\Omega$ from $M1$ due to lack of stock or delay in delivery time, then its second option is taken into consideration to acquire 70R of $20k\Omega$ from $M2$. To start the production, Electron needs all types of components specified in its request, so a partial combination (e.g. 100R of $10k\Omega$, 100C of 70mF and 30PCB, but without 70 connectors $DB35$ type) is not useful for him. For an optional transaction, Electron must not acquire more then one product (e.g. for the first optional transaction he must not acquire both 100R of $10k\Omega$ and 70R of $20k\Omega$) because then he will remain with unnecessary products. For example, a pack of products that solves the customer's options is: 100R of $10k\Omega$, and 100C of 70mF, and 70 connectors $DB35$ type and 30PCB.

A similar scenario can be used in B2C applications. In this case, the customer is a person that likes electronics and wants to build an electronic hobby kit, and for this he uses the online catalog to order the needed components.

3 SECURITY REQUIREMENTS

In what follows, we will discuss the security requirements we want to achieve in our optimistic fair exchange protocol for complex transactions: *effectiveness*, *fairness*, *timeliness*, *non-repudiation* and *confidentiality*. These requirements are stated and analyzed in (Ferrer-Gomila et al., 2010), (Liu et al., 2011) for certified electronic mail protocols, (Draper-Gil et al., 2013) for contract signing protocols, and (Djuric and Gasevic, 2015) for electronic payment protocols for physical products.

Effectiveness requires that if every party involved in the complex transactions protocol behaves honestly, does not want to prematurely terminate the protocol, and no communication error occurs, then the customer receives his expected digital receipts from merchants, and the merchants receive their payments from the customer, without any intervention of Trusted Third Party (TTP). This is a requirement for *optimistic protocols*, where TTP intervenes only in case of unexpected situations, such as a network communication errors or dishonest behavior of one party.

Fairness for complex transactions requires:

- for any optional transaction from the complex transaction, the customer obtains exactly one digital receipt for the product's payment, and
- for any aggregate transaction from the complex transaction, the customer obtains the digital receipts for the payments of all products,

and each merchant obtains the corresponding payment for the product, or none of them obtains nothing.

This requirement corresponds to the *strong fairness* requirement stated in (Asokan, 1998).

Timeliness requires that any party involved in the complex transactions protocol can be sure that the protocol execution will be finished at a certain finite point of time, and that after the protocol finish point the level of fairness achieved cannot be degraded.

Non-repudiation in the complex transaction protocol requires that neither the customer nor any of merchants can deny their involvement in the complex transaction.

Confidentiality in the complex transaction protocol requires that the content of messages sent between participating parties is accessible only to authorized parties.

4 RELATED WORK

Until now there are protocols proposed for the payment for physical products, that provide fair exchange and consider only one customer and one merchant (Birjoveanu, 2015),(Djuric and Gasevic, 2015),(Alaraj, 2012),(Li et al., 2006),(Zhang et al., 2006).

There are known many multi-party fair exchange protocols proposed with applications in e-commerce transactions for buying digital goods (Liu, 2009), digital signature of contracts (Draper-Gil et al., 2013), (Mukhamedov and Ryan, 2008), certified e-mail (Zhou et al., 2005) and non-repudiation (Yanping and Liaojun, 2009), (Onieva et al., 2009). Despite great variety of multi-party fair exchange protocols proposed until now, there is no solution to address our problem: complex transactions where a customer wants to buy several physical products from different merchants, providing fair exchange.

From all solutions for multi-party fair exchange, we distinguish some of them (Liu, 2009), (Draper-Gil et al., 2013), (Yanping and Liaojun, 2009) that solve related problems with our problem, but in other scenarios.

The scenario most closest to our scenario is the one proposed by (Liu, 2009), where in an aggregate transaction a customer wants to buy several digital products from different merchants. The solution from (Liu, 2009) can not be applied to our problem because in our problem we want to obtain fair exchange between payments for physical products and digital receipts for physical products. Also, in (Liu, 2009) are taken into consideration only aggregate transactions, but optional transactions are not considered, and timeliness requirement is not assured.

In (Draper-Gil et al., 2013), a multi-two party contract signing protocol is proposed, where a consumer and many providers want to sign a contract pairwise. The solution from (Draper-Gil et al., 2013) assures weak fairness and can not be applied to our problem because it is applied in a contract signing scenario that is different from our scenario. Weak fairness requires that all parties receive the expected items, or all honest parties will have enough evidence to prove that they have behaved correctly in front of an arbiter.

In (Yanping and Liaojun, 2009), an optimistic multi-party non-repudiation protocol is proposed, which allows the sender to exchange different messages with multiple recipients for non-repudiation evidences. The solution from (Yanping and Liaojun, 2009) can not be applied to our problem because it does not take into consideration atomicity and is applied in a non-repudiation scenario, while our scenario requires atomicity and fair exchange between payments for physical products and digital receipts for physical products.

As a result, from all known solutions for multi-party fair exchange problems related with our problem, no one can be applied to solve our problem.

5 THE FEIPS PROTOCOL

Our protocol is based on the fair exchange internet payment protocol (FEIPS) for the payment of physical products, proposed in (Djuric and Gasevic, 2015), that is why we briefly describe it. This protocol considers only one customer and one merchant. The FEIPS protocol involves the customer (the payment Web segment), the merchant, the payment gateway and the bank.

The payment Web segment is a soft digitally signed by payment gateway and it is automatically downloaded by customer from merchant before the protocol execution. The role of the payment Web segment is to perform customer's side protocol actions.

The FEIPS protocol consists of three sub-protocols: the setup sub-protocol, the exchange sub-protocol and the resolution sub-protocol. All messages transmitted in protocol are protected using hybrid encryption. In the setup sub-protocol, the customer sends his session public key to the merchant, that replies with an unique identifier of the transaction. In the exchange sub-protocol, the customer sends to the merchant a payment message encrypted with the payment gateway's public key and the purchase order. If the merchant agrees with the order received from the customer, then he sends the payment message to the payment gateway. On reception of the payment message, the payment gateway decrypts it, verifies the payment information and if the customer is authorized to use the card. If all checks are successfully, the payment gateway sends the payment message to the bank. Depending on the customer's account balance, the bank makes or not the transfer and provides an appropriate response to the payment gateway that forwards it to the merchant. Finally, the merchant sends the response to the customer. The response is digitally signed by the payment gateway and it means the digital receipt of the payment for the ordered product. The fair exchange in the FEIPS protocol is defined w.r.t. exchange of electronic payment for a digital receipt. The resolution sub-protocol is used to provide fair exchange for cases in that the customer pays, but he does not receive the digital receipt because the merchant behaves dishonest or a network communication error appears. In the resolution sub-protocol, the customer sends to the payment gateway (TTP) a request for response, and the payment gateway sends him the response. (Djuric and Gasevic, 2015) have shown that the FEIPS protocol ensures fairness and confidentiality using the AVISPA tool (Vigano, 2006).

6 THE COMPLEX TRANSACTIONS PROTOCOL

In the complex transactions protocol (*CTP*), we consider that one customer can buy products in complex transactions from many merchants. *CTP* has the following participants: the customer (the payment Web segment), the merchants, the payment gateway and the bank.

Table 1 presents the notations used in the description of *CTP*. We use hybrid encryption with the same meaning as in (Djuric and Gasevic, 2015). Hybrid encryption $\{m\}_{PubKA}$ of the message m with the public key *PubkA* means $\{m\}_K$, $\{K\}_{PubKA}$: the message m is encrypted with an AES session symmetric key K, which is in turn encrypted using *PubKA*. If two parties use the session symmetric key K in a hybrid encryption, then they will use K to hybrid encryption of all the messages that will be transmitted between them for the remainder of session. We make some considerations about the communication channels we will use in *CTP*. (Ferrer-Gomila et al., 2010) consider three types of communication channels: *operational*, *resilient* and *unreliable*. Operational communication channels (messages are correctly received in a finite amount of time) impose a not realistic assumption for the current networks. In *CTP*, we consider resilient communication channels (messages can be delayed but not lost) between *PG* and *C*, respectively between *PG* and *M*, that is similarly with assumption from (Djuric and Gasevic, 2015). The other communication channels are unreliable (messages can be lost).

6.1 The Preparation Phase

Before *CTP* execution, a preparation phase is needed. The customer is browsing through the online catalog where the products from merchants are posted. After the customer decides the products pack he wants to buy and the options/alternatives for each product from the pack, he clicks a "submit" button on the online catalog and the download of the payment Web segment is started. The payment Web segment has the same role as in the FEIPS protocol, thus we use the term *customer* and *payment web segment* interchangeable, the context indicating which of them we refer to. The payment Web segment is a software digitally signed by payment gateway that runs on the customer's computer. The payment Web segment requires from customer the credit card information and a challenge code that will be used to authenticate and authorize the customer for using the credit card. For each subtransaction involved in the complex transaction (corresponding to the products the customer wishes to buy), the payment Web segment generates a RSA session public/private key pair for customer. We consider that the payment Web segment has the digital certificates for the public keys of each merchant and payment gateway. Also, each merchant/payment gateway has the digital certificate for the payment gateway/each merchant's public key.

For an aggregate transaction, we define the *aggregation* operator, denoted by $\wedge$, as follows: $Pid_1 \wedge \ldots \wedge Pid_k$ meaning that C wishes to buy exactly k products with product's identifiers $Pid_1, \ldots, Pid_k$. For an optional transaction, we define the *option* operator, denoted by $\vee$, as follows: $Pid_1 \vee \ldots \vee Pid_k$ meaning that C wishes to buy a product that is exactly one of the products with product's identifiers $Pid_1, \ldots, Pid_k$, where the apparition order of the product's identifiers is the priority given by C. This means that C wishes first of all to buy the product Pid_1, but if this is not possible, his second option is Pid_2, and so on.

From the choices of C describing the sequence of products he wishes to buy, we build a tree over the product identifiers selected by C using $\wedge$ and $\vee$ operators. To represent the tree, we use the *left-child, right-sibling representation* in that each internal node corresponds to one of the above operators or to an identifier, while each leaf node corresponds to an identifier. Each node of the tree is represented by a structure with the following fields: *info* for storing the useful information (identifier or one of the operators), *left* for pointing to the leftmost child of node, and *right* for pointing to the sibling of the node immediately

Table 1: Notations used in the protocol description.

Notation	Interpretation
C, PG, Mi	Identity of Customer, Payment Gateway, Merchant i, where $1 \leq i \leq n$
$PubKA$, $\{m\}_{PubKA}$	RSA public key of the party A, hybrid encryption of the message m with $PubKA$
$h(m)$	The digest of the message m obtained by applying of a hash function h (SHA-2)
$SigA(m)$	RSA digital signature of A on $h(m)$
$A \rightarrow B$:m	A sends the message m to B

to the right. The access to tree is realized trough the root. An example of tree derived from the complex transaction from section 2, is shown in Figure 1.

Pid_1 corresponds to R of $10k\Omega$, Pid_2 to R of $20k\Omega$, Pid_3 to C of 100mF, Pid_4 to C of 70mF, Pid_5 to connectors $DB35$ type and Pid_6 to PCB. The root node has $\wedge$ operator as *info*. The root does not have any right sibling and its children are two nodes having $\vee$ operator as *info* and the nodes with the *info* Pid_5 and Pid_6.

Next, we will describe the subtransaction protocol (*STP*) in which the customer *C* buys a certain physical product from a certain merchant *M*.

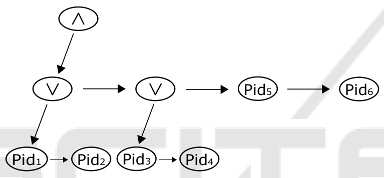

Figure 1: Tree describing the customer's choices in left-child, right-sibling representation.

6.2 The Subtransaction Protocol

CTP uses *STP*. *STP* is based on FEIPS protocol, but *STP* has a significant difference from the FEIPS protocol, namely, *STP* consists of four sub-protocols: the setup sub-protocol, the exchange sub-protocol and two resolution sub-protocols.

Next, we will describe *STP*'s sub-protocols messages that are graphically represented in Figure 2.

6.2.1 Setup Sub-protocol

Message 1: $C \rightarrow M$:$\{PubKC\}_{PubKM}$

In the first message, the payment Web segment sends to *M* the customer's session public key *PubKC* generated in the preparation phase. The message is hybrid encrypted with *M*'s public key *PubKM*.

Message 2: $M \rightarrow C$:$\{Sid, SigM(Sid)\}_{PubKC}$, or $M \rightarrow C$:$\{Resp, Sid, SigM(Resp, Sid)\}_{PubKC}$, where $Resp = ABORT$

Upon receiving the first message, *M* generates a fresh random number *Sid* that will be used as an unique identifier of the subtransaction. *M* sends to *C*, *Sid* and his signature on *Sid*, both encrypted with *PubKC*. If upon receiving the first message, for any reason, *M* does not want to continue the setup sub-protocol, then he will send to *C* a signed response *Resp* with the value *ABORT*. Upon receiving the message 2, *C* decrypts it and authenticates *M* by checking *M*'s signature.

6.2.2 Exchange Sub-protocol

Message 3: $C \rightarrow M$:$\{PM, PO\}_{PubKM}$

If *C* receives a message 2 that does not contain a response *ABORT* and he authenticates *M*, then in the third message, the payment Web segment sends to *M* a payment message *PM* and a purchase order message *PO*, both encrypted with *PubKM*.

$PM = \{PI, SigC(PI)\}_{PubKPG}$

PM is build by payment Web segment by encrypting with *PG*'s public key of the payment information *PI* and the customer's signature on *PI*. The encryption of *PI* with *PG*'s public key assures that *PI* cannot be found out by *M*.

$PI = CardN, CCode, Sid, Amount, PubKC, NC, M$

PI contains the data provided by the user: card number *CardN* and a challenge code *CCode* issued by bank. The challenge code is provided to user by bank via SMS an it has a minimum length of four characters. Also, *PI* contains *Sid*, the amount *Amount*, *PubKC*, a fresh nonce *NC* generated by *C*, and the merchant's identity *M*.

$PO = OI, SigC(OI)$

PO is build from the order information *OI* provided by *C* and the signature of *C* on *OI*. *OI* contains the order description for the product *OrderDesc*, *Sid*, and *Amount*.

$OI = OrderDesc, Sid, Amount$

Upon receiving the message 3, *M* decrypts it and checks the signature of *C* on *OI*. If *M* agrees with *PO* received from *C*, then he stores *PO* as an evidence of *C*'s order and sends the message 4 to *PG*. Otherwise, *M* sends to *C* a message containing a response *ABORT* for aborting the subtransaction.

Message 4: $M \rightarrow PG$: $\{PM, SigM(Sid, PubKC, Amount)\}_{PubKPG}$

In the message 4, *M* sends to *PG* the payment message *PM* and his signature on the subtransaction identifier, *C*'s public key and amount. Upon receiving the message 4, *PG* decrypts it, checks *C*'s signature on *PI* and checks if *C* is authorized to use the card by checking if the combination of *CardN* and *CCode* is valid. If these checks are successfully passed, then also *C* proves as being the owner of the public key *PubKC*. *PG* checks *M*'s signature, and if the checking is successfully, then it has the confirmation that both *C* and *M* agreed on *Sid*, *PubKC* and *Amount*. Also, *PG* checks the freshness of *PubKC*, *Sid* and *NC* to avoid any replay attack from dishonest merchants. If some check fails, then *PG* sends to *M* a response *ABORT* for aborting the subtransaction. If all checks are successfully, *PG* sends the payment message to the bank. The bank checks *C*'s account balance, and if it is enough, then the bank makes the transfer in *M*'s account providing an response *YES* ($Resp = YES$) to *PG* that forwards it to *M* in the message 5. Otherwise, if checking *C*'s account balance fails, then also the transfer fails and the bank provides an response *ABORT* ($Resp = ABORT$) to *PG* that forwards it to *M* in the mesage 5. Also, *PG* stores the messages 4 and 5 in its databases as an evidence of the subtransactions details.

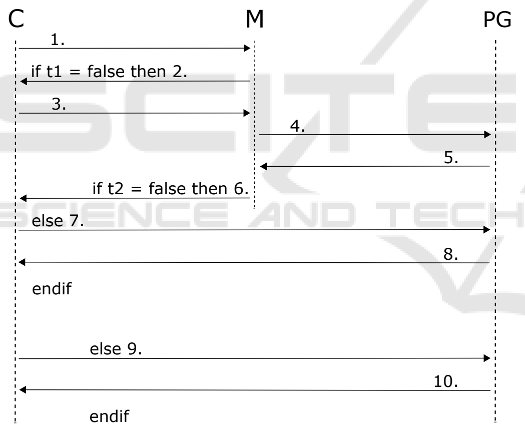

Figure 2: *STP* mesage flow.

Message 5: $PG \rightarrow M$:
$\{Resp, Sid, SigPG(Resp, Sid, Amount, NC)\}_{PubKM}$

Upon receiving the message 5, *M* decrypts it and sends to *C*, in the message 6, the response and *PG*'s signature on response both encrypted with *PubKC*.

Message 6: $M \rightarrow C$:
$\{Resp, Sid, SigPG(Resp, Sid, Amount, NC)\}_{PubKC}$

C decrypts message 6 and checks *PG*'s signature. If checking is successfully, then *C* has the guarantee of the response's authenticity (it is from *PG*) and it corresponds to the current subtransaction. The presence in response of *Sid*, *Amount* and *NC* proves the response's freshness and it is not replayed by a dishonest merchant. A response with $Resp = YES$ means that the subtransaction successfully finished and the content of message 6 ($Resp, Sid, SigPG(Resp, Sid, Amount, NC)$) is a digital receipt for the payment of product. A response with $Resp = ABORT$ means that the content of message 6 is a proof of the subtransaction's abort.

6.2.3 Resolution 1 Sub-protocol

If *C* initiates *STP*, but he does not receive any message from *M* or receives an invalid message 2 from *M*, then the current subtransaction's state is undefined. If we consider only the current subtransaction, then *STP* does not give any benefit to any party: the payment is not sent to *M* and no digital receipt to *C* as well. However, this case must be solved if we reason that the current subtransaction belongs to an aggregate transaction that contains subtransactions that already successfully finished, as we will see in *CTP*. In this case, a timeout interval $t1$ (e.g. in the order of seconds or minutes) is defined, in which *C* waits the message 2 from *M*. If $t1$ expires and *C* does not receive the message 2 from *M* or receives an invalid message, then *C* initiates the resolution 1 sub-protocol with *PG* (the messages 7 and 8) to receive a response w.r.t. the current subtransaction.

Message 7: $C \rightarrow PG$:$\{PubKC)\}_{PubKPG}$

C sends to *PG* in the message 7 a response request for the current subtransaction. Upon reception, *PG* decrypts the message, checks that no response has been generated for *PubKC*, generates a subtransaction identifier *Sid* and sends to *C* in message 8 a signed *ABORT* response. Also, *PG* stores the response in its databases.

Message 8: $PG \rightarrow C$:
$\{Resp, Sid, SigPG(Resp, Sid)\}_{PubKC}$, where $Resp = ABORT$

Upon receiving the message 8, C decrypts it and authenticates the response by checking *PG*'s signature.

6.2.4 Resolution 2 Sub-protocol

If *C* sends the payment in message 3, but he does not receive message 6, or receives an invalid message from *M*, then an unfair case appears: *C* sends the payment and it was processed, but *C* did not receive any response. In this case, a timeout interval $t2$ (e.g. in the order of seconds or minutes) is defined, in which *C* waits message 6 from *M*. If $t2$ expires and *C* does not receive message 6 from *M* or receives an invalid message, then *C* initiates the resolution 2 sub-protocol with *PG* (the messages 9 and 10) to receive a response w.r.t. the current subtransaction.

Message 9: $C \to PG$:$\{Sid, Amount, NC, PubKC,$
$SigC(Sid, Amount, NC, PubKC)\}_{PubKPG}$

Upon receiving the message 9, *PG* decrypts it and checks if a response has been generated for the entry *Sid*, *Amount*, *NC* and *PubKC*. If *PG* finds in its database a response for the entry above, and checking the signature of *C* using *PubKC* is successfully, then it sends to *C* the response in message 10. Otherwise, if *PG* does not find a response for the entry above, then *PG* sends to *C* an *ABORT* response in message 10 and stores it in its database. If *PG* receives the payment in message 4 (sent by *M* afterward or replayed by *M*) that contains the information from the entry above, then *PG* sends also an *ABORT* response to *M*.

Message 10: $PG \to C$:
$\{Resp, Sid, SigPG(Resp, Sid, Amount, NC)\}_{PubKC}$

6.3 CTP Description

In the complex transaction protocol we proposed, a subtransaction s, denoted by $STP(C, M_i, Pid_i)$, is an instance of *STP* in which *C* buys the physical product with Pid_i identifier from the merchant M_i. We define $St(s)$ the state of the subtransaction s as being the content of one of the messages 2, 6, 8 or 10 (identical with the message 6) that the customer will receive in s. More exactly, $St(s)$ is:

- $(Resp, Sid, SigM(Resp, Sid))$, where $Resp = ABORT$, or
- $(Resp, Sid, SigPG(Resp, Sid, Amount, NC))$, where $Resp \in \{YES, ABORT\}$, or
- $(Resp, Sid, SigPG(Resp, Sid))$, where $Resp = ABORT$.

For $St(s)$ a state of the subtransaction s, we denote by $St(s).Resp$ the response ($Resp$) in $St(s)$, and by $St(s).Sig$ the signature in $St(s)$.

We define $Ns(p)$ - the state of the node p as a sequence of subtransaction states $St(s_1) \dots St(s_m)$ corresponding to the product defined by p. For a node p, $Ns(p)$ is calculated depending on the $p \to info$ as follows:

- if $p \to info = Pid_i$, where $1 \leq i \leq n$, then $Ns(p) = St(STP(C, M_i, Pid_i))$. For simplicity, we consider that M_i is the merchant that sells the product with Pid_i identifier, where $1 \leq i \leq n$.
- if $p \to info = \vee$, then

$$Ns(p) = \begin{cases} Ns(l), & \text{if } \exists\, l \text{, the leftmost child} \\ & \text{of } p \text{ such that } St(s).Resp = \\ & YES\text{, for all } St(s) \in Ns(l) \\ Ns(r), & \text{otherwise} \end{cases}$$

where r is the rightmost child of p.

The node p corresponds to $\vee$ operator w.r.t. the customer's choices and this preferences are prioritized by appearance in the child nodes of p from left to the right. $Ns(p)$ is the node state of the leftmost child of p, denoted $Ns(l)$, for which all subtransactions from $Ns(l)$ have successfully finished *STP*. Otherwise, if all subtransactions from node states of all children of p are aborted, then $Ns(p)$ is the node state of the rightmost child of p.

- if $p \to info = \wedge$, and $c_1, \dots, c_k$ are all children of p, then we have two cases:
  1. if $St(s).Resp = YES$, for any $St(s)$ from $Ns(c_j)$, for any $1 \leq j \leq k$, then $Ns(p) = Ns(c_1) \dots Ns(c_k)$.
  2. otherwise, let be c_j, where $1 \leq j \leq k$, the leftmost child of p with $Ns(c_j) = St(s_{j1}) \dots St(s_{jm})$ such that $St.(s_{jl}).Resp = ABORT$, for all $1 \leq l \leq m$. In this case, $Ns(p) = Ns(c_1) \dots Ns(c_j)$. Even if the subtransactions states from $Ns(c_1), \dots, Ns(c_{j-1})$ are YES (that means that the subtransactions from $Ns(c_1), \dots, Ns(c_{j-1})$ have successfully finished *STP*), the aborted subtransactions $s_{j1}, \dots, s_{jm}$ from $Ns(c_j)$ lead to aborting the entire aggregate transaction corresponding to p. That is why all subtransactions states from $Ns(c_1), \dots, Ns(c_{j-1})$ will be aborted. Thus, will set $St(s).Resp = ABORT$, for any $St(s) \in Ns(c_r)$, for any $1 \leq r \leq j-1$.

  Because the node p corresponds to $\wedge$ operator, $Ns(p)$ is the sequence of node states of p's children. For efficiency, the sequence of node states of p's children is calculated until c_j the leftmost child of p for that $Ns(c_j)$ contains only aborted subtransaction states.

Thus, $Ns(p)$ contains a sequence of subtransaction states in which either all subtransactions successfully finished *STP* or all subtransactions are aborted.

CTP, described in Table 2, recursively calculates $Ns(t)$ (t is the root of the tree derived from the customer's choices), traversing the tree in a similar manner with depth-first search. For any node p of the tree, we use a *child* array to store the node states of all children of p.

At the lines 2-3, the protocol computes $Ns(p)$ for a node p, depending on the node state of the left most child of p. For a node p with a least two children, the while loop (the 5-12 lines) computes the node state of any child of p except the left most one. We remark that way in which node state is computed is essential to obtain the fair exchange and atomicity of

Table 2: Complex transactions protocol.

```
CTP(t)
1.  if (t → left ≠ NULL)  child[0] = CTP(t → left);
2.  if ((t → info = ∨ and $St(s).Resp = YES$, for all $St(s)$ from child[0]) or
3.      (t → info = ∧ and $St(s).Resp = ABORT$, for all $St(s)$ from child[0]))  Ns(t) = child[0];  return Ns(t);
4.  j = 1;  k = t → left → right;
5.  while (k ≠ NULL)
6.          child[j] = CTP(k);
7.          if (t → info = ∨ and $St(s).Resp = YES$, for all $St(s)$ from child[j])  Ns(t) = child[j];  return Ns(t);
8.          if (t → info = ∧ and $St(s).Resp = ABORT$, for all $St(s)$ from child[j])
9.                  for (c = 0; c ≤ j; c = c + 1)  Ns(t) = Ns(t)child[c]; end for
10.                 AggregateAbort(Ns(t));  return Ns(t);
11.         k = k → right;  j = j + 1;
12. end while
13. if (t → info = $Pid_i$)  Ns(t) = $St(STP(C, M_i, Pid_i))$;  return Ns(t);
14. else if (t → info = ∨)  k = t → left;
15.                         while (k → right ≠ NULL)  k = k → right; end while
16.                         Ns(t) = Ns(k);  return Ns(t);
17.     else  for (c = 0; c ≤ j - 1; c = c + 1)  Ns(t) = Ns(t)child[c]; end for
18.             return Ns(t);
19.     end if
20. end if
```

a complex transaction (lines 7-10): if an aborted subtransaction/sequence of subtransactions leads to aborting the entire aggregate transaction, but some subtransactions from the aggregate transaction successfully completed *STP*, then the ones that are successfully must also be stored in the node state corresponding to ∧ operator (line 9) and afterwards aborted by applying *AggregateAbort* sub-protocol (line 10). We will describe *AggregateAbort* sub-protocol in the section 6.3.1.

At line 13, the protocol computes $Ns(p)$ for a node p with a product identifier as *info*.

The node state for a node that corresponds to ∨ operator, for which all subtransactions states from all its children are aborted, is computed at lines $14-16$. The node state for a node that corresponds to ∧ operator for that all subtransactions states from all its children successfully completed *STP*, is computed at lines $17-18$.

6.3.1 AggregateAbort Sub-protocol

As we discussed in the previous section, in *CTP*, there may be cases in which an aborted subtransaction/sequence of subtransactions leads to aborting the entire aggregate transaction, but some subtransactions from the aggregate transaction successfully completed *STP*. As a result, an unfair case occurs for C: the entire aggregate transaction is not successful, but C has paid for certain products. For example, for a node p that corresponds to ∧ operator, *CTP* computes the node state $Ns(p) = St(s_1)\dots St(s_k)St(s_{k+1})\dots St(s_m)$, where $St(s_i).Resp = YES$ for any $1 \le i \le k$ and $St(s_j).Resp = ABORT$ for any $k+1 \le j \le m$. The entire complex transaction corresponding to the node p is not completed successfully because the component subtransactions $s_{k+1},\dots,s_m$ are aborted, but in the same complex transaction C paid for the products involved in the subtransactions $s_1,\dots,s_k$. So, the fairness will be obtained by applying *AggregateAbort*($Ns(p)$) sub-protocol in that entire aggregate transaction will be aborted. Next, we will describe *AggregateAbort*($Ns(p)$) sub-protocol.

To solve the unfair case mentioned above, the payment Web segment initiates the *AggregateAbort*($Ns(p)$) sub-protocol by sending to *PG* in the message 11 a customer request to abort $Ns(p)$. The message 11 is build from $Ns(p)$ and the customer's signature on $Ns(p)$, both encrypted with *PubKPG*.

Message 11: $C \rightarrow PG$:
$$\{Ns(p), SigC(Ns(p))\}_{PubKPG}$$

Upon receiving the message 11, *PG* decrypts it, obtains the sequence of subtransaction states $St(s_1)\dots St(s_m)$ in $Ns(p)$, and checks the signature of C on $Ns(p)$ to be sure that this request comes from C. *PG* checks the signatures involved in all subtransaction states from $Ns(p)$ as follows:

- *PG* checks *M*'s signature, for any $St(s) \in Ns(p)$, such that $St(s) = (Resp, Sid, SigM(Resp, Sid))$ with $Resp = ABORT$;
- *PG* searches into its database the appropriate entry for $St(s)$ and checks his signature, for any $St(s) \in Ns(p)$, such that $St(s) = (Resp, Sid, SigPG(Resp, Sid, Amount, NC))$ with $Resp \in \{YES, ABORT\}$, or $St(s) = (Resp, Sid, SigPG(Resp, Sid))$ with $Resp = ABORT$.

If all checks are successfully passed, then *PG* sends to the bank the customer's request. The bank aborts any subtransaction's state $St(s) = (Resp, Sid, SigPG(Resp, Sid, Amount, NC)) \in Ns(p)$, such that $St(s).Resp = YES$ by canceling the transfer corresponding to $St(s)$ from customer's account into merchant's account, and updating $St(s)$ by:

1. $OldSt(s) = St(s)$,
2. $St(s).Resp = ABORT$ and
3. $St(s).Sig = SigPG(St(s).Resp, Sid, Amount, NC, OldSt(s))$.

By updating $St(s)$ as above, $St(s)$ becomes aborted, and *PG*'s signature is updated including the old subtransaction state. In this way, any party (C, M_i, or *PG*) can check afterward *PG*'s signature from the updated $St(s)$ to ensure that the subtransaction s that successfully completed *STP* has been authorized aborted. Thus, we remark that no party can have 2 different independent subtransaction states for the same subtransaction.

The bank sends the new $Ns(p)$ computed above (in that any subtransaction state is aborted) to *PG* that forwards it to *C* in message 12.

Message 12: $PG \rightarrow C{:}\{Ns(p), SigPG(Ns(p))\}_{PubKC}$

PG stores message 12 in its database for transaction's evidence.

Also, in message 13, *PG* sends simultaneously the subtransaction state $St(s) \in Ns(p)$ just aborted by the above procedure, to each merchant *M* involved in the subtransaction s.

Message 13: $PG \rightarrow M{:}\{St(s)\}_{PubKM}$

7 SECURITY ANALYSIS

In this section, we will analyze the security requirements stated in section 3 for our *CTP*.

7.1 Effectiveness

If every party involved in the complex transaction protocol from Table 2 behaves according to the protocol's steps, does not want to prematurely terminate the protocol and there are no network communication delays/errors, then our protocol assures that the customer receives the digital receipts from merchants, and each merchant receives his corresponding payment from the customer without TTP involvement. Therefore, our protocol meets the effectiveness requirement.

7.2 Fairness

In our *CTP*, the fairness may not be insured only for *C*. There are some arguments for this. The payment Web segment performs customer's side protocol actions and is a soft digitally signed by *PG* that is a trusted party. So, any corruption of the payment Web segment by user or *M* is impossible. Moreover, *C* receives the digital receipt for payment only after *C* sends the payment to *M*. We remark that *C* can not cheat *M* by sending an inappropriate payment message *PM* directly to *PG*, because *PM* is sent first by *C* to *M* in message 3, and after *M* agrees with *PO*, then *M* sends *PM* to *PG* in message 4. As we saw in section 6.2.2, if all checks of *PG* w.r.t message 4 are successfully, then it has the confirmation that both *C* and *M* agreed on the subtransaction data (*Sid*, *PubKC* and *Amount*). From these arguments, *M* is in a more advantageous position than *C* in the exchange of the payment for the digital receipt and *C* is the only party that could have losses.

CTP uses *STP*. To obtain fairness in *CTP*, a necessary but not sufficient condition is to obtain fairness in *STP*.

STP is based on the FEIPS protocol, with the major difference that *STP* has the sub-protocol resolution 1 in addition to the FEIPS protocol. (Djuric and Gasevic, 2015) have shown that the FEIPS protocol ensures fairness using AVISPA tool for automated validation of large-scale Internet security protocols that provides four back-ends that implements different verification techniques (Vigano, 2006). More exactly, in (Djuric and Gasevic, 2015), the authors used for FEIPS's verification two back-ends from the AVISPA tool: Cl-Atse (Constraint-Logic-based Attack Searcher) - a model checker that uses constraint solving techniques (Turuani, 2006), and OFMC (On-the-Fly Model-Checker) - a model checker that uses symbolic techniques (Basin et al., 2005). In AVISPA, the fair exchange requirement between payment and digital receipt for the FEIPS protocol, is ensured by proving the authentication goals *PG authenticates C on PI* and *C authenticates PG on digital receipt*, or by finding attacks for both authentication goals. The fairness verification results obtained in AVISPA for the FEIPS protocol showed that there is no attack (Djuric and Gasevic, 2015).

However, the only case in *STP* in that the fairness for *C* is not insured is when *C* sends the payment in message 3, the payment is successfully processed, but he does not receive the digital receipt for payment in message 6, or receives an invalid message from *M*. This case may arise in the following scenarios: *M* sends the payment to *PG* in the message 4, but *M* does not receive message 5 from *PG*, or *M* receives message 5 but does not send the message 6 to *C*, or *M* receives the message 5 but sends an invalid message to *C*. The scenarios above can appear because *M* behaves dishonest or a network communication error occurs. In this case, *C* waits for the message 6 from *M* until the timeout interval $t2$ expires in which case *C* initiates the resolution 2 sub-protocol with *PG* to receive the digital receipt. After *PG* receives the message 9, he sends to *C* in message 10 the digital receipt for payment because the payment has been successfully processed and *PG* finds the digital receipt in its databases. We remark that if *C* initiates the resolution 2 sub-protocol and *M* sends the message 4 to *PG* afterward, the fairness is preserved because *PG* will send to both *C* and *M* an ABORT response as is mentioned in section 6.2.4. Thus, the unfair case is solved: *M* receives the payment and *C* the digital receipt, or *M* does not receives the payment and *C* receives an *ABORT* response. The resolution 1 sub-protocol from section 6.2.3 is necessary only when we take into consideration the complex transactions (in which is mandatory to have a defined state for each performed subtransaction), to solve the case in that in a subtransaction *C* initiates *STP*, but he does not receive any message from *M* or receives an invalid message 2 from *M*. This case can appear because *M* behaves dishonest or because of a network communication error. In this case, *C* waits for message 2 from *M* until the timeout interval $t1$ expires in which case *C* initiates the resolution 1 sub-protocol with *PG*, and *PG* sends to *C* an ABORT response for the current subtransaction. We also note that in the FEIPS protocol was not necessary to consider a resolution 1 sub-protocol as in section 6.2.3, because the FEIPS protocol take into consideration only one customer and one merchant. As a result of the discussion above, from the fact that the FEIPS protocol ensures fairness, we obtain that *STP* ensures fairness.

The fairness obtained in all subtransations from a complex transaction does not directly implies that fairness is ensured in entire complex transaction. So, in addition to fairness in *STP*, to obtain fairness in *CTP*, two requirements must be also ensured. First, for any optional transaction from the complex transaction, *C* obtains exactly one digital receipt for the payment of only one product, and corresponding merchant obtains the payment for the product, or none of them obtains nothing. The product is either an individual product, or an aggregate product that corresponds to a $\wedge$ operator. Also, the digital receipt is either an individual receipt corresponding to an individual product, or is a sequence of receipts corresponding to an aggregate product. As we have seen in Table 2, *CTP* ensures this requirement: the node state corresponding to $\vee$ operator is the node state of its leftmost child for which all subtransactions have successfully finished *STP*. Otherwise, if all subtransactions from node states of all children of the node $\vee$ are aborted, then its node state is the node state of the rightmost child of $\vee$. Secondly, for any aggregate transaction from the complex transaction, *C* obtains the digital receipts for the payments of all products, and each merchant obtains the corresponding payment for the product, or none of them obtains nothing. In *CTP*, the node state for a node corresponding to the $\wedge$ operator is computed as follows: if all subtransactions from all node states of all children of the node $\wedge$ have successfully finished *STP*, then the node state of $\wedge$ is the sequence of node states of its children. Otherwise, the node state of $\wedge$ is the sequence of node states of its children until to the leftmost child that contains only aborted subtransaction states in its node state. But, in this last case, an unfair case occurs for *C*: the entire aggregate transaction is not successful, but *C* has paid for certain products and he received the digital receipt for these. We note that this is the only case in that the fairness for *C* can be violated in *CTP*. In this case, as we have seen in *CTP*, the *AggregateAbort* sub-protocol is applied. More exactly, *C* (payment Web segment) initiates the *AggregateAbort* sub-protocol from section 6.3.1 to abort any subtransaction that successfully finished *STP* and that belongs to the unsuccessful aggregate transaction, by sending to *PG* (message 11) the node state corresponding to $\wedge$ operator. After checking the signatures from all received subtransaction's states, *PG* sends the customer's request to the bank that aborts all subtransactions that successfully finished *STP* (including the canceling the corresponding transfer) and that belongs to the unsuccessful aggregate transaction. The bank sends to *PG* the new node state coresponding to $\wedge$ operator where all subtransaction's states are aborted. Also, *C* and each merchant involved in these aborted subtransactions receive from *PG* (messages 12 and 13) the new subtransaction's states aborted. Thus, the unfair case for *C* is solved: the entire aggregate transaction is aborted, meaning that any subtransaction which belongs to the aggregate transaction is aborted.

As a result, fairness exchange of payment for digital receipt in complex transactions and complex transactions atomicity are preserved.

7.3 Timeliness

Any party can be sure that *CTP* execution will be finished at a certain finite point of time, for two reasons. First, we have introduced two timeout intervals in *STP* when *C* waits the message 2 from *M*, respectively, when *C* waits the message 6 from *M*. If these timeout intervals expires, then *C* initiates the resolution 1 sub-protocol with *PG*, respectively, *C* initiates the resolution 2 sub-protocol with *PG*. Secondly, if in *CTP*, a complex transaction contains subtransactions that successfully finished *STP* and also contains an aborted subtransaction, then the *AggregateAbort* sub-protocol is executed between *C* and *PG*, to abort any subtransaction from the complex transaction. The communication channel between *C* and *PG* is resilient, and as a result, *CTP* execution will be finished at a certain finite point of time. After the *CTP* finish point, the level of fairness achieved cannot be degraded. If after the finish point, *C* has the digital receipts of payments for products, then each merchant obtains also the corresponding payment for his product. On the other side, if each merchant involved obtains the payment for his product, then, after the protocol finish point, *C* gets the corresponding digital receipts from the messages 6, or messages 10. Moreover, if after the finish point, *C* has not received the digital receipts, then also the corresponding merchant does not get the payment; if each merchant involved has not get the payment, then *C* also does not receive the digital receipts.

7.4 Non-repudiation

In any subtransaction from *CTP*, *C* cannot deny its participation in subtransaction because he cannot deny its signature on the payment information *PI*. If *C* tries to deny its participation in a subtransaction, *PG* has in its database the evidence of *C*'s signature on *PI*. Also, no merchant can deny its participation in a subtransaction because he cannot deny its signature which he includes in the message 4. If a merchant tries to deny its participation in a subtransaction, *PG* has the evidence of the merchant's signature on the subtransaction identifier *Sid*, *C*'s public key *PubKC* and the subtransaction's amount *Amount*.

7.5 Confidentiality

Every message transmitted between parties involved in any subtransaction from *CTP* is hybrid encrypted with the public key of the receiver: every message is encrypted by sender with a symmetric key, and this symmetric key is encrypted with the receiver's public key. So, only the authorized receiver of a message can read the message's content. In particular, in any subtransaction from *CTP*, the confidentiality of the card number *CardN* between *C* and *PG* is obtained because *C* sends (in message 3) *CardN* hybrid encrypted with *PG*'s public key. Also, in any subtransaction from *CTP*, the confidentiality of the order description *OrderDesc* between *C* and *M* is obtained because *C* sends *OrderDesc* hybrid encrypted with *M*'s public key. As a result, *CTP* ensures the confidentiality requirement. This requirement is also satisfied for the FEIPS protocol only taken into consideration one customer and one merchant.

8 COMPARATIVE ANALYSIS

Table 3 provides a comparative analysis between the security requirements obtained by our protocol and the security requirements obtained by the most related solutions to our proposal.

9 CONCLUSIONS

In this paper, we proposed an optimistic fair-exchange e-commerce protocol for complex transactions. Our protocol is based on FEIPS protocol (that considers only one customer and one merchant) for which was formally proved fairness and confidentiality requirements using AVISPA. The main strengths of our protocol are: ensures strong fair-exchange and atomicity for complex transactions, uses payment gateway as offline TTP, provides timeliness, non-repudiation, integrity and confidentiality.

Table 3: Multi-party fair-exchange e-commerce protocols: a comparative analysis.

	Our protocol	Liu	Draper-Gil	Yanping
Scenario	Payment for physical products in complex transactions	Payment for digital products in aggregate transactions	Contract signing	Non-repudiation
Atomicity	Y	Y	Y	N
Effectiveness	Y	Y	Y	Y
Fairness	Strong	Weak	Weak	Strong
Timeliness	Y	N	Y	Y
Non-repudiation	Y	Y	Y	Y
Confidentiality	Y	N	Y	Y

Y=YES, N=NO

REFERENCES

Alaraj, A. (2012). Fairness in physical products delivery protocol. *International Journal of Computer Networks & Communications (IJCNC).*

Asokan, N. (1998). Fairness in electronic commerce. *PhD Thesis, University of Waterloo, Canada.*

Basin, D., Modersheim, S., and Vigano, L. (2005). Ofmc: A symbolic model-checker for security protocols. *International Journal of Information Security.*

Birjoveanu, C. V. (2015). Anonymity and fair-exchange in e-commerce protocol for physical products delivery. In *12th International Conference on Security and Cryptography*. SCITEPRESS.

Djuric, Z. and Gasevic, D. (2015). Feips: A secure fair-exchange payment system for internet transactions. *The Computer Journal.*

Draper-Gil, G., Ferrer-Gomila, J. L., Hinarejos, M. F., and Zhou, J. (2013). An asynchronous optimistic protocol for atomic multi-two-party contract signing. *The Computer Journal.*

Ferrer-Gomila, J. L., Onieva, J. A., Payeras, M., and Lopez, J. (2010). Certified electronic mail: properties revisited. *Computers & Security.*

Li, H., Kou, W., and Du, X. (2006). Fair e-commerce protocols without a third party. In *11th IEEE Symposium on Computers and Communications*. IEEE.

Liu, Y. (2009). An optimistic fair protocol for aggregate exchange. In *2nd International Conference on Future Information Technology and Management Engineering*. IEEE.

Liu, Z., Pang, J., and Zhang, C. (2011). Verification of a key chain based ttp transparent cem protocol. *Electronic Notes in Theoretical Computer Science.*

Mukhamedov, A. and Ryan, M. D. (2008). Fair multi-party contract signing using private contract signatures. *Information and Computation.*

Onieva, J. A., Lopez, J., and Zhou, J. (2009). *Secure Multi-Party Non-Repudiation Protocols and Applications*. Springer.

Turuani, M. (2006). The cl-atse protocol analyser. In *17th International Conference on Rewriting Techniques and Applications*. Springer.

Vigano, L. (2006). Automated security protocol analysis with the avispa tool. *Electronic Notes in Theoretical Computer Science.*

Yanping, L. and Liaojun, P. (2009). Multi-party non-repudiation protocol with different message exchanged. In *5th International Conference on Information Assurance and Security*. IEEE.

Zhang, Q., Markantonakis, K., and Mayes, K. (2006). A practical fair exchange e-payment protocol for anonymous purchase and physical delivery. In *4th ACS/IEEE International Conference on Computer Systems and Applications*. IEEE.

Zhou, J., Onieva, J. A., and Lopez, J. (2005). Optimised multi-party certified email protocols. *Information Management & Computer Security Journal.*