Secure E-Commerce Protocol with Complex Trading Capabilities of Intermediaries

Cătălin V. Bîrjoveanu[1] and Mirela Bîrjoveanu[2]
[1]Department of Computer Science, "Al.I.Cuza" University of Iaşi, Iaşi, Romania
[2]Vitesco Technologies, Iaşi, Romania

Keywords: Security Protocols, Formal Verification, Multi-Party E-Commerce, Complex Transactions, Strong Fairness.

Abstract: Up to now, there are many multi-party fair exchange protocols with applications in buying physical/digital goods, digital signature of contracts and certified e-mail, but there is no e-commerce protocol that allows multiple intermediaries to perform aggregate, chained or optional transactions. In this paper, we propose the first multi-party e-commerce complex transaction protocol that allows the customer to acquire some physical products through many intermediaries and providers. Considering complex transactions rise new challenges for assuring strong fairness, that are not appearing in two-party transactions. The objective of our proposal is to ensure strong fairness, effectiveness, timeliness, non-repudiation and confidentiality in a multi-party scenario. The formal verification of our proposal using Cl-AtSe model checker proves that all security requirements mentioned above are satisfied.

1 INTRODUCTION

E-commerce is a widely used term that is an umbrella which includes many types of commercial activities. The security threats to e-commerce and the growing losses they cause lead to the need of secure protocols to protect from them.

In this paper, we propose a secure e-commerce protocol with applications in supply chain, considering intermediaries with complex trading capabilities. In our proposal, the customer plans to acquire some physical products through many intermediaries and providers. An intermediary is a party which to fulfill a buying request received from customer or another intermediary, has the capability to perform a chained, aggregate or optional transaction with other intermediaries/providers. An intermediary is engaged in a *chained transaction* if he receives a request for a product, and to fulfill it, in his turn requests the product from a provider or another intermediary. When an intermediary wants to acquire an entire pack of products, he is engaged in an *aggregate transaction*. There are situations in which the buying request can not be fulfilled in terms of delivery time, quantity, etc. In these cases, is useful for the party that requested the product to be able to express more options for products with similar features, engaging in this way in an *optional transaction* (only one product from the expressed options is acquired). The complex trading capabilities of the intermediaries lead to a new business model in which a complex transaction is a composition of chained, aggregate and optional transactions.

Aggregate and optional transactions are very important for robust procurement strategy in supply chain. Optional transactions are important because they minimize the risk of relying on a single intermediary or provider, which can cause a disruption following a natural disaster or geopolitical events. Companies that source from different providers create resilience in their procurement process. On the other hand, aggregate transactions allow the companies not to remain with temporary unnecessary stocks (parts that they cannot use due to other parts that are missing), so reducing the inventory carrying costs.

Commonly, an e-commerce protocol involving multiple parties ensures strong fairness if after the protocol execution all parties will receive their expected items or none do.

In the literature, we find multi-party fair exchange e-commerce protocols with applications in buying physical products (AlTawy et al., 2017), digital products (Liu, 2009) and digital signature of contracts (Ferrer-Gomila and Hinarejos, 2021). These proposals do not consider any intermediaries.

Instead, there are few solutions that consider intermediaries in different multi-party scenarios: consid-

Bîrjoveanu, C. and Bîrjoveanu, M.
Secure E-Commerce Protocol with Complex Trading Capabilities of Intermediaries.
DOI: 10.5220/0012127100003555
In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT 2023)*, pages 495-502
ISBN: 978-989-758-666-8; ISSN: 2184-7711

ering only one intermediary (Carbonell et al., 2009; Onieva et al., 2004), and considering more intermediaries (Draper-Gil et al., 2013), (Bîrjoveanu and Bîrjoveanu, 2020). In (Carbonell et al., 2009), a customer can buy products from several providers through an intermediary agent integrated in 3-D Secure Protocol. In this solution, the intermediary can not perform aggregate transactions, because in the same transaction (in which the customer buys many products), the authorization process is realized separately for each individual buying request. Also, the optional transactions are not considered. In (Onieva et al., 2004) is proposed a protocol for exchange of non-repudiation evidences, but without dealing with aggregate or optional transactions. In (Bîrjoveanu and Bîrjoveanu, 2020), a multi-party protocol that allows a customer to fairly acquire physical products from different providers through many intermediaries is proposed. However, in this solution the intermediaries are restricted to perform only chained transactions, without the capability to perform aggregate or optional transactions. In (Draper-Gil et al., 2013), the intermediaries are used to enable contract signing between multiple parties ensuring only weak fairness, in a different scenario then the one approached in our paper. Although this solution considers aggregate transaction, it does not consider optional transaction.

The complex transactions rise new challenges for assuring strong fairness, that are not appearing in two-party transactions. An issue appears when the customer successfully buys only a part of the components of an aggregate product. In this case, strong fairness is ensured for all transactions composing the aggregate transaction, but strong fairness for the entire aggregate transaction is not guaranteed. Another issue appears when the customer successfully buys more than one product in an optional transaction. Strong fairness for the optional transaction is not assured, although strong fairness is guaranteed for all transactions composing the optional transaction. In a chained transaction an intermediary can buy a product on demand, but afterward he cannot provide it to the customer or intermediary who requested it due to various reasons (insufficient funds, malicious behavior, etc). In this case, strong fairness for all transactions belonging to the chained transaction is satisfied, even if strong fairness for the chained transaction is not satisfied.

Contributions The related work presented above emphasizes the absence of multi-party e-commerce protocols considering intermediaries with complex trading capabilities. In this paper, we propose the first multi-party e-commerce protocol that allows intermediaries to perform aggregate, chained or optional transactions depending on the received request. These complex trading capabilities of the intermediaries are more appropriate to the real world applications, such as supply chain scenarios. Ensuring strong fairness is a challenging issue in environments where multiple parties are involved, as we showed above. We design the complex transaction protocol using a modular approach by designing a sub-protocol for each type of transaction from the complex transaction.

This design approach allows us to demonstrate the correctness of our complex transaction protocol proposal by demonstrating the correctness of each sub-protocol using Cl-AtSe model checker. The formal specification and verification is challenging because we must take into consideration the communications performed between sub-protocols when they are integrated in the complex transaction protocol. The verification results prove that our complex transaction protocol satisfies all aimed security requirements: strong fairness, effectiveness, timeliness, non-repudiation and confidentiality.

The paper is structured as follows: Sect. 2 presents an use case of our protocol, Sect. 3 defines security requirements, Sect. 4 describes our protocol. Sect. 5 sketches the security analysis of our protocol and Sect. 6 contains the conclusion.

2 APPLICATIONS

A toy company KidsBots (KB) is preparing the production for a new toy robot. To start the production, it needs the following components: motherboard, 32 led panel, 12V DC motors, and body plastic parts (bp). KB identify the necessary components from the online catalog of MasterBroker (MB) intermediary. For the motherboard, KB identified two options having similar features: Motherboard ITX (mb_1), and if it is not available, Motherboard 4 (mb_2). For 32 led panel, KB needs the model 32LP (lp). The toy company has three options for 12V DC motors: FastDC (m_1) or HpDC (m_2) or MetalDC (m_3). So, KB prepares its order to acquire from MB the needed parts, as a complex transaction: $((mb_1 \vee mb_2) \wedge lp) \wedge (m_1 \vee m_2 \vee m_3) \wedge bp$. $\wedge$ represents an aggregate transaction, and $\vee$ an optional one. The complex transaction is an aggregate transaction in which the first component is an aggregate transaction, the second is an optional transaction and third is an individual product. The complex transaction is illustrated in Fig. 1.

After MB receives the request from KB, it splits it in three components: places the order for $(mb_1 \vee mb_2) \wedge lp$ to PCBShop, the order for $(m_1 \vee m_2 \vee m_3)$ to SensorActuatorComp (SensA), and the order for bp to PlasticRoboP. When receiving the order for $(mb_1 \vee$

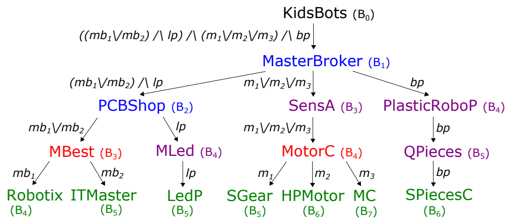

Figure 1: Supply chain applications.

$mb_2) \wedge lp$, PCBShop places an order for $(mb_1 \vee mb_2)$ to MBest that initiates in its turn an order to the provider Robotix to acquire $mb1$, but if this is not available, afterward initiates an order to the provider ITMaster for mb_2. To acquire lp, PCBShop initiates a transaction with MLed that in its turn initiates a chained transaction with the provider LedP.

In Fig. 1, the intermediaries colored in blue perform aggregate transactions, the one colored in red perform optional transactions and the one colored in magenta perform chained transactions. The green color is associated to the providers.

3 SECURITY REQUIREMENTS

In our *Protocol with Complex Trading Capabilities of Intermediaries (PCTCI)* we want to ensure the following security requirements: *strong fairness*, *effectiveness*, *timeliness*, *non-repudiation*, and *confidentiality*. We will introduce these security requirements.

Strong Fairness. We will gradually define it, from two-party to multi-party scenarios. *A two-party e-commerce sub-protocol ensures strong fairness* if after its execution, the party that initiates the sub-protocol receives a successful payment evidence from the corresponding receiver and the receiver receives the payment for the product from the corresponding initiator (successful subtransaction), or none do (aborted subtransaction). *A chained e-commerce sub-protocol ensures strong fairness* if after its execution, either all subtransactions from chain are successfully completed, or all are aborted. *An aggregate e-commerce sub-protocol ensures strong fairness* if after its execution, either all component subtransactions of aggregate transaction are successfully completed, or all are aborted. *An optional e-commerce sub-protocol ensures strong fairness* if after its execution, either only one component subtransaction of optional transaction is successful, or all are aborted.

PCTCI ensures strong fairness if after its execution, any instance of two-party, aggregate, optional and chained sub-protocol ensures strong fairness and all of them provides the same fairness level (either all aggregate, optional and chained transactions are successful, or all are aborted).

Effectiveness requires that if every party involved in *PCTCI* behaves honestly and no communication error occurs, then after protocol execution strong fairness is guaranteed, with success of complex transaction, without Trusted Third Party (*TTP*) mediation.

PCTCI guarantees timeliness if any party involved in *PCTCI* can be sure that protocol execution will be finished at a certain finite point of time without losing strong fairness.

Non-repudiation in PCTCI prevents any party to falsely deny its involvement in *PCTCI*.

PCTCI ensures confidentiality if the message's content communicated between the participating parties is not accessible to unauthorized parties.

4 THE PROTOCOL

The following participants are involved in *PCTCI*: the customer, the intermediaries, the providers, the payment gateway and the bank. The payment gateway is used as interface between intermediaries/providers and the bank to facilitate making payments. Table 1 presents the notations used in the description of *PCTCI*, and Table 2 provides the detailed structure of the protocol's messages. The communication channels between *PG* and any other party are considered resilient, through which messages can be delayed but not lost (Onieva et al., 2009). The communication channels between any other parties are unreliable, meaning that the messages can be lost. We consider that each participant has the digital certificates for the public keys of each participant he communicates with. C searches the needed products in the intermediary's online catalog, order them as a complex transaction and send it to the intermediary when C clicks the "submit" button. The intermediary acquires the products requested by C from many others intermediaries/providers, any intermediary acquiring in his turn the products by performing any type of transaction: aggregate, optional or chained.

Next, we give an overview about the protocol's functionality. C initiates *PCTCI* sending $PO_{0,1}$ to B_1. To accomplish C's request, B_1 initiates a transaction $t_{1,k}$, sending $PO_{1,m}$ to the intermediary/provider B_m, where $2 \leq m \leq k$. The transaction $t_{1,k}$ can be $at_{1,k}$, $ot_{1,k}$ or $s_{1,k}$. Each intermediary B_m can initiate an aggregate, optional or chained transaction. This process continue until the providers are reached. The complex transaction is completed backwards from the providers through intermediaries until C, as follows. Each provider sends to the intermediary from which received the purchase request, the corresponding pay-

Table 1: Notations used in the protocol description.

Notation	Interpretation
P, C, PG, B_i	Set of providers, Identity of Customer, Payment Gateway, Intermediary (Broker) i; $B_0 = C$;
$s_{i,j}$	One-to-one subtransaction initiated by B_i with B_j
$at_{i,j}$ / $ot_{i,j}$	One-to-many aggregate / optional transaction initiated by B_i with $B_{i+1},...,B_j$
$t_{i,j}$	Transaction $s_{i,j}$, $at_{i,j}$ or $ot_{i,j}$
$PO_{i,j}$ / $PR_{i,j}$	Purchase Order of B_i to B_j / Payment Request of B_j to get payment from B_i
$PE_{i,j}$ / $APE_{i,j}$	Payment Evidence of B_i and B_j in $s_{i,j}$ / Aborted Payment Evidence of B_i and B_j in $s_{i,j}$
$\overline{E}_{j,k}$	Payment Evidence in $t_{j,k}$ that B_j sends to B_i in $s_{i,j}$
$X \rightarrow Z : m$	A party X sends the message m to a party Z
$X \Rightarrow B_i : m_i$	A party X simultaneously sends the messages m_i to the set of parties $\{B_i/1 \leq i \leq n\}$
$\{m\}_{PkX}$	$\{m\}_K, \{K\}_{PkX}$ - hybrid encryption of m with PkX using the AES session symmetric key K
$SigX(m)$; Y/A	RSA digital signature of X on $h(m)$, where h is a hash function; $YES/ABORT$

Table 2: Protocol messages details.

$$PO_{i,j} = \{PM_i,\ OI_i\}_{PkB_j} \quad PM_i = \{PI_i,\ SigB_i(PI_i\}_{PkPG} \quad PI_i = B_i,\ Cn_i,\ Otp_i,\ Id_{i,j},\ Am_{i,j},\ B_j$$

$$OI_i = B_i,\ B_j,\ Pid_{i,j},\ Id_{i,j},\ Am_{i,j},\ SigB_i(B_i,\ B_j,\ Pid_{i,j},\ Id_{i,j},\ Am_{i,j})$$

$$PR_{i,j} = \{PM_i,\ \overline{Pid}_{i,j},\ SigB_j(\overline{Pid}_{i,j},\ Id_{i,j},\ B_i,\ B_j,\ \overline{Am}_{i,j})\}_{PkPG}$$

$$\overline{Pid}_{i,j} = \begin{cases} Pid_{i,j}, & \text{if } B_j \in P \quad (1.1) \\ PE_{j,k}.Pid, & \text{if } t_{j,k} = s_{j,k} \text{ and } PE_{j,k}.Resp = Y \quad (1.2) \\ PE_{j,j+1}.Pid,...,PE_{j,k}.Pid, & \text{if } t_{j,k} = at_{j,k} \text{ and } PE_{j,m}.Resp = Y \text{, for all } j+1 \leq m \leq k \quad (1.3) \\ PE_{j,m}.Pid, & \text{if } t_{j,k} = ot_{j,k} \text{ and } PE_{j,m} \text{ is the first evidence with } PE_{j,m}.Resp = Y,\ j+1 \leq m \leq k \quad (1.4) \end{cases}$$

$$PE_{i,j} = Resp, B_i, B_j, Id_{i,j}, \overline{Pid}_{i,j}, SigPG(Resp, B_i, B_j, Id_{i,j}, \overline{Pid}_{i,j}, \overline{Am}_{i,j}), E_{i,j}$$

$$E_{i,j} = Resp, Id_{i,j}, \overline{Pid}_{i,j}, SigPG(Resp, Id_{i,j}, \overline{Pid}_{i,j})$$

$PE_{i,j}.Pid/PE_{i,j}.Am$ - The product identifier *Pid*/ amount *Am* from $PE_{i,j}$

$PE_{i,j}.Resp/E_{i,j}.Resp$ - The response *Resp* in $PE_{i,j}$/$E_{i,j}$

$\overline{Am}_{i,j}$ is defined in a similar manner as $\overline{Pid}_{i,j}$ by replacing *Pid* with *Am* in the definition of $\overline{Pid}_{i,j}$ above

$$\overline{E}_{j,k} = \begin{cases} Cert(B_j), & \text{if } B_j \in P \quad (2.1) \\ E_{j,k}, & \text{if } t_{j,k} = s_{j,k} \text{ and } E_{j,k}.Resp = Y \quad (2.2) \\ E_{j,j+1},...,E_{j,k}, & \text{if } t_{j,k} = at_{j,k} \text{ and } E_{j,m}.Resp = Y \text{, for all } j+1 \leq m \leq k \quad (2.3) \\ E_{j,m}, & \text{if } t_{j,k} = ot_{j,k} \text{ and } E_{j,m} \text{ is the first evidence with } E_{j,m}.Resp = Y,\ j+1 \leq m \leq k \quad (2.4) \end{cases}$$

$\overline{E}_{j,k}.Resp$ - The sequence of responses from all evidences from $\overline{E}_{j,k}$

$\overline{E}_{j,k}.Resp = Y$ - The responses from all evidences from $\overline{E}_{j,k}$ are Y

$$APE_{i,j} = A, B_i, B_j, Id_{i,j}, \overline{Pid}_{i,j}, SigPG(A, B_i, B_j, Id_{i,j}, \overline{Pid}_{i,j}, \overline{Am}_{i,j}, PE_{i,j}), AE_{i,j}$$

$$AE_{i,j} = A, Id_{i,j}, \overline{Pid}_{i,j}, SigPG(A, Id_{i,j}, \overline{Pid}_{i,j}, E_{i,j})$$

ment evidence obtained from *PG*. The success or abortion of each $s_{i,j}$ depends not only on the successful/aborted $PE_{i,j}$ corresponding to $s_{i,j}$, but also on the success/abortion of the transaction $t_{j,k}$ that follows $s_{i,j}$ (meaning successful/aborted payment evidence $\overline{E}_{j,k}$ computed by B_j in $t_{j,k}$). Thus, B_i is ensured that $s_{i,j}$ is successfully completed only if it receives from B_j two successful payment evidences: $PE_{i,j}$ and $\overline{E}_{j,k}$. Consequently, B_i is assured that either the whole sequence of transactions starting with $s_{i,j}$ is successfully completed or aborted. Finally, C receives from B_1 the corresponding $PE_{0,1}$ and $\overline{E}_{1,k}$, assuring him about success or abortion the entire complex transaction. In *PCTCI* we identify 3 possible behaviors of B_j:

1. B_j receives a request from B_i in $s_{i,j}$ and to fulfill it, he decides to send a new request to B_k in $s_{j,k}$.
2. B_j receives an aggregate request from B_i in $s_{i,j}$ and to fulfill it, he decides to send $k-j$ requests to $B_{j+1}, \ldots, B_k$ in an aggregate transaction $at_{j,k}$.
3. B_j receives an optional request from B_i in $s_{i,j}$ and to fulfill it, he sends at most $k-j$ requests to $B_{j+1}, \ldots, B_k$ in an optional transaction $ot_{j,k}$.

For each of the above behaviors of B_j, we describe a sub-protocol played by him. So, the *Chained Transaction sub-protocol (CTP)* corresponds to case 1, *Aggregate Transaction sub-protocol (ATP)* to case 2, and *Optional Transaction sub-protocol (OTP)* to case 3. *PCTCI* consists of *CTP*, *ATP*, *OTP*, *Resolution 1 sub-protocol (Res1)* and *Resolution 2 sub-protocol (Res2)* that will be described in the next sections.

Table 3: Chained Transaction sub-protocol played by B_j.

1. $B_i \rightarrow B_j : PO_{i,j}$
2. **if** $(B_j \in P)$ $B_j \rightarrow PG : PR_{i,j}$
3. $PG \rightarrow B_j : \{PE_{i,j}\}_{PkB_j}$
4. $B_j \rightarrow B_i : \{PE_{i,j},\ Cert(B_j)\}_{PkB_i}$
5. **else** $B_j \rightarrow B_k : PO_{j,k}$
6. **if** ($B_k \rightarrow B_j : \{PE_{j,k},\ \overline{E}_{k,l}\}_{PkB_j}$ in $s_{j,k}$,
7. with $PE_{j,k}.Resp = Y$ and $\overline{E}_{k,l}.Resp = Y$)
8. $B_j \rightarrow PG : PR_{i,j}$
9. $PG \rightarrow B_j : \{PE_{i,j}\}_{PkB_j}$
10. **if** ($PE_{i,j}.Resp$=Y) $B_j \rightarrow B_i : \{PE_{i,j},\ \overline{E}_{j,k}\}_{PkB_i}$
11. **else** $B_j \rightarrow B_i : \{PE_{i,j}\}_{PkB_i}$; *Res1*$(B_j)$ **end if**
12. **else if** ($B_k \rightarrow B_j : \{PE_{j,k}, \overline{E}_{k,l}\}_{PkB_j}$, with
13. $PE_{j,k}.Resp \neq A$) *Res2*(B_j) **end if**
14. $B_j \rightarrow PG : \{PE_{j,k},\ PM_i,\ OI_i\}_{PkPG}$
15. $PG \rightarrow B_j : \{PE_{i,j}\}_{PkB_j}$
16. $PG \rightarrow B_i : \{PE_{i,j}\}_{PkB_i}$
17. **end if** **end if**

4.1 Chained Transaction Sub-Protocol

In this section, we will describe *CTP* played by an intermediary or provider B_j that receives a product purchase request from C or other intermediary B_i in $s_{i,j}$. *CTP* played by B_j is presented in Table 3. We consider that in any subtransaction, a provider supplies only one individual product. If B_j is a provider, then in $s_{i,j}$ he delivers to B_i the product requested by him. If B_j is an intermediary, then he can receive from B_i a request for an individual, aggregate or optional product. In this case, B_j initiates a new $s_{j,k}$ with B_k to acquire the product requested by B_i.

In $s_{i,j}$, B_i sends $PO_{i,j}$ to B_j to buy a physical product. B_i builds $PO_{i,j}$ by encrypting with B_j's public key of the payment message PM_i and the order information OI_i. PM_i contains the payment information PI_i provided by B_i and the signature of B_i on PI_i, both encrypted with PG's public key. PI_i consists of card number Cn_i, one-time password Otp_i issued by bank, the identifier $Id_{i,j}$ and the amount $Am_{i,j}$. Each $s_{i,j}$ is uniquely identified by an identifier $Id_{i,j}$ generated as follows: $Id_{i,j} = Id_{r,i}N_{i,j}$, where $N_{i,j}$ is a fresh random number generated by B_i and $Id_{r,i}$ is the identifier of the previous subtransaction $s_{r,i}$. If $i = 0$, then $Id_{r,i}$ is the empty string. So, the identifier of a subtransaction is the sequence of all numbers generated in all previous subtransactions until it. OI_i includes the identities of the intermediaries, the product identifier $Pid_{i,j}$, $Id_{i,j}$, $Am_{i,j}$ and B_i's signature on these information.

Upon reception of $PO_{i,j}$, B_j checks the order information. If B_j is provider (line 2), then he sends $PR_{i,j}$ to PG to get payment from B_i. $PR_{i,j}$ is built from PM_i, $Pid_{i,j}$ and B_j's signature on the information described in Table 2. PG checks PI_i's authenticity, and checks Cn_i and Otp_i to verify that B_i is authorized to use the card. By verifying B_j's signature, PG is ensured that B_i and B_j agreed on $Pid_{i,j}$, $Am_{i,j}$ and $Id_{i,j}$.

PG sends to B_j an aborted $PE_{i,j}$ (*Resp*=A) in case some of above checks are failed. If all checks are successful, PG sends the payment message to the bank. Depending on B_i's account balance, the bank makes or not the transfer in the B_j's account providing to PG a successful (*Resp*=Y) or aborted $PE_{i,j}$. PG sends $PE_{i,j}$ to B_j (line 3) and stores it together with $PR_{i,j}$.

B_j decrypts $\{PE_{i,j}\}_{PkB_j}$, checks $PE_{i,j}$'s authenticity and sends $\{PE_{i,j}, Cert(B_j)\}_{PkB_i}$ to B_i, where $Cert(B_j)$ is the provider certificate. Upon reception, B_i checks the authenticity of $Cert(B_j)$ and $PE_{i,j}$.

If B_j is not the provider (line 5), then he stores $PO_{i,j}$ and initiates $s_{j,k}$ with B_k by sending $PO_{j,k}$ to buy $Pid_{i,j}$ requested by B_i. In this case, to respond the request received from B_i, B_j must ensure that either all the transactions that follows $s_{i,j}$ have been successfully completed, or all have been aborted.

$s_{j,k}$ is aborted if B_j and B_k received the aborted $PE_{j,k}$. $s_{j,k}$ is successful if B_k received a successful $PE_{j,k}$ and B_j received two successful evidences: $PE_{j,k}$ and $\overline{E}_{k,l}$. $\overline{E}_{k,l}$ is the evidence in $t_{k,l}$ that B_k sends to B_j to inform it that $t_{k,l}$ was successfully finished.

$\overline{E}_{j,k}$ is defined depending on the type of transaction initiated by B_j with B_k. As we can see in def. (2.1) Table 2, if B_j is provider, then $\overline{E}_{j,k}$ is $Cert(B_j)$. In case B_j initiates $s_{j,k}$ with B_k, then $\overline{E}_{j,k}$ is $E_{j,k}$ from the successful $PE_{j,k}$ (def. (2.2) Table 2).

We remark that the role of $\overline{E}_{k,l}$ is essential in the way the sequence of transactions starting with $s_{i,j}$ is finished. Therefore (at line 6) in $s_{j,k}$, B_j waits to receive $PE_{j,k}$ and $\overline{E}_{k,l}$ from B_k. The success of both $PE_{j,k}$ and $\overline{E}_{k,l}$ (line 7) ensures B_j that the sequence of transactions starting with $s_{j,k}$ is successfully finished and consequently (line 8) B_j sends $PR_{i,j}$ to PG in $s_{i,j}$. $PR_{i,j}$ includes the identifier $\overline{Pid}_{i,j}$ of the product successfully purchased by B_j in $s_{j,k}$ (def. (1.2) Table 2), and the amount $\overline{Am}_{i,j}$ (see Table 2). B_j checks $PE_{i,j}$ and if it is successful, then sends it together with $\overline{E}_{j,k}$ to B_i (line 10). B_i checks the success of $PE_{i,j}$, PG's signatures to verify the evidence's authenticity and also the corresponding identifiers to ensure the freshness of evidences and their belonging to successive subtransactions. If all checks are satisfied, then $PE_{i,j}$ ensures B_i that $s_{i,j}$ was successful and the successful $\overline{E}_{j,k}$ ensures B_i that $s_{j,k}$ was successful. If B_j receives from PG an aborted $PE_{i,j}$, then he sends it to B_i (line 11) as a proof of $s_{i,j}$ abortion, and applies *Res1* to abort the sequence of transactions starting with $s_{i,j}$. *Res1* will be detailed below in Sect. 4.4.

Table 4: Aggregate Transaction sub-protocol played by B_j.

1. $B_i \rightarrow B_j : PO_{i,j}$
2. $a = 0$;
3. $B_j \Rightarrow \{B_m / j+1 \leq m \leq k\} : PO_{j,m}$
4. $B_m \rightarrow B_j : \{PE_{j,m},\ \overline{E}_{m,l}\}_{PkB_j}$ in $s_{j,m}$, $j+1 \leq m \leq k$
5. **for** $(m = j+1;\ m \leq k;\ m = m+1)$
6.   **if** (not $(PE_{j,m}.Resp = Y$ and $\overline{E}_{m,l}.Resp = Y))$
7. $a = m$; **break**; **end if** **end for**
8. **if** $(a = 0)$
9. $B_j \rightarrow PG : PR_{i,j}$
10. $PG \rightarrow B_j : \{PE_{i,j}\}_{PkB_j}$
11.   **if** $(PE_{i,j}.Resp = Y)$ $B_j \rightarrow B_i : \{PE_{i,j},\ \overline{E}_{j,k}\}_{PkB_i}$
12.   **else** $B_j \rightarrow B_i : \{PE_{i,j}\}_{PkB_i}$; $Res1(B_j)$ **end if**
13. **else** $B_j \rightarrow PG : \{PE_{j,j+1}, ..., PE_{j,k}, PM_i, OI_i\}_{PkPG}$
14.   **for** $(m = j+1;\ m \leq k;\ m = m+1)$
15.   **if** $(PE_{j,m}.Resp{=}Y)$ $PG \rightarrow B_j : \{APE_{j,m}\}_{PkB_j}$
16. $PG \rightarrow B_m : \{APE_{j,m}\}_{PkB_m}$
17. $Res1(B_m)$ **end if**
18.   **else if** $(PE_{j,m}.Resp \neq A)$ $Res2(B_j)$ **end if**
19.   **end if** **end for**
20. $PG \rightarrow B_j : \{PE_{i,j}\}_{PkB_j}$
21. $PG \rightarrow B_i : \{PE_{i,j}\}_{PkB_i}$
22. **end if**

If B_j receives from B_k an invalid message, or a successful $PE_{j,k}$ but $\overline{E}_{k,l}$ is missing or contains at least an aborted evidence (lines $12-13$), then he applies *Res2* sub-protocol to abort the sequence of transactions starting with $s_{j,k}$. *Res2* sub-protocol will be detailed in Sect. 4.5. Further, to abort $s_{i,j}$, B_j sends to PG (line 14) a request containing PM_i, OI_i and the aborted $PE_{j,k}$. PG checks $PE_{j,k}$, PM_i and OI_i for their authenticity, and aborts $s_{i,j}$ by generating the aborted $PE_{i,j}$ and sends it to B_i and B_j.

Therefore, after applying *CTP* by B_j, either all transactions starting with $s_{i,j}$ are successfully finished, or all aborted.

4.2 Aggregate Transaction Sub-Protocol

Table 4 describes *ATP* played by B_j that receives an aggregate request $PO_{i,j}$ from B_i in $s_{i,j}$ and to fulfill it, he initiates an aggregate transaction $at_{j,k}$. *ATP*'s messages are represented in Fig. 2. After B_j checks $PO_{i,j}$, he initiates $at_{j,k}$ by simultaneously sending $PO_{j,m}$ to the set of intermediaries $\{B_m / j+1 \leq m \leq k\}$ (line 3). The product required by B_j to B_m in each $s_{j,m}$, is a component of the aggregate product required by B_i.

How B_j responds to the request from B_i depends on the success/failure of all transactions that follows $s_{i,j}$. So, before responding to B_i's request, B_j must ensure that $at_{j,k}$ is successful or aborted. Thus, at the line 4, B_j receives from B_m, the payment evidences

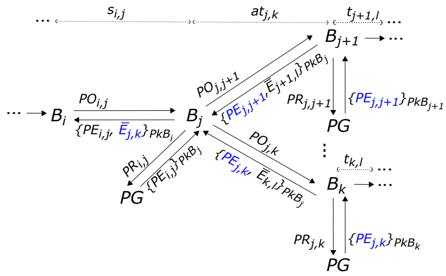

Figure 2: *ATP*'s messages flow played by B_j.

corresponding to the request sent by him to B_m in $s_{j,m}$. In this step, either B_j receives the evidences from all $B_{j+1}, \ldots, B_k$ or only from a part of them, or from none (e.g., due to network communication errors). The variable a is used to store the first aborted substransaction from $at_{j,k}$, if it exists. In the **for** loop (lines 5-7), B_j checks if all $s_{j,m}$ from $at_{j,k}$ are successful. If in iteration m of the **for** loop, both evidences received by B_j from B_m in $s_{j,m}$ are successful ($PE_{j,m}$ in $s_{j,m}$ and $\overline{E}_{m,l}$ from $t_{m,l}$), then B_j is ensured that all transactions starting with $s_{j,m}$ are successful. Otherwise, a will store m (line 7).

If after **for** loop, a is 0, then $at_{j,k}$ is successful and B_j sends $PR_{i,j}$ to PG in $s_{i,j}$ (line 9). $PR_{i,j}$ includes $\overline{Pid}_{i,j}$ that contains the sequence of product's identifiers successfully purchased by B_j in all successful subtransactions from $at_{j,k}$ (def. (1.3) Table 2). On reception of a successful $PE_{i,j}$ from PG, B_j sends it together with $\overline{E}_{j,k}$ to B_i (line 11). In this case, $\overline{E}_{j,k}$ is the sequence of all successful evidences $E_{j,j+1}, \ldots, E_{j,k}$ (def. (2.3), Table 2). Upon reception, B_i is ensured that $s_{i,j}$ and $at_{j,k}$ are successful. If B_j receives from PG an aborted $PE_{i,j}$, then $s_{i,j}$ is aborted. In this case, B_j sends $PE_{i,j}$ to B_i (line 12) and applies *Res1* to abort the sequence of transactions starting with $s_{i,j}$.

If after **for** loop, $a \neq 0$, then a will store a value m such that $s_{j,m}$ is the first aborted subtransaction from $at_{j,k}$. In this case, at least $s_{j,m}$ from $at_{j,k}$ is aborted. But, there are subtransactions of $at_{j,k}$ that are successfully completed (at least $s_{j,j+1}, \ldots, s_{j,m-1}$). Consequently, $at_{j,k}$ has both aborted and successful subtransactions. To obtain fairness, we must ensure that all sequences of transactions starting with $s_{j,j+1}, \ldots, s_{j,k}$ are aborted. For this, B_j sends to PG (line 13) a request containing $PE_{j,j+1}, \ldots, PE_{j,k}$, PM_i and OI_i. PG checks the authenticity of all received components, and if checks are passed, he sends to the bank B_j's request to abort each successful $s_{j,m}$ (line 15). The bank cancels the transfer from B_j into B_m's account, aborts the successful $PE_{j,m}$ by generating the corresponding $APE_{j,m}$ as in Table 2, and sends it to PG. Also, PG forwards $APE_{j,m}$ to B_j and B_m as a proof of $s_{j,m}$'s abortion (lines 15-16). Further, *Res1*

Table 5: Optional Transaction sub-protocol played by B_j.

1.	$B_i \rightarrow B_j : PO_{i,j}$
2.	$a = 0$;
3.	**for** $(m = j+1;\ m \leq k;\ m = m+1)$
4.	$B_j \rightarrow B_m : PO_{j,m}$
5.	**if** $(B_m \rightarrow B_j : \{PE_{j,m}, \overline{E}_{m,l}\}_{PkB_j}$,
6.	with $PE_{j,m}.Resp = Y$ and $\overline{E}_{m,l}.Resp = Y)$
7.	$B_j \rightarrow PG : PR_{i,j}$
8.	$PG \rightarrow B_j : \{PE_{i,j}\}_{PkB_j}$
9.	**if** $(PE_{i,j}.Resp{=}Y)$ $B_j \rightarrow B_i : \{PE_{i,j}, \overline{E}_{j,k}\}_{PkB_i}$
10.	**else** $B_j \rightarrow B_i : \{PE_{i,j}\}_{PkB_i}$; $Res1(B_j)$ **end if**
11.	$a = m$; **break**;
12.	**else if** $(B_m \rightarrow B_j : \{PE_{j,m}, \overline{E}_{m,l}\}_{PkB_j}$,
13.	with $PE_{j,m}.Resp \neq A)$ $Res2(B_j)$ **end if**
14.	**end if** **end for**
15.	**if** $(a = 0)$
16.	$B_j \rightarrow PG : \{PE_{j,j+1}, ..., PE_{j,k}, PM_i, OI_i\}_{PkPG}$
17.	$PG \rightarrow B_j : \{PE_{i,j}\}_{PkB_j}$
18.	$PG \rightarrow B_i : \{PE_{i,j}\}_{PkB_i}$ **end if**

is applied by B_m to abort the sequence of transactions starting with $s_{j,m}$ (line 17). In case of $PE_{j,m}$ is missing or cannot be understood by *PG* (line 18), then *Res2* sub-protocol is applied by B_j to abort the sequence of transactions starting with $s_{j,m}$. *PG* aborts $s_{i,j}$ by generating the aborted $PE_{i,j}$ and sends it to B_i and B_j.

Therefore, after applying *ATP* by B_j, either all transactions starting with $s_{i,j}$ are successfully finished, or all aborted.

4.3 Optional Transaction Sub-Protocol

OTP played by B_j is described in Table 5. B_j receives an optional request $PO_{i,j}$ from B_i in $s_{i,j}$ (line 1) and he initiates the optional transaction $ot_{j,k}$ (**for** loop). B_j initiates $s_{j,j+1}, \ldots, s_{j,k}$, in this order (line 4), until one of these is successfully completed or all of them are aborted. The variable a is used to store the first successful substransaction from $ot_{j,k}$, if it exists. If in the mth iteration of **for** loop, B_j receives both successful $PE_{j,m}$ and $\overline{E}_{m,l}$ from B_m (lines 5-6), then B_j is ensured that $s_{j,m}$ is successfully completed. This corresponds to the success of $ot_{j,k}$. In this case, B_j sends $PR_{i,j}$ to *PG* (line 7). If B_j receives a successful $PE_{i,j}$ from *PG*, then he sends it together with $\overline{E}_{j,k}$ (=$E_{j,m}$) to B_i (line 9). Upon reception, B_i is ensured that $s_{i,j}$ and $ot_{j,k}$ were successful. Otherwise, if B_j receives from *PG* an aborted $PE_{i,j}$, then he sends it to B_i (line 10) and applies *Res1* to abort the sequence of transactions starting with $s_{i,j}$. If in mth iteration, B_j receives from B_m an invalid message, or a successful $PE_{j,m}$ but $\overline{E}_{m,l}$ is missing or aborted (lines 12-13), then he applies *Res2* sub-protocol.

Table 6: Resolution 1 Sub-protocol.

$Res1(B_j)$
for $(r = j+1;\ r \leq j+n;\ r = r+1)$
if $(PE_{i,j}.Resp = A$ and $PE_{j,r}.Resp = Y)$
$B_j \rightarrow PG : \{PE_{i,j}, PE_{j,r}\}_{PkPG}$
else if (exists $APE_{i,j}$ and $PE_{j,r}.Resp = Y)$
$B_j \rightarrow PG : \{APE_{i,j}, PE_{j,r}\}_{PkPG}$ **end if** **end if**
$PG \rightarrow B_j : \{APE_{j,r}\}_{PkB_j}$
$PG \rightarrow B_r : \{APE_{j,r}\}_{PkB_r}$
$Res1(B_r)$
end for

If the value of a is 0 after **for** loop, then $ot_{j,k}$ is aborted. In this case, to maintain fairness, B_j sends to *PG* (line 16) a request to abort $s_{i,j}$. *PG* checks the received request, generates the aborted $PE_{i,j}$ and sends it to B_i and B_j to abort $s_{i,j}$. As a result, after applying *OTP* by B_j, either $s_{i,j}$ and exactly one option from $ot_{j,k}$ are successfully finished, or all subtransactions starting with $s_{i,j}$ are aborted.

4.4 Resolution 1 Sub-Protocol

Let be n, the maximum number of subtransactions in any aggregate or optional transaction. If in the complex transaction, $s_{i,j}$ is the first aborted subtransaction from the sequence of transactions starting with $s_{i,j}$, then fairness in *PCTCI* is not ensured. This is because all the transactions $t_{j,j+n}, t_{j+1,j+n+1}, \ldots, t_{j+n,j+2n}, \ldots$, from the transactions sequence that follows $s_{i,j}$ are successful, and $s_{i,j}$ is aborted. So, there is an unfair case for B_j: he has paid for products in the successful $t_{j,j+n}$, but he cannot supply them to B_i. To ensure fairness, B_j applies *Res1* described in Table 6 to abort the sequence of transactions starting with $s_{i,j}$.

Each iteration r of the **for** loop recursively aborts the sequence of transactions starting with $s_{j,r}$, where $j+1 \leq r \leq j+n$. For the aborted $s_{i,j}$, B_j has either an aborted $PE_{i,j}$ or an $APE_{i,j}$. To abort $s_{j,r}$, B_j sends $PE_{i,j}$/$APE_{i,j}$ and the successful $PE_{j,r}$ to *PG*. *PG* checks them, aborts $s_{j,r}$ by generating $APE_{j,r}$, and sends it to B_j and B_r. Next, *Res1* is recursively applied by each next intermediary B_r.

4.5 Resolution 2 Sub-Protocol

If in $s_{i,j}$, B_i sends $PO_{i,j}$, but he receives from B_j invalid evidences, or a successful $PE_{i,j}$ but $\overline{E}_{j,k}$ is missing or contains at least an aborted evidence, then fairness is not ensured. In this scenario, there is an unfair case for B_i: after he sends $PO_{i,j}$, he waits from B_j evidences that ensures him about successful or aborted $s_{i,j}$, but he does not receive evidences to confirm that. In this case, for any $s_{i,j}$, we define a timeout t in which

B_i waits the payment evidences from B_j. If t expires and B_i receives from B_j invalid evidences, or a successful $PE_{i,j}$ but $\overline{E}_{j,k}$ is missing or contains at least an aborted evidence, then B_i initiates *Res2* with *PG* to abort the sequence of transactions starting with $s_{i,j}$.

For this, B_i sends to *PG* a request containing PM_i , OI_i and the invalid evidences (if these are received). *PG* checks PM_i , OI_i and any invalid evidences. If *PG* finds in its database a successful $PE_{i,j}$, then *PG* aborts it by generating $APE_{i,j}$. If *PG* finds in its database an aborted $PE_{i,j}$, then will send it to B_i and B_j. If *PG* doesn't find $PE_{i,j}$, then he generates an aborted $PE_{i,j}$. In all cases, *PG* sends the aborted evidence to B_i and B_j as a proof of aborting $s_{i,j}$. Further, B_j applies *Res1* to abort the sequence of transactions starting with $s_{i,j}$.

As a result, fairness is also ensured for B_i.

5 FORMAL VERIFICATION

To formally verify *PCTCI*, we use Cl-AtSe (Constraint-Logic based Attack Searcher) model checker (Turuani, 2006) from AVISPA (Automated Validation of Internet Security Protocols and Applications) (Vigano, 2006), widely used by academic and industry researchers in the field of security protocols. An overview of the AVISPA tool and how to use it to formally verify complex protocols is provided in (Bîrjoveanu and Bîrjoveanu, 2022). The formal proof that *PCTCI* satisfies the security requirements demands the formal proof that each of the sub-protocols *ATP*, *OTP* and *CTP* of *PCTCI* satisfies the corresponding security requirements. A detailed description of *PCTCI*'s formal verification and full specification can be found at (Bîrjoveanu and Bîrjoveanu, 2023)). Our specification includes also use cases that require application of *Res1* and *Res2* sub-protocols.

Our results regarding the formal verification of *ATP*, *OTP* and *CTP* using Cl-AtSe prove that the following security requirements are satisfied: strong fairness, strong authentication, confidentiality, non-repudiation and integrity. In (Bîrjoveanu and Bîrjoveanu, 2023), we show how *PCTCI* derives its security requirements based on security requirements guaranteed in each of its sub-protocols.

6 CONCLUSIONS

In this paper, we proposed an e-commerce protocol with applications in supply chain, that distinguishes from the existing solutions by enabling the intermediary agents to perform aggregate, chained and optional transactions. All these features add complexity compared to the tradition e-commerce model. So, guaranteeing security requirements like strong fairness, effectiveness, timeliness, non-repudiation and confidentiality is challenging. We overcome this challenge and formally proved using AVISPA that our proposal ensures the required security properties.

REFERENCES

AlTawy, R., ElSheikh, M., Youssef, A., and Gong, G. (2017). Lelantos: A blockchain-based anonymous physical delivery system. In *PST'17, 15th Annual Conference on Privacy, Security and Trust*. IEEE.

Bîrjoveanu, C. V. and Bîrjoveanu, M. (2020). Fair exchange e-commerce protocol for multi-chained complex transactions. In *17th International Joint Conference on e-Business and Telecommunications - Volume 2: ICE-B*. SCITEPRESS.

Bîrjoveanu, C. V. and Bîrjoveanu, M. (2022). *Formal Verification of Multi-party Fair Exchange E-Commerce Protocols. In: Secure Multi-Party E-Commerce Protocols. SpringerBriefs in Computer Science*. Springer.

Bîrjoveanu, C. V. and Bîrjoveanu, M. (2023). *Formal Verification of PCTCI*. https://github.com/PCTCIDevelopers/PCTCI-Verification Last accessed May 23, 2023.

Carbonell, M., Sierra, J. M., and Lopez, J. (2009). Secure multiparty payment with an intermediary entity. *Computers & Security*.

Draper-Gil, G., Zhou, J., Ferrer-Gomila, J. L., and Hinarejos, M. F. (2013). An optimistic fair exchange protocol with active intermediaries. *International Journal of Information Security*.

Ferrer-Gomila, J. L. and Hinarejos, M. F. (2021). A multiparty contract signing solution based on blockchain. *Electronics*.

Liu, Y. (2009). An optimistic fair protocol for aggregate exchange. In *FITME'09, 2nd International Conference on Future Information Technology and Management Engineering*. IEEE.

Onieva, J., Zhou, J., Lopez, J., and Carbonell, M. (2004). Agent-mediated non-repudiation protocols. *Electronic Commerce Research and Applications*.

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*.