Modular Fair Exchange Protocols for Electronic Commerce

Holger Vogt* Henning Pagnia Felix C. Gärtner†
Darmstadt University of Technology
Department of Computer Science
D-64283 Darmstadt, Germany
{holgervo|pagnia|felix}@informatik.tu-darmstadt.de

Abstract

Recently, research has focused on enabling fair exchange between payment and electronically shipped items. The reason for this is the growing importance of Electronic Commerce and the increasing number of applications in this area. Although a considerable number of fair exchange protocols exist, they usually have been defined for special scenarios and thus only work under particular assumptions. Furthermore, these protocols provide different degrees of fairness and cause different communication overhead.

The purpose of this paper is to present a unifying solution to the problem. We do this by defining a suite of protocol modules which allow to compose protocols where the achieved degree of fairness can be enhanced step by step. The advantage of the stepwise approach is that after each step one can decide if the provided degree of fairness is acceptable or if one is willing to spend more in order to reach a higher degree of fairness. We show the applicability of our approach by deriving a novel efficient fair exchange protocol.

1. Introduction

“Electronic commerce” via the Internet is currently one of the most rapidly increasing markets. In e-commerce, companies use the network for advertising as well as for supporting their business transactions, while most often the Internet is used for internal communication, marketing and support. There is also a steadily increasing number of providers which also sell their products electronically. One of the next major challenges in electronic commerce will probably be the establishment of pay-per-use applications for digital services, i.e. services which can be entirely rendered via an electronic network. Examples for such services are the delivery of video or audio data, the purchase of computer software, the transfer of digital money, the writing of a receipt, or querying a database, but also the provision of telephone lines or Internet access.

A common characteristic of electronic services is that they normally cannot be revoked, i.e., once the service has been granted then the service provider has no effective means to force the recipient to return it. This is particularly true if the two business partners reside in different countries with differing regional law regulations. Therefore, the exchange of two digital services should take place simultaneously in order to guarantee fairness for both involved parties. Unfortunately, real simultaneousness can in general not be achieved because digital services cannot be granted instantaneously. The reason for this is that any type of data always requires a certain amount of time to be transmitted. Hence, during the exchange either party might intentionally interrupt the transmission at any time or the network itself might fail, thereby interrupting communication. If at this time one party has already completed its service but the other party has not, then the exchange was unfair.

As already mentioned, services can be either the delivery of arbitrary (digital) items or the provision of a service, like the provision of a communication link. For the latter type of service *gradual exchange* protocols [14] can be used. The basic idea behind gradual exchange is to repeatedly grant small low-value portions of the services. Hence, interrupting the exchange can only lead to one party gaining a small advantage over the other. Thereby the amount of “unfairness” which a participating party may experience is minimized. A precondition for gradual exchange protocols is that the services in question must be divisible into parts of “near-to-equal” value. Obviously, the smaller these parts become the more communication overhead increases. Conversely, splitting the service into larger parts leads to a

[*] This author’s work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the PhD program (Graduiertenkolleg) “Enabling Technologies for Electronic Commerce” at Darmstadt University of Technology.

[†] This author’s work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the PhD program “Intelligente Systeme für die Informations- und Automatisierungstechnik” at Darmstadt University of Technology.

non-negligible loss in case the protocol is interrupted. In order to avoid this situation a different protocol for fairly exchanging the individual parts is required. For services or items which are not divisible in the sense described above, such a protocol must be used anyway.

The first protocols of this type (and also the simplest ones that have appeared in the literature) always involve the active participation of a trusted third party (also called "trustee") in every run. Such protocols have been presented by Bürk and Pfitzmann [5] and by Franklin and Reiter [6]. Requiring the active participation of a trusted third party in every exchange has some obvious drawbacks (such as the potential performance bottleneck or the need for permanent availability). These drawbacks can be partly circumvented by *optimistic* fair exchange protocols [2–4]. In optimistic exchange protocols both participating parties try to handle the exchange on their own and only call for the participation of a trusted third party if something went wrong during the exchange. If the protocol is known to ensure fairness both parties are aware that they cannot gain an advantage by acting maliciously. Therefore, the situation in which the assistance of the trustee is required is not likely to happen in practice.

Protocols which do not involve a trusted third party require special item properties in order to work correctly (for example the divisibility of money in Jakobsson's *ripping coins* [8]) or make it necessary to resolve a dispute externally. In the latter case it is important that sufficient evidence is gathered to make the participants concern provable. This can be done, for example, by using the notion of a publicly visible *blackboard* [11] or by using the existing Internet infrastructure [13].

The fair exchange protocols which have been presented in the literature are diverse and at first sight appear incomparable, even in the amount of fairness they offer. In this paper we present a unifying approach to describe fair exchange protocols. By analyzing the exchange process we show that many existing protocols are in fact a composition of different protocol modules with distinct functionality. By separating the concerns and identifying these modules we are able to construct a new and even more efficient protocol for fair exchange.

The remainder of the paper is structured as follows. We discuss different notions of fairness in Section 2. Subsequently, we present our modular approach to fair exchange protocols in Section 3. This is where we show how to compose given modules into a suite of protocols solving the fair exchange problem for different levels of fairness. We conclude our paper with a discussion of our approach and some future research directions in Section 4.

2. Fairness

An intuitive way to define fairness is the following: An exchange is both, *completed* and *fair* if both parties have received the desired item. If neither party receives nor loses anything valuable then the exchange is incomplete but still fair. All other outcomes are unfair since one party has gained an advantage over the other. A protocol is called fair if under all valid conditions the exchange always ends fair.

A straightforward way to guarantee fairness is to design a protocol such that after each protocol step fairness holds. (Otherwise, the party which has gained an advantage could immediately interrupt the protocol which now ends unfair.) One possibility to achieve this is to use an active third party which first collects both items and then — after checking their validity — performs the swap. Obvious disadvantages of this protocol exist:

1. The third party must be completely trusted, i.e., it must follow the protocol and particularly not collude with either party.

2. The third party cannot be implemented stateless since it must wait for the items of both parties before the exchange can commence. This implies a non-negligible memory overhead and the need for complex mechanisms for crash recovery.

3. A considerable amount of work is delegated to the third party, resulting in high computational load.

We will discuss methods alleviating these problems later in this section.

2.1. Validation of items

A problem which is inherent to all exchange protocols is how to check the items. In order to be able to do this it is important that a sufficiently detailed specification exists for both items. For some kind of items, for example digital money, the validation is rather simple. Another example is a widely used software package which can be checked by computing a cryptographic hash value and comparing it against a trusted reference value which is publicly available [6]. Problems with this solution however can occur if the software contains a serial number or an individual watermark for copyright protection. There are other items which are difficult to check: E.g., a common description of a software package usually contains a list of features which cannot be checked during a formal verification process. Promises like "high performance" are not accurate enough in order to be verified formally. What the customer expects from this will usually not meet reality. Therefore, for an accurate validation a complete and formal specification is required. Besides the fact that in some cases this is impossible to obtain, in most other cases it is very costly. Consequently, for the exchange we can at best guarantee that a delivered item meets a given specification. But we cannot guarantee that it meets the other party's expectations which might go beyond. So overall, we must assume that for the items a sufficiently accurate specification exists against which they can be verified (although this might be costly).

2.2. A hierarchy of fairness definitions

Several definitions of fairness have been proposed [1, 7, 16]. The most prominent definition is the one by Asokan [1] in which he distinguishes between strong and weak fairness. For weak fairness it is required that — in case of a failed exchange — either party can prove that it has behaved correctly, i.e., it has followed the prescribed exchange protocol. The proofs must then be shown to an arbiter outside of the system who has the power to establish fairness, usually by forcing both parties into cooperation. The problem with this is that in most countries it is still unclear how such a proof must look like in order to fulfill local law regulations. In any case, a lawsuit is expensive and its outcome might be rather uncertain. Therefore, it is desirable to resolve as many conflicts as possible within the exchange system itself. If the third party is sufficiently powerful it can automatically process the proofs and decide how to proceed. The advantage of this is that conflicts are now automatically processed within the exchange system, thus improving the degree of fairness.

In extension to the definitions of Asokan [1], we propose the following hierarchy of fairness guarantees:

F_6: Fairness can be guaranteed automatically by the system without further cooperation of the other party.

F_5: Fairness can be guaranteed automatically by the system with eventual cooperation of the other party.

F_4: Fairness can be achieved automatically by the system through providing a compensation for a suffered disadvantage.

F_3: Fairness can only be guaranteed outside the system without further cooperation of the other party.

F_2: Fairness can only be guaranteed outside the system with eventual cooperation of the other party.

F_1: Fairness can only be achieved outside the system by providing a compensation for a suffered disadvantage.

F_0: No fairness.

The fairness definitions F_4 to F_6 are supposed to be stronger than the others because conflicts can be resolved automatically without the need for a subsequent external dispute. Strong fairness by the definition of Asokan [1] corresponds to F_6. As Asokan does not make any assumptions about the willingness of the parties to cooperate, F_5 can also be regarded as strong fairness. Gärtner et al. [7] call F_5 *eventually strong fairness*. The difference between F_6 and F_5 lies in the additional assumption made about the participants, i.e., whether they can be eventually forced to cooperate. In practice, this can be achieved by using a trusted computing environment such as a smart card. The categories F_3 and F_2 match Asokan's weak fairness definition. The categories F_1 and F_4 describe a different fairness concept in which it is assumed that a non-delivered item can be substituted by a different one (e.g., a payment) which compensates the loss. Because this does not match the original intention of the exchange process we have ranked compensation as a method to achieve fairness which is weaker than the others.

2.3. Special item properties

Special properties of the exchanged items can help the third party to resolve conflicts. In this section, we describe two of these properties in more detail, namely "generatability" and "revocability" [1].

2.3.1. Generatability

A generatable item is an item which can be generated by the trustee in case the receiving party can prove that it has behaved correctly. There are different methods to make an item generatable, among them are the following:

1. A party forwards a copy of its item to the trustee who stores it for a possible subsequent dispute. The party is provided with a signed receipt which can be presented to any other party as a proof that the item is generatable by the trustee.

2. A party encrypts its item by a random key. This key is then deposited at the trustee who returns a receipt for it, which the party can from now on use as a proof for the generatability of the key. Note, that this receipt cannot be regarded as a proof for the generatability of the item, since it cannot be guaranteed that the encrypted item can successfully be decrypted.

3. A party encrypts its item by a random key and ensures that this key can be decrypted by the trustee. This can be done with the help of a public key cryptosystem: the party encrypts the random key with the trustee's public key and forwards this — as part of the exchange

protocol — to the other party. The latter cannot decrypt this random key and hence not the item, but the trustee could do so (provided that the correct item was correctly encrypted).

The burden which is placed on the trustee decreases from method 1 to method 3. While in method 1 the trustee must store the entire item he only needs to store the decryption key in method 2. Method 3 is the most efficient one in terms of storage space: the trustee can use a single key — namely the own private key — for decrypting any item which was made generatable.

2.3.2. Revocability

An item is called revocable if the trustee can revoke it in case it has sufficient evidence to do so. Revocable items are, for example, payments since payment systems usually support revocability. Other items which might be revocable are digital signatures or certificates which provide the right to use a service. It should be noted that items must not be revocable by a party itself. Otherwise, after a correctly terminated fair exchange, one of the parties could easily revoke the delivered item and thus gain an unfair advantage over the other party. Only the trustee should be allowed to revoke items.

3. Modular fair exchange protocols

In this section we show how different notions of fairness can be realized by combining appropriate program modules to an exchange protocol as shown in Figure 1. The advantage of this modular approach is that for different scenarios suitable solutions can be composed. These solutions may depend on the properties of the exchanged items, the power of a third party, the effort which is acceptable for the exchange, or on other properties like anonymity of the parties. We first describe the underlying system model. Then we present the required modules and discuss possible implementations.

3.1. System model

We consider a system to consist of a finite set of processing elements (also called nodes) which communicate through asynchronous message passing. The parties involved in a fair exchange (customers, vendors, trustees) are assumed to reside on distinct nodes which usually are geographically separated. Messages are sent through a communication subsystem which allows direct communication between any two nodes in the system regardless of the underlying physical topology. Message passing is point-to-point and reliable with FIFO delivery order. We place no timeliness restrictions on the relative processing speed of individual nodes so that we have the *asynchronous* model of distributed systems [15]. While trustees are assumed to follow the protocol correctly, customers and vendors can act maliciously by stopping to proceed further in a protocol or by sending corrupted messages. We assume however that such incorrect behavior can be detected either by a party which follows the protocol correctly or by a trustee.

3.2. Definition of fair exchange modules

In an exchange protocol at least two parties are involved: The customer C and the vendor V. Some actions also require the cooperation of a trusted third party or trustee T. The customer has an item i_C and the vendor possesses the item i_V. Although in most cases, the customer's item will be a payment, we use i_C as a more universal notion for it here.

3.2.1. Module M_1: Negotiate

In a first step the customer and the vendor negotiate about the exchange. They have to agree on a specification (i.e., a formal description of each other's item) which enables them to verify whether the item received during the exchange protocol is the one which was expected. When both parties know which items shall be exchanged, they also agree on which fair exchange protocol should be used and which modules are used in order to implement it. Additionally, they agree on the name of the trustee possibly involved if the protocol relies on one. If the protocol uses compensation as a method for re-establishing fairness (cf. fairness definitions F_1 and F_4) then the two parties must also agree on an appropriate compensation. After completion of "Negotiate" the exchange itself can be started.

3.2.2. Module M_2: Prepare to exchange

The vendor promises that he will deliver his item i_V after the customer has sent the item i_C. This is a verifiable commitment which can be used in an external dispute (e.g., a lawsuit) in case that the vendor has misbehaved. If the vendor refuses to give this commitment, the exchange protocol is aborted. In this case, fairness F_6 is achieved since nothing valuable has been exchanged yet.

3.2.3. Module M_3: Exchange

The customer sends i_C to the vendor, who checks this item against the specification. If it is the expected item, he sends i_V to the customer, who also checks the specification obtained in module M_1. The outcome of this action can be that

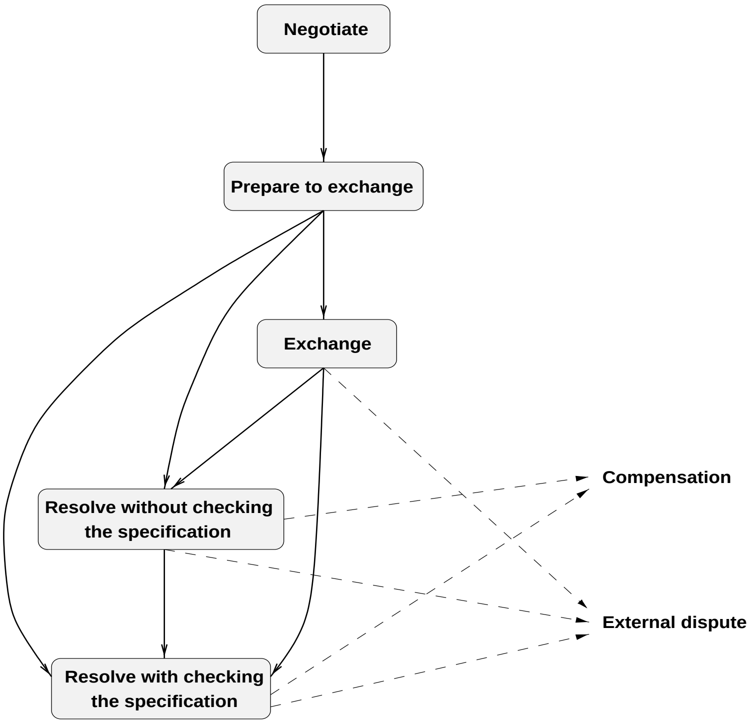
Negotiate
Prepare to exchange
Exchange
Resolve without checking the specification
Resolve with checking the specification
Compensation
External dispute

Figure 1. Composing fair exchange protocols by using the modular approach

1. both parties have the expected item,
2. none of them has an item, or
3. the vendor has i_C, while the customer has received nothing.

In the first two cases fairness F_6 is achieved, and the protocol can terminate at this point. In case 3, fairness must be re-established in one of the subsequent steps. One possible solution is to start an external dispute which however, guarantees only a lower degree of fairness (F_1, F_2, or F_3). Alternatively, the modules M_4 and M_5 can be used.

3.2.4. Module M_4: Resolve without checking the specification

If item i_V is generatable then the trustee tries to generate i_V for the customer. If the trustee succeeds, fairness F_6 is recovered. If not, either an external dispute has to be started or module M_5 must be invoked. Alternatively, compensation can be invoked if it was pre-arranged in module M_1. If the delivered item i_V does not match its specification (obtained during module M_1), module M_5 should be started. In most cases, we expect the trustee not to run into problems when trying to generate the item, because the vendor has already committed himself to deliver his item during module M_2 which can subsequently be used as a proof against him.

3.2.5. Module M_5: Resolve with checking the specification

The trustee checks the specification of the vendor's item i_V against the delivered or generated item. If this verification fails or if i_V is not available, the trustee has two possibilities: He can either use compensation (achieving fairness F_4) if pre-arranged or he can re-establish the state, in which neither the vendor nor the customer has the other's item (fairness F_6). If this cannot be done, fairness cannot be guaranteed within the system and an external dispute must be started.

3.2.6. Combining the modules

We can now set up fair exchange protocols by combining these modules. This approach is very flexible, because it provides several protocol variations: if after the execution of a module the fair exchange is completed, then subsequent steps using the other modules are not needed. It is also possible to use only a subset of the five modules by simply omitting one or more modules. However, the sequence of the modules must not be changed because this would result in incorrect protocols. Figure 1 gives an overview of the possible combinations. If a solution outside of the system is acceptable, the exchange protocol can also end with an external dispute.

3.3. Module implementations

In this section, we provide pseudo-code for possible implementations of the modules. The exchange is hereby considered to be the exchange between payment (which was i_C in the previous section) and the product that the customer wants to buy (which was i_V). In the following, we do not consider compensation as a method to gain fairness in order to ease presentation. It can however be easily incorporated into the implementations. We use the notation $\langle$event$\rangle$: $\langle$description$\rangle$ to describe the individual steps of the implementation, where $\langle$event$\rangle$ can be the sending of a message from participant X to Y (designated by $X \rightarrow Y$) or some local computation of a participant (designated by its name). The $\langle$description$\rangle$ is a brief explanation of the type of message sent or the type of actions performed locally. We assume that secure cryptosystems are available and denote encryption and decryption functions using key x by e_x and d_x, respectively. We use the capital letters D_x and E_x whenever an asymmetric cryptosystem is applied for party x. A participant can also produce digital signatures. To ease reading, we will abbreviate signing a message m and obtaining a signature s by $s := \textit{sign}(m)$. Finally, we also assume the availability of a strongly collision-free cryptographic hash function h. All these assumptions do not impose restrictions on the practicality of our approach, since sufficient means exist to realize these functions in practice [9].

3.3.1. First implementation of module M_1

A straightforward way to implement the negotiation module is the following:

Protocol I_{1a}

$C \rightarrow V$	:	spec$_{prod}$, T, set of possible protocols supported by C
$V \rightarrow C$	:	spec$_{pay}$, T, set of possible protocols supported by V

The customer and the vendor exchange the specification of the items which they want to receive and agree on a trustee T which can possibly be invoked. They also determine which exchange protocols are acceptable to both of them. Then the customer dynamically chooses one of these protocols with which he proceeds. For the following implementations it is assumed that for every message it is known to which protocol it belongs. This prevents various attacks that can be constructed by mixing steps from different protocols.

3.3.2. First implementation of module M_2

The "Prepare to exchange" module can be implemented like this:

Protocol I_{2a}

$C \rightarrow V$	:	order product
V	:	choose a random key R
		encrypt product with R, i.e.
		$\textit{EP} := e_R(\text{product})$
		compute hash $H := h(\textit{EP})$
		encrypt R for the trustee, i.e.
		$R_T := E_T(R)$
		$\textit{Sig}_V := \textit{sign}(\text{spec}_{prod}, \text{spec}_{pay}, T, H, E_T(R))$
$V \rightarrow C$	:	$\textit{Sig}_V$, *EP*, R_T
C	:	compute hash $H := h(\textit{EP})$
		verify signature $\textit{Sig}_V$

In this implementation the customer receives the product encrypted with a random key R, the key R encrypted with the trustee's public key, and a signature from the vendor to commit on this exchange. The main idea is that the further exchange process is reduced to the exchange of payment and R. Furthermore, the trustee will be able to compute R, after the vendor has sent the correct value R_T.

After execution of this module, it is still possible to abort the exchange if, for example, signature verification has revealed a bad signature.

3.3.3. First implementation of module M_3

The exchange can be implemented like this:

Protocol I_{3a}

$C \rightarrow V$	:	payment
V	:	check payment
$V \rightarrow C$	:	R
C	:	decrypt product, i.e.
		product $:= d_R(\textit{EP}) = d_R(e_R(\text{product}))$
		check product against spec$_{prod}$

This type of exchange is called an optimistic exchange [2], because no third party is required unless a conflict occurs. As most exchange processes can be assumed to run without failures, optimistic protocols can substantially reduce the load that is put on the third party. If a conflict occurs, the customer must decide which actions should be used in order to re-establish fairness.

3.3.4. First implementation of module M_4

Module M_4 is a solution for re-establishing fairness, if the previous modules failed to achieve fairness. It can be implemented like this:

Protocol I_{4a}

```
$C \rightarrow T$ : payment, spec$_{prod}$, spec$_{pay}$, $T$, $H$, $R_T$, $Sig_V$
$T$ : verify signature
      decode key: $R := D_T(R_T)$
      check payment
      deposit payment
$T \rightarrow C$ : $R$
$C$ : decrypt product, i.e.
      product $:= d_R(EP) = d_R(e_R$(product))
      check product against spec$_{prod}$
```

The trustee decrypts R, which he sends to the customer in exchange for the payment. The product is decrypted by the customer, so that he has to check it by himself. This implementation obviously relies on the vendor, who has to provide the correct values, so that the trustee can generate the correct key R. Since the vendor might have sent incorrect values, fairness cannot be completely guaranteed.

3.3.5. First implementation of module M_5

In this implementation the trustee checks the specification of the product. When he detects a failure, he must be able to revoke the payment, if it was already sent to the vendor. This guarantees fairness F_6 after the execution of this module.

Protocol I_{5a}

```
$C \rightarrow T$ : payment, spec$_{prod}$, spec$_{pay}$, $T$, $EP$, $R_T$, $Sig_V$
$T$ : verify signature
      check payment
      decode key $R := D_T(R_T)$
      decode product, i.e.
      product $:= d_R(EP) = d_R(e_R$(product))
      check product versus spec$_{prod}$:
      if "product OK" then
         deposit payment
         $T \rightarrow C$ : product
      elseif "product not OK and payment was
            already sent to the vendor" then
         revoke payment
```

3.3.6. Second implementation of module M_5

One of the last steps of implementation I_{5a} is that the trustee sends the product to the customer. The following alternative implementation can be used in order to minimize the amount of transferred data:

Protocol I_{5b}

```
$C \rightarrow T$ : payment, spec$_{prod}$, spec$_{pay}$, $T$, $EP$, $R_T$, $Sig_V$
$T$ : verify signature
      check payment
      decode key $R := D_T(R_T)$
      decode product, i.e.
      product $:= d_R(EP) = d_R(e_R$(product))
      check product versus spec$_{prod}$:
      if "product OK" then
         deposit payment
            $T \rightarrow C$ : $R$
               $C$ : decrypt product, i.e.
                     product $:= d_R(EP))$
      elseif "product not OK and payment was
            already sent to the vendor" then
         revoke payment
```

This solution works better than I_{5a} if, for example, the customer has only a slow modem connection to the trustee. In this case I_{5b} should be used for minimizing the transmission time.

3.3.7. Second implementation of module M_2

The implementations above are particularly designed for optimistic fair exchange protocols. For an exchange involving an active trustee, the modules M_2 and M_3 can be implemented in the following manner.

Protocol I_{2b}

```
$C \rightarrow T$ : payment, spec$_{prod}$, spec$_{pay}$
$V \rightarrow T$ : product, spec$_{prod}$, spec$_{pay}$
```

3.3.8. Second implementation of module M_3

Because the trustee possesses both, payment and product, he can check in advance if these items match their specification. If the checks fail, the exchange will abort without losing fairness.

Protocol I_{3b}

```
$T$ : check payment, check product versus spec$_{prod}$
$T \rightarrow C$ : payment
$T \rightarrow V$ : product
```

3.4. Composing protocols

The module implementations described in the previous section can be combined in different ways according to the rules displayed in Figure 1. The most important compositions are listed below:

P_1: $\langle I_{1a}, I_{2b}, I_{3b} \rangle$
This is the basic active exchange protocol for fairness F_6 which is used in several protocols (e.g. [5], [6]).

P_2: $\langle I_{1a}, I_{2a}, I_{3a}, \text{external dispute} \rangle$
This is a protocol for a weaker fairness (F_2/F_3). A detailed discussion of this class of protocols can be found in [1].

P_3: $\langle I_{1a}, I_{2a}, I_{3a}, I_{4a}, \text{external dispute} \rangle$
This is another weak fairness (F_2/F_3) protocol for the scenario of non-revocable payments.

P_4: $\langle I_{1a}, I_{2a}, I_{3a}, I_{5a} \rangle$
This is an optimistic protocol which ensures fairness F_6 inside the system. See [12] for a complete description.

P_5: $\langle I_{1a}, I_{2a}, I_{3a}, I_{5b} \rangle$
This describes a variation of P_4 with a fewer amount of transferred data in the case of conflict.

P_6: $\langle I_{1a}, I_{2a}, I_{3a}, I_{4a}, I_{5a} \text{ or } I_{5b} \rangle$
This is a very efficient optimistic fair exchange protocol providing fairness F_6, which will be elaborated on at the end of this section.

P_7: $\langle I_{1a}, I_{2a}, I_{4a}, \text{external dispute} \rangle$
This is an alternative implementation for the active protocol P_1. The NetBill payment protocol [16] uses a similar idea to ensure fairness.

P_8: $\langle I_{1a}, I_{2a}, I_{5a} \text{ or } I_{5b} \rangle$
This protocol is also a variation of the active protocol P_1.

The protocols P_1, P_2, P_4, and P_7 correspond to the existing protocols cited in the short protocol descriptions given above. The other protocols are so far unpublished.

P_3 is a novel variation of a weak fairness protocol which first makes an attempt to re-establish fairness automatically inside the system. Only if this fails an external dispute is started.

As already stated above, P_5 is a variation of P_4. The only difference between the two protocols is that at the end of Module M_5 the trustee does not send the product to the customer but the key R. The customer then decrypts the product himself.

In protocol P_8 after the exchange was prepared by the customer and the vendor, the trustee is used to finally perform the swap of product and payment. It should be noted that although P_8 invokes I_{5a} it is not necessary to use a payment system with revocability. The reason for this is that the payment was never sent to the vendor during the previous steps and therefore it does not need to be revoked. Different to this, the protocols P_4, P_5, and P$_6$ require the payment to be revocable.

Protocol P_6 is an interesting novel protocol which is now described in more detail.

Discussion of protocol P_6. After the negotiation phase in I_{1a} some preconditions for P_6 must be checked: In module M_5 the trustee should be able to revoke the payment. This is necessary for the (rather improbable) case that the vendor has received the payment during I_{3a} but the trustee is not able to generate the product in I_{4a}, due to a misbehaving vendor in I_{2a}. Without revocability the customer can either initiate an external dispute (this is equal to protocol P_2 or P_3) or he must rely on an active trustee (this ends up in protocol P_1 or P_7). The advantage of revocable payment is that with its help fairness can be guaranteed inside the system and that the trustee is not actively involved in any fault-free exchange. This is true for protocol P_6 which tries to involve the trustee as seldom as possible. This effectively reduces communication traffic caused at the trustee, since we can assume that most exchanges are executed without experiencing any failure. Thus, in the normal case the protocol terminates immediately after performing the exchange in I_{3a}.

The implementation I_{4a} attempts to re-establish fairness in the case of a failure. As in most cases the trustee will be able to compute the key R, so that the customer can decrypt the product. A lot of failures can be solved by invoking I_{4a}. Furthermore the implementation of I_{4a} is very efficient because only a minimal set of values has to be transferred to and from the trustee.

It should not be necessary to call I_{5a} or I_{5b} very often, so that even expensive computations during these modules (e.g. for checking the product) might be tolerable.

It should be noted that an additional property of this protocol is that it allows the customer to remain anonymous when buying a product from the vendor. In this case however, both the payment-system and the communication connection between the two parties must support anonymity.

4. Conclusions

Fair exchange is a problem of substantial practical significance in electronic commerce. Products, payments and services must be exchanged fairly to ensure the continuing growth of the electronic marketplace. In order to increase the trust that participating parties place in exchange services it is important to state precisely the guarantees of different protocols with respect to fairness and efficiency. When this has been done, customers and vendors can select a certain protocol that suits the application or situation needs best. For example, if products of considerable value (like a new CAD-program) are exchanged, both parties will probably favor a protocol which guarantees a strong fairness F_4–F_6 even if this comes at a higher cost (because they might have to pay a trustee). On the other hand, both parties might be willing to agree on a weakly fair F_1–F_3 protocol if they simply exchange the latest football results.

The fair exchange protocols which have been presented in literature are diverse and at first sight incomparable even in the degree of fairness they offer. We have analyzed the exchange process and we have tried to show that a lot of these protocols are in fact a composition of several modules with distinct functionality. By separating the concerns and identifying these modules we were able to construct a set of new and even more efficient protocols for fair exchange.

Moreover, by using our compositional approach it is now possible to construct exchange protocols for a given level of fairness and given item properties almost dynamically. In practical settings this enables the vision that a customer can choose an item from an on-line catalogue, select the desired level of fairness and have the rest of the exchange be executed automatically; it is no longer necessary for the user to select a specific protocol which is clearly a step towards more user friendliness.

The compositionality of our protocols also lends itself to modular verification of the protocols in the direction of the well-known software engineering paradigm of stepwise refinement. For this, it is however necessary to formalize the module specifications much more rigorously. We expect that this will reveal some potential for methodological improvements of our approach since the distinction between module specification and implementation has not been sharp enough. This is obvious from the fact that some combinations of modules (e.g., $\langle I_{1a}, I_{2b}, I_{3a} \rangle$) are not possible. A clearer differentiation between both concepts will lead the way to a more rigorous verification. This and the implementation of the given suite of protocols within our experimental testbed will be the focus of our continuing work.

References

[1] N. Asokan. *Fairness in electronic commerce*. PhD thesis, University of Waterloo, May 1998.

[2] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In T. Matsumoto, editor, *4th ACM Conference on Computer and Communications Security*, pages 8–17, Zurich, Switzerland, Apr. 1997. ACM Press.

[3] N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 86–99, May 1998.

[4] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In K. Nyberg, editor, *EUROCRYPT '98*, Lecture Notes in Computer Science, pages 591–606. Springer-Verlag, 1998. A longer version is available as Technical Report RZ 2973 (#93019), IBM Research, November 1997 at `http://www.zurich.ibm.com/Technology/Security/publications/1997/ASW97b.ps.gz`.

[5] H. Bürk and A. Pfitzmann. Value exchange systems enabling security and unobservability. *Computers & Security*, 9(8):715–721, 1990.

[6] M. K. Franklin and M. K. Reiter. Fair exchange with a semi-trusted third party. In T. Matsumoto, editor, *4th ACM Conference on Computer and Communications Security*, pages 1–5,7, Zurich, Switzerland, Apr. 1997. ACM Press.

[7] F. C. Gärtner, H. Pagnia, and H. Vogt. Approaching a formal definition of fairness in electronic commerce. In *Proceedings of the International Workshop on Electronic Commerce (WELCOM'99)*, Lausanne, Switzerland, Oct. 1999.

[8] M. Jakobsson. Ripping coins for fair exchange. In L. C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology—EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 220–230. Springer-Verlag, 21–25 May 1995.

[9] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[10] S. Mullender, editor. *Distributed Systems*. Addison-Wesley, second edition, 1993.

[11] H. Pagnia and R. Jansen. Towards multiple-payment schemes for digital money. In R. Hirschfeld, editor, *Financial Cryptography: First International Conference, FC '97*, volume 1318 of *Lecture Notes in Computer Science*, pages 203–215, Anguilla, British West Indies, 24–28 Feb. 1997. Springer-Verlag.

[12] H. Pagnia and H. Vogt. Exchanging goods and payment in electronic business transactions. In *Proceedings of the Third European Research Seminar on Advances in Distributed Systems (ERSADS)*, Madeira Island, Portugal, Apr. 1999.

[13] J. Riordan and B. Schneier. A certified e-mail protocol with no trusted third party. In *Proceedings of the 13th Annual Computer Security Applications Conference*, Dec. 1998.

[14] T. W. Sandholm and V. R. Lesser. Equilibrium analysis of the possibilities of unenforced exchange in multiagent systems. In C. S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 694–703, San Mateo, Aug. 20–25 1995. Morgan Kaufmann.

[15] F. B. Schneider. What good are models and what models are good? In Mullender [10], chapter 2, pages 17–26.

[16] J. D. Tygar. Atomicity in electronic commerce. In *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing (PODC'96)*, pages 8–26, New York, May 1996. ACM.