
Abstract: There are many multi-party fair exchange protocols with applications in e-commerce transactions for buying digital products, buying physical products, digital signature of contracts, certified e-mail and non-repudiation. However, none of the existing multi-party fair exchange protocols can be applied to provide fair exchange in complex transactions where a customer buys digital products from different merchants while maintaining atomicity. In this paper, we propose a multi-party fair exchange e-commerce protocol for complex transactions in that the customer wants to buy several digital products from different merchants. Our protocol has applications to B2C/B2B scenarios. In addition to strong fairness, our protocol provides effectiveness, timeliness, non-repudiation and confidentiality.

1 INTRODUCTION

In electronic commerce transactions, a common practice is that a customer wishes to buy a pack of products/services composed of several products (digital or physical)/services from different merchants. The atomicity is an important security requirement for such e-commerce transactions in that the customer is interested in buying all products from the pack or no product at all. We will refer to this type of e-commerce transactions as aggregate/atomic transactions. Also, there are cases in which 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. Such e-commerce transactions are suitable when the customer needs more flexibility in expressing his own buying options. We will refer to this type of e-commerce transactions as optional transactions. The complex transactions are the combination in any form of aggregate and optional transactions.

Fair exchange is an essential security requirement in e-commerce protocols. There are a variety of e-commerce protocols that consider only one customer and one merchant, for buying digital or physical products. For example, (Alaraj and Munro, 2010) achieves fair exchange of payment for digital product, and (Djuric and Gasevic, 2015) achieves fair exchange of payment for physical product. 

Up to date, many multi-party fair exchange protocols are known with applications in e-commerce transactions for buying digital goods (Liu, 2009), buying physical goods (Birjoveanu and Birjoveanu, 2018), digital signature of contracts (Draper-Gil et al., 2013), certified e-mail (Zhou et al., 2005) and non-repudiation (Yanping and Liaojun, 2009). But, none of the multi-party fair exchange protocols proposed can be applied to our problem: providing fair exchange in complex transactions where a customer wants to buy digital products from different merchants.

Some multi-party fair exchange protocols (Liu, 2009), (Birjoveanu and Birjoveanu, 2018), (Draper-Gil et al., 2013) solve related problems with our problem, but in other scenarios. The solution from (Liu, 2009) can not be applied to our problem because it considers only an aggregate transaction where a customer wants to buy several digital products from different merchants. Moreover, (Liu, 2009) does not provide strong fairness, but only weak fairness. Weak fairness requires that all parties receive the expected items, or all honest parties will have enough evidence to prove that they behaved correctly in front of an arbiter. Another two important security requirements are not assured in (Liu, 2009): timeliness and confidentiality. The solution from (Draper-Gil et al., 2013) is a multi-two party contract signing protocol, where
a consumer and many providers want to sign a contract pairwise, assuring weak fairness. Because this solution applies to a contract signing scenario different from our scenario, it can not be applied to our problem. (Birjoveanu and Birjoveanu, 2018) proposes an e-commerce protocol for complex transactions in that the customer wants to buy different physical products from different merchants. This solution can not be applied to our problem because it provides fair exchange between digital receipts and digital payments for physical products in complex transactions, while in our scenario the complex transactions involves digital products. The major difference is that in our scenario the digital products must be effectively delivered to customers within the protocol, making the transaction flow more difficult, and this scenario can not be solved by solution from (Birjoveanu and Birjoveanu, 2018).

Our Contribution. In this paper, we propose a multi-party fair exchange e-commerce protocol for complex transactions in that the customer wants to buy different digital products from different merchants. Our protocol has applications to B2C/B2B scenarios. In addition to strong fairness, our protocol provides effectiveness, timeliness, non-repudiation and confidentiality.

The paper is structured as follows: section 2 gives application examples of our protocol, section 3 defines the security requirements, section 4 presents our protocol, section 5 contains the security analysis of the protocol, and the conclusions are in section 6.

2 B2C/B2B APPLICATIONS

Our protocol has use cases in Business to Consumer (B2C) and Business to Business (B2B) applications. For a B2C application, we consider a person Bob that wants to change his job in the future into an IT related field, and to achieve this, he plans to study in his free time. After studying the market requirements, Bob decides to go in one of the following two directions: first is programming and web development, the second is databases. Bob builds his learning plan by browsing through an online course catalog where chargeable downloadable digital courses from different providers, different domains, from which Bob can choose. Bob starts to build his learning plan by choosing courses in the order of his preferences. For programming area, he wants to learn programming techniques and a programming language. For programming technique his first option is the course CP1 Algorithms from Stanford University or if this is not available due to an course update or review process, then his second option is CP2 Data structure and Algorithms from San Diego University. For programming language he prefers CP3 Introduction to Java from San Diego University but if this is not possible, then CP4 Java Programming from University of London. For web development he wants to study CW1 Web development: basic concepts from University of New Mexico or if this is not possible CW2 Web application for everybody from University of Michigan. For databases he wants only CD1 Introduction to structured query language from University of Michigan.

Bob will prepare his learning plan in form of a complex transaction: ((CP1 or CP2) and (CP3 or CP4) and (CW1 or CW2)) or CD1. The complex transaction is an optional transaction in which first preference is an aggregate transaction and the second preference is a single product CD1. The aggregate transaction is composed from three optional transactions.

If instead of a single person, we have a company who wants to offer technical or soft skills trainings to their employees based on their possible development plans, then we have a scenario for B2B application.

3 SECURITY REQUIREMENTS

Next, we will present the following security requirements: effectiveness, fairness, timeliness, non-repudiation and confidentiality which will be achieved in our Multi-Party E-Commerce Protocol for Digital Products (MPPDP). Also, these requirements are stated and analyzed in (Birjoveanu and Birjoveanu, 2018) for multi-party e-commerce protocols for physical products.

Effectiveness requires that if every party involved in MPPDP behaves honestly, does not want to prematurely terminate the protocol, and no communication error occurs, then the customer receives his expected digital products from merchants, and the merchants receive their payments from 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 in MPPDP requires:

- for any optional transaction from the complex transaction, the customer obtains exactly one digital product for the product’s payment, and
- for any aggregate transaction from the complex transaction, the customer obtains the digital products 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 **strong fairness**.

**Timeliness** requires that any party involved in **MPPDP** 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** requires that neither the customer nor any of merchants can deny their involvement in **MPPDP**.

**Confidentiality** in **MPPDP** requires that the content of messages sent between participating parties is accessible only to authorized parties.

### 4 MULTI-PARTY E-COMMERCE PROTOCOL

Our protocol uses as building block the multi-party e-commerce protocol for physical products proposed in (Birjoveanu and Birjoveanu, 2018).

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

Table 1 presents the notations used in the description of **MPPDP**. We use hybrid encryption with the same meaning as in (Birjoveanu and Birjoveanu, 2018). Hybrid encryption \( \{m\}_{\text{Pub}K} \) of the message \( m \) with the public key \( \text{Pub}K \) means \( \{m\}_K, \{K\}_{\text{Pub}K} \): the message \( m \) is encrypted with an AES session symmetric key \( K \), which is in turn encrypted using \( \text{Pub}K \). 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. In **MPPDP**, we consider the same types of communication channels as in (Birjoveanu and Birjoveanu, 2018): resilient communication channels between \( \text{PG} \) and \( C \), respectively between \( \text{PG} \) and each merchant; the other communication channels are unreliable.

#### 4.1 Preparation Phase

Before **MPPDP** execution, a preparation phase is needed. For every digital product \( P \) that a merchant \( M \) wants to sell, he posts on the online catalog a digital certificate of \( P \), \( \text{PCert} \), issued by the certificate authority \( CA \).

\[ \text{PCert} = CI, \text{sigCA}(CI) \]

\( \text{PCert} \) is build by \( CA \) from certification information \( CI \) and \( CA \)’s signature on \( CI \).

\[ CI = \text{ProductDesc}, \text{Pid}, \text{Price}, \{P\}_K, \{K\}_{\text{Pub}KG} \]

\( CI \) contains: the description \( \text{ProductDesc} \) of \( P \), an unique \( P \)’s identifier \( \text{Pid} \), the price \( \text{Price} \) of \( P \), and \( \{P\}_K, \{K\}_{\text{Pub}KG} \) the hybrid encryption of \( P \) with \( \text{Pub}KG \) using the symmetric key \( K \). \( \text{PCert} \) is unique for the digital product \( P \) and certifies that \( P \) has the description \( \text{ProductDesc} \), the identifier \( \text{Pid} \), the price \( \text{Price} \) and a hybrid encrypted version. Also, \( M \) receives the key \( K \) from \( CA \) on a private channel.

The customer is browsing through the online catalog where the products from merchants are posted. After the customer decides the digital products pack he wants to buy and the options for each product from the pack, he clicks a "submit" button on the online catalog and the download of the payment Web segment and the digital product certificates is started. The payment Web segment has the same role as in the protocol from (Birjoveanu and Birjoveanu, 2018). 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 \( PG \) that runs on the customer’s computer, and has the digital certificates for the public keys of each merchant, \( PG \) and \( CA \). The payment Web segment checks each digital product certificate using \( CA \)’s public key. 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, the payment Web segment generates a RSA session public/private key pair for customer. We consider that each merchant has the digital certificates for the public keys of \( PG \) and \( CA \). Also, \( PG \) has the digital certificates for the public keys of each merchant and \( CA \).

For an aggregate transaction, we define the **aggregation** operator, denoted by \( \land \), as follows: \( \text{Pid}_1 \land \ldots \land \text{Pid}_k \) meaning that \( C \) wishes to buy exactly \( k \) products with product’s identifiers \( \text{Pid}_1, \ldots, \text{Pid}_k \). For an optional transaction, we define the **option** operator, denoted by \( \lor \), as follows: \( \text{Pid}_1 \lor \ldots \lor \text{Pid}_k \) meaning that \( C \) wishes to buy a product that is exactly one of the products with product’s identifiers \( \text{Pid}_1, \ldots, \text{Pid}_k \), where the apparition order of the product’s identifiers is the priority given by \( C \).

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 \( \land \) and \( \lor \) operators. To represent the tree, we use the **left-child, rightsibling representation** in that each internal node corresponds to one of the above operators or to an identi-
Table 1: Notations used in the protocol description.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>C, PG, CA, M, PubKA, {m} \text{PubKA}</td>
<td>Identity of Customer, Payment Gateway, Certificate Authority, Merchant i, where 1 \leq i \leq n</td>
</tr>
<tr>
<td>h(m)</td>
<td>RSA public key of the party A, hybrid encryption of the message m with PubKA</td>
</tr>
<tr>
<td>SigA(m)</td>
<td>The digest of the message m obtained by applying of a hash function h (SHA-2)</td>
</tr>
<tr>
<td>A \rightarrow B:m</td>
<td>A sends the message m to B</td>
</tr>
</tbody>
</table>

Table 1: Notations used in the protocol description.

4.2 Payment Sub-protocol

Payment sub-protocol uses the Subtransaction Payment SPayment sub-protocol. First, we will describe SPayment sub-protocol in that the customer C agrees with a certain merchant M to buy a certain digital product, C sends to M the payment for product and M sends to C the digital receipt for payment.

4.2.1 SPayment Sub-protocol

SPayment consists of four sub-protocols: the setup sub-protocol, the pay sub-protocol and two resolution sub-protocols. Next, we will describe SPayment’s sub-protocols messages that are graphically represented in Figure 2.

Setup Sub-protocol. In the first message, the payment Web segment sends to M the customer’s session public key PubKC generated in the preparation phase.

Message 1.1: C \rightarrow M: \{PubKC\}_\text{PubKM}

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 for any reason, M does not want to continue the setup sub-protocol, then he will send instead to C a signed response Resp with the value ABORT. On reception, C decrypts it and authenticates M by checking M’s signature.

Message 1.2: M \rightarrow C: \{Sid, SigM(Sid)\}_\text{PubKC} \text{, or } M \rightarrow C: \{Resp, Sid, SigM(Resp, Sid)\}_\text{PubKC}, where Resp = ABORT

Pay Sub-protocol. If C does not receive an ABORT response and he authenticates M, then the payment Web segment sends to M in the message 1.3 a payment message PM and a purchase order message PO, both encrypted with PubKM.

Message 1.3: C \rightarrow M: \{PM, PO\}_\text{PubKM}

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

The payment Web segment builds PM by encrypting with PG’s public key the payment information PI and the customer’s signature on PI.

PI = CardN, CCode, Sid, Price, PubKC, NC, M

PI contains the data provided by user: card number CardN and a challenge code CCode issued by bank to user via SMS which has a minimum length of four characters. PI contains in addition Sid, Price, PubKC, a fresh nonce NC generated by C, and the merchant’s identity M.

PO = OI, SigC(OI)

The payment Web segment builds PO from the order information OI provided by C and the signature of C on OI. OI contains OrderDesc – the order description for product, Pid, Sid and Price. 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 1.4 to PG. Otherwise, M sends to C an ABORT response.

Message 1.4: M \rightarrow PG: \{PM, SigM(Sid, PubKC, Price)\}_\text{PubKPG}

In the message 1.4, M sends to PG the payment message PM and his signature on Sid, PubKC and
Price. 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 is successful, then it has the confirmation that both C and M agreed on Sid, PubKC and Price. 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 an ABORT response for aborting the sub-transaction. If all checks are successful, PG sends PM 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 a YES response (Resp = YES) to PG that forwards it to M in the message 1.5. Otherwise, if checking C’s account balance fails, then also the transfer fails and the bank provides an ABORT response (Resp = ABORT) to PG that forwards it to M in the message 1.5. As an evidence of the sub-transaction details, PG stores the messages 1.4 and 1.5 in its databases.

Figure 2: SPayment message flow.

Message 1.5: PG → M:
{Resp,Sid,SigPG(Resp,Sid,Price,NC)}_{PubKPG}
M decrypts the message 1.5, checks PG’s signature, and sends to C the message 1.6.

Message 1.6: M → C:
{Resp,Sid,SigPG(Resp,Sid,Price,NC)}_{PubKC}
C decrypts message 1.6 and checks PG’s signature. If checking is successful, then C has the guarantee of the response’s authenticity and it corresponds to the current sub-transaction. The presence in response of Sid, Price and NC proves the response’s freshness. A response with Resp=YES means that the payment subtransaction successfully finished and the content of message 1.6 (Resp,Sid,SigPG(Resp,Sid,Price,NC)) is a digital receipt for the payment of product. A response with Resp=ABORT means that the content of message 1.6 is a proof of the subtransaction’s abort.

Resolution 1 Sub-protocol. If C initiates SPayment, but he does not receive any message from M or receives an invalid message 1.2 from M, then the current subtransaction’s state is undefined. If we consider only the current subtransaction, then SPayment does not give any benefit to any party. However, this case must be solved if we reason that the current subtransaction belongs to an aggregate transaction that contains subtransactions in which the payment already successfully finished, as we will see in Payment sub-protocol. In this case, a timeout t1 is defined, in which C waits the message 1.2 from M. If t1 expires and C does not receive the message 1.2 from M or receives an invalid message, then C initiates the Resolution 1 sub-protocol with PG.

Message 1.7: C → PG:{[PubKC]}_{PubKPG}
C sends to PG in the message 1.7 a response request for the current subtransaction. PG decrypts it, checks that no response has been generated for PubKC, generates a subtransaction identifier Sid and sends to C in message 1.8 a signed ABORT response. Also, PG stores the response in its databases.

Message 1.8: PG → C:
{Resp,Sid,SigPG(Resp,Sid)}_{PubKPG}, where Resp = ABORT. C decrypts and authenticates it.

Resolution 2 Sub-protocol. If C sends the payment in message 1.3, but he does not receive message 1.6, 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 t2 is defined, in which C waits message 1.6 from M. If t2 expires and C does not receive message 1.6 from M or receives an invalid message, then C initiates the Resolution 2 sub-protocol with PG.

Message 1.9: C → PG:{Sid,Price,NC,PubKC, SigC(Sid,Price,NC,PubKC)}_{PubKPG}
PG decrypts it and checks if a response has been generated for Sid, Price, NC and PubKC. If PG finds in its database a response, and checking the signature of C using PubKC is successful, then sends to C the response in message 1.10. Otherwise, if PG does not find a response, then sends to C an ABORT response in message 1.10 and stores it.

Message 1.10: PG → C:
{Resp,Sid,SigPG(Resp,Sid,Price,NC)}_{PubKPG}

4.2.2 Payment Sub-protocol Description

In Payment sub-protocol, a subtransaction s, denoted by SPayment(C,M,PID), is an instance of SPayment in which C buys the digital product with PID identifier from M. We define S(t) the state of the
Table 2: Payment sub-protocol.

\[\begin{align*}
\text{Payment}(t) & \quad 1. \text{if} \ (t \to \text{left} \neq \text{NULL}) \ \text{child}[0] = \text{Payment}(t \to \text{left}); \\
& \quad 2. \text{if} \ ((t \to \text{info} = \text{in} \text{v} \text{a} \text{g} \text{e}) \text{Resp} = \text{YES}, \text{for all } S(t) \text{from child}[0]) \text{ or} \\
& \quad 3. \ (t \to \text{info} = \text{a} \text{nd} \text{ and } S(t) \text{.Resp} = \text{ABORT}, \text{for all } S(t) \text{from child}[0]) \ Ns(t) = \text{child}[0]; \text{return } Ns(t); \\
& \quad 4. \ j = 1; \ k = t \to \text{left} \to \text{right}; \\
& \quad 5. \text{while} \ (k \neq \text{NULL}) \\
& \quad \quad 6. \ \text{child}[j] = \text{Payment}(k); \\
& \quad \quad 7. \text{if} \ (t \to \text{info} = \text{in} \text{v} \text{a} \text{g} \text{e} \text{and} \ S(t) \text{.Resp} = \text{YES}, \text{for all } S(t) \text{from child}[j]) \ Ns(t) = \text{child}[j]; \text{return } Ns(t); \\
& \quad \quad 8. \text{if} \ (t \to \text{info} = \text{a} \text{nd} \ S(t) \text{.Resp} = \text{ABORT}, \text{for all } S(t) \text{from child}[j]) \\
& \quad \quad \quad \text{for} \ (c = 0; c \leq j; c = c + 1) \ Ns(t) = Ns(t) \text{child}[c]; \text{end for} \\
& \quad \quad 9. \ \text{AggregateAbort}(Ns(t)); \text{return } Ns(t); \\
& \quad \quad 10. \ k = k \to \text{right}; j = j + 1; \text{end while} \\
& \quad 12. \text{if} \ (t \to \text{info} = \text{Pid}_0) \ Ns(t) = SPayment(C, M, \text{Pid}_0); \text{return } Ns(t); \\
& \quad \text{else if} \ (t \to \text{info} = \text{in} \text{v} \text{a} \text{g} \text{e}) \ k = t \to \text{left}; \\
& \quad \quad \text{while} \ (k \to \text{right} \neq \text{NULL}) \ k = k \to \text{right}; \text{end while} \\
& \quad \quad 14. \ Ns(t) = Ns(k); \text{return } Ns(t); \\
& \quad 16. \text{else for} \ (c = 0; c \leq j - 1; c = c + 1) \ Ns(t) = Ns(t) \text{child}[c]; \text{end for} \\
& \quad \text{return } Ns(t); \text{end if} \text{ end if}
\end{align*}\]

Subtransaction \(s\) as being the content of one of the messages 1.2, 1.6, 1.8 or 1.10 that \(C\) will receive in \(s\).

For \(S(t)\) a state of the subtransaction \(s\), we denote by \(S(t)\).Resp the response (Resp) in \(S(t)\).

We define \(Ns(p)\) - the state of the node \(p\) as a sequence of subtransaction states \(S(t_1)\ldots S(t_m)\) corresponding to the product defined by \(p\) similarly as in (Birjoveanu and Birjoveanu, 2018).

The Payement sub-protocol, 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 \(p\).

Although some subtransactions from the aggregate transaction successfully completed \(SPayment\), there may be cases in which an aborted subtransaction/sequence of subtransactions leads to aborting the entire aggregate transaction. Therefore, an unfair case occurs for \(C\): the aggregate transaction is not successful, but \(C\) has paid for certain products. For example, for a node \(p\) that corresponds to \& operator, the Payment sub-protocol computes \(Ns(p) = S(t_1)\ldots S(t_k)S(t_{k+1})\ldots S(t_m)\), where \(S(t_i)\).Resp = YES for any \(1 \leq i \leq k\) and \(S(t_j)\).Resp = ABORT for any \(k + 1 \leq j \leq m\). The complex transaction corresponding to \(p\) is unsuccessful because \(s_{k+1}, s_m\) are aborted, but \(C\) paid for the products involved in \(s_1, s_k\). So, fairness will be obtained by applying \(AggregateAbort(Ns(p))\) sub-protocol from (Birjoveanu and Birjoveanu, 2018) in that entire aggregate transaction will be aborted.

After Payment sub-protocol is completed, \(Ns(t - root)\) is the sequence of the subtransaction states for which either all subtransactions successfully completed \(SPayment\) or all aborted.

### 4.3 Delivery Sub-protocol

If after Payment all subtransactions from \(Ns(t - root)\) successfully completed \(SPayment\), then for each of these subtransactions the Delivery sub-protocol is executed. In Delivery sub-protocol, the decryption key of the digital product involved in the subtransaction is sent from the corresponding merchant \(M\) to \(C\). Below, we describe the Delivery sub-protocol for an arbitrary subtransaction \(s\) as shown in Figure 3.

![Delivery message flow](image)

The payment web segment initiates in the message 2.1 the Delivery sub-protocol by sending to \(M\) the state of the subtransaction \(s\) that contains the digital receipt for the payment of product ordered in \(s\).
Message 2.1: $C \rightarrow M: \{St(s)\}_{PubKM}$, where $St(s) = (Resp, Sid, SigPG(Resp, Sid, Price, NC))$ with $Resp = YES$. $M$ decrypts it, checks $PG$'s signature and checks if it has already stored this digital receipt. If all checks are successful, then $M$ sends to $C$ the decryption key $K$ of the product ordered in $s$.

Message 2.2: $M \rightarrow C: \{Sid, SigM(Sid, K)\}_{PubKC}$ Upon receiving, $C$ decrypts it, recovers the key $K$, checks $M$'s signature, and uses $K$ to obtain the digital product $P$ by decrypting $\{P\}_K$ from $PCert$.

**Resolution 3 Sub-protocol.** If $C$ sends in message 2.1 the digital receipt for the payment of product ordered in $s$, but he does not receive message 2.2, or receives an invalid decryption key, then an unfair case appears: $C$ sends the payment for product and it was processed, but $C$ did not receive the corresponding decryption key for product’s decryption. In this case, a timeout $t_3$ is defined, in which $C$ waits message 2.2 from $M$. If $t_3$ expires and $C$ does not receive message 2.2 from $M$ or receives an invalid decryption key, then $C$ initiates the Resolution 3 sub-protocol with $PG$. The message 2.3 is build from $PO$, the state of $s$, and the product’s certificate $PCert$, all of them encrypted with $PubKPG$.

Message 2.3: $C \rightarrow PG:\{PO, St(s), PCert\}_{PubKPG}$, with $St(s) = (Resp, Sid, SigPG(Resp, Sid, Price, NC))$ and $Resp = YES$. $PG$ decrypts it and recovers $PO$, $St(s)$ and $PCert$. $PG$ checks the signature of $C$ from $PO$, his signature from $St(s)$, and if $Sid$ and $Price$ from $PO$, respectively $St(s)$ are the same. If these checks are satisfied, then $PG$ is ensured that the digital receipt from $St(s)$ corresponds to the product ordered by $C$ in $PO$. $PG$ checks $CA$’s signature from $PCert$, and if the product identifier $Pid$ from $PCert$ is identical with the one from $PO$. Then, $PG$ has the guarantee that the digital receipt from $St(s)$ corresponds to the product (ordered in $PO$) that has the digital certificate $PCert$. $PG$ decrypts $\{K\}_{PubKPG}$ (from $PCert$) with his private key and recovers the key $K$ that will send to $C$ in the message 2.4.

Message 2.4: $PG \rightarrow C: \{Sid, K, SigPG(Sid, K)\}_{PubKCC}$ $C$ decrypts it, recovers $K$ and uses it to obtain $P$.

5 SECURITY ANALYSIS

In what follows, we will analyze the security requirements stated in section 3 for our MPPDP.

**Effectiveness.** If every party involved in MPPDP behaves according to the protocol’s steps, does not want to prematurely terminate the protocol and there are no network communication delays/erros, then MPPDP assures that $C$ receives the digital products from merchants, and each merchant receives his payment from $C$ without TTP involvement.

**Fairness.** MPPDP consists of the Payment sub-protocol followed by the Delivery sub-protocol. Fair exchange of payments for digital products in complex transactions in MPPDP is obtained from fair exchange of payments for digital receipts in complex transactions in Payment and fair exchange of digital receipts for decryption keys of digital products in Delivery.

Payment uses $SPayment$. Fairness in $SPayment$ is obtained by a similarly analysis with the one from (Birjoveanu and Birjoveanu, 2018) for fairness of STP protocol.

To obtain fairness in Payment, two requirements must be ensured in addition to fairness in $SPayment$. 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 product, or none of them obtains nothing. This requirement is satisfied in Payment from the way in which the node state corresponding to $\lor$ operator is computed (Table 2). 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 payment for product, or none of them obtains nothing. In Payment, the node state for a node corresponding to the $\land$ operator is computed as follows: if all subtransactions from all node states of all children of the node $\land$ have successfully finished $SPayment$, then the node state of $\land$ is the sequence of node states of its children. Otherwise, an unfair scenario can occur for $C$ if the entire aggregate transaction is not successful, but $C$ has paid for certain products and he received the digital receipts for these. To obtain fairness in this case, the AggregateAbort sub-protocol is applied to abort any subtransaction that successfully finished $SPayment$ and that belongs to the unsuccessful aggregate transaction. So, fair exchange of payments for digital receipts in Payment is provided.

If after Payment all subtransactions from $Ns(t - root)$ are aborted, then Delivery is not applied and fairness in MPPDP is obtained from fairness of Payment: $C$ does not obtain any digital receipt for payment and no merchant receives the payment. If after Payment all subtransactions from $Ns(t)$ successfully completed $SPayment$, then for each of these subtransactions the Delivery sub-protocol is executed. In Delivery, the fairness for $C$ is not insured only in one case: in some subtransaction from $Ns(t)$, $C$ sends the digital receipt for payment to $M$ in message 2.1, but he does not receive the corresponding decryption key in message 2.2. In this case, $C$ waits for the message.
2.2 until the timeout $t_3$ expires and then $C$ initiates the Resolution 3 sub-protocol with $PG$ to receive the corresponding decryption key. As we can see, the unfair case is solved and fairness in Delivery is provided: $C$ receives the corresponding decryption keys for digital products and each merchant receives the payment for corresponding product. So, fairness in MPPDP and atomicity of complex transactions are preserved.

**Timeliness.** In MPPDP, we have introduced three timeouts: $t_1$ and $t_2$ in $SPayment$, and $t_3$ in Delivery. If for $1 \leq i \leq 3$, a timeout $t_i$ expires, then $C$ initiates the Resolution $i$ sub-protocol with $PG$. Moreover, if in MPPDP, the AggregateAbort sub-protocol is executed, then the communication channels used are resilient: between $C$ and $PG$, respectively between each merchant and $PG$. So, the execution of MPPDP will be finished at a certain finite point of time.

If after MPPDP finish point, $C$ has the digital products, then each merchant obtains the corresponding payment for his product. Also, if each merchant involved obtains the payment for his product, then, after the protocol finish point, $C$ gets the corresponding digital products because he receives the corresponding decryption keys of the digital products from Delivery or Resolution 3. On the other side, if each merchant did not get the payment, then $C$ does not obtain the digital products. Conversely, if $C$ has not obtained the digital products, then also no merchant gets the payment. Thus, after MPPDP finish point, the level of fairness achieved cannot be degraded.

**Non-repudiation.** $C$ cannot deny its participation in MPPDP because in any subtransaction from MPPDP, $PG$ stores in its database $C$’s signature on $Pl$ and thus, $C$ cannot deny its signature on $Pl$. Also, no merchant can deny its participation in MPPDP because in any subtransaction from MPPDP, $PG$ has the evidence of the merchant’s signature (from message 1.4) on $Sid$, $PubK$ and $Price$.

**Confidentiality.** In MPPDP, only the authorized receiver of any message can read the message’s content because every message transmitted is hybrid encrypted with the receiver’s public key. For example, the confidentiality of $CardN$ between $C$ and $PG$ is obtained because $C$ sends $CardN$ hybrid encrypted with $PG$’s public key.

### 6 CONCLUSIONS

In this paper, we proposed an optimistic multi-party fair exchange e-commerce protocol for complex transactions applicable to B2C and B2B scenarios. Table 3 emphasizes the security requirements ensured for our protocol and compares these with the security requirements obtained by the most related solutions to our proposal. The highlights of our protocol are strong fair exchange, atomicity, usage of $PG$ as offline TTP, timeliness, non-repudiation and confidentiality.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>MPPDP</th>
<th>Liu</th>
<th>Birjoveanu Draper-Gil</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Atomicity</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Fairness</td>
<td>Strong</td>
<td>Weak</td>
<td>Strong</td>
</tr>
<tr>
<td>Timeliness</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Non-repudiation</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

1 = Payment for digital products in complex transactions  
2 = Payment for digital products in aggregate transactions  
3 = Payment for physical products in complex transactions  
4 = Contract signing  
Y=YES, N=NO

Future work will include extending the proposed protocol by taken into consideration of active intermediary agents between customer and merchants.

### REFERENCES


