Supporting Fair Exchange in Mobile Environments

Holger Vogt (holgervo@informatik.tu-darmstadt.de), Felix C. Gärtner (felix@informatik.tu-darmstadt.de) and Henning Pagnia (pagnia@computer.org)
Department of Computer Science, Darmstadt University of Technology, D-64283 Darmstadt, Germany

Abstract. Mobile commerce over the Internet always includes the exchange of electronic goods. Fair exchange protocols establish fairness and ensure that both participants can engage in the exchange without the risk of suffering a disadvantage (e.g., losing their money without receiving anything for it). In general, fair exchange protocols require the continuous availability of an external *trusted third party* (TTP), a dedicated site which is trusted by both participants. Implementations of TTPs for fair exchange have been proposed to be based on carefully secured Internet hosts in order to establish trust. In this paper we present solutions to the fair exchange problem in mobile environments, where customers frequently disconnect from the network and thus continuous availability of the external TTP is not given. Our approach utilizes tamper-poof hardware on the customer's side partly taking over the duties of the TTP. Besides supporting disconnected operations our approach also allows the proper handling of *time-sensitive items* (i.e., items which lose value over time), a feature which previous protocols lack.

Keywords: mobile commerce, fair exchange, disconnected operations, tamper-proof hardware, time-sensitive items

1. Introduction

Among the many technical issues involved in building infrastructures for mobile commerce, a central one is to develop solutions for the exchange of two intangible items over an electronic network. Music files, electronic documents, football results, and weather information are examples of intangible items which can be traded in return for electronic money. The parties involved in the exchange usually start in a state of mutual distrust. Solutions must then ensure that neither party is in danger of suffering a disadvantage, like losing its money without receiving anything for it. Protocols solving this task are called *fair exchange protocols.*

Fair exchange protocols are particularly difficult to design for mobile environments where network availability is rather unpredictable. If a vendor sends for example a music file to a customer, there is no guarantee that it will reach the latter within a certain time. Consequently, fair exchange protocols operating in these types of environments can merely guarantee *eventual delivery* of an item assuming that the net-

work communication will eventually be reestablished. Eventual delivery though is too weak for items that lose value over time, like location dependent information. For instance, consider Alice walking through New York. Through her mobile device she requests information about restaurants in the area of 5th Avenue and 42nd Street. While waiting for the reply Alice continues to walk around town, and as the information finally appears on her display, she has already reached the far end of Central Park. Alice is reluctant to walk all the way back now and so the information is of little value to her. However, the service provider has already billed her the full price of the service. Such items which lose their value over time are called *time-sensitive* (Asokan, 1998). Time-sensitive items occur in many applications, among them location based services, like the restaurant guide described above, or stock exchange data services.

Besides time-sensitivity, *security* is a further aspect which renders fair exchange difficult. In order to overcome mutual distrust in distributed business environments, protocols rely on a *trusted intermediary* to enable the fair exchange. Such an intermediary is a dedicated computer which is usually called *trustee* or *trusted third party* (TTP). The machine either actively participates in the exchange or is contacted to resolve a dispute in case something went wrong. In any case it is essential that both parties which engage in a fair exchange *trust* this machine to provide reasonably high availability and to correctly follow its protocol. Particularly the latter is difficult to guarantee if the computer is, for example, located at the vendor's office. Then no guarantee can be given that the hard- and software has not been tampered with. This has given rise to the use of *trusted tamper-proof hardware* in fair exchange protocols such as the *trusted processing environment* (TPE) of Wilhelm (1999) to solve fair exchange with mobile agents (Pagnia et al., 2000).

Today, tamper-proof hardware to implement a TPE, like the IBM 4758 PCI card (IBM, 2002), is available but, unfortunately, it is also quite expensive. On the other hand, smart cards, a different form of tamper-proof hardware, are rather cheap but their limited memory and processing power render them unsuitable for implementing a full-blown TPE. Karjoth (2000) has shown that in some cases a fully featured TPE is not needed to perform certain business transactions. The idea here is that only *critical operations* (like encrypting or signing messages) need to be performed on trusted hardware while a lot of processing work can be performed through untrusted hardware handling solely encrypted items. In this paper we elaborate this idea, and the central question which we pursue is the following: How can fair exchange be implemented with smart cards in a mobile computing environment?

The difficulties in designing protocols under the above design goals stem from the restricted processing power of the trusted hardware (i.e., the smart card) as well as from user mobility leading to temporary disconnection from the network. Throughout this paper, we consider the case in which a vendor sells an electronic item to a mobile user who pays for the item electronically. Under the assumption that a mobile user utilizes a tamper-proof smart card we develop solutions for fair exchange.

Our contributions are manyfold. Firstly, only a few papers (Zhou and Lam, 1999; Vogt et al., 2001) exist so far discussing the use of secure hardware for fair exchange and an extensive investigation of the hardware's additional capabilities compared to software implementations is still missing. Secondly, we show how to use smart cards in order to build efficient protocols for the exchange of arbitrary data including time-sensitive information. Existing protocols can give no guarantees on when a desired time-sensitive item will actually be received. In contrast to that, our protocols can be adapted so that items are transmitted to the user's device in advance and that before delivery the customer can decide whether he still wants to receive the item or not. As this decision requires no network communication, timely delivery is assured, even if the customer is disconnected from the vendor during this decision. In order to avoid that a time-sensitive item loses its value, it is important that the time between the vendor transmitting the item and the smart card delivering it to the customer is sufficiently short. We thirdly present a protocol minimizing this time by utilizing a technique we call *deferred item checking*. In deferred item checking the exchanged item needs not be checked beforehand. This allows to delay the costly checking process or even to delegate it to an external TTP.

More efficient protocols can be constructed by using different notions of fairness. As a further contribution we present a fair exchange protocol providing only incentives to behave correctly, instead of strictly enforcing fairness. This protocol allows the local exchange of items even when the smart card temporarily cannot send messages to the vendor during the exchange. This is especially advantageous for the case of asymmetric communication channels between the customer and the vendor.

The paper is structured as follows: We first summarize the basic terminology of fair exchange in Section 2 and then present the system assumptions as well as a basic solution for hardware-supported fair exchange in Section 3. Techniques for efficient implementation of these protocols are also proposed in this section. In Section 4 we present extended exchange protocols which ensure fairness even for time-sensitive items. Protocols with incentives for fair behavior instead

of strict enforcement of fairness are discussed in Section 5. Finally, we compare the advantages of our suggested protocols with "conventional" protocols without hardware support in Section 6.

2. Fair Exchange Concepts

In a fair exchange two parties are involved and each of them starts with an electronic item and a formal description of what that party would like to receive in exchange for its own item. In our case the first party is a vendor V offering an arbitrary electronic item and the second party is the customer C who is willing to pay for the item with electronic money.

The majority of fair exchange protocols is designed to achieve *strong fairness* as defined by Asokan (Asokan et al., 1997; Asokan, 1998). According to this definition a *fair exchange protocol* is a protocol which implements the following three requirements between V and C:

1. **Effectiveness:** If both parties behave according to the protocol, both do not want to abandon the exchange, and both items match the description then, when the protocol has completed, C has received V's item and V has received the payment from C.

2. **Termination:** The protocol will terminate for a party which behaves correctly (i.e., according to the protocol).

3. **Fairness:** If one party does not behave according to the protocol or if one item does not match the description, no honest participant will win or lose anything valuable.

As noted in the introduction, fair exchange protocols rely on the services of a TTP, which is trusted by both participants. In fact it is impossible to achieve this kind of fairness, if the two parties try to execute the exchange without any TTP (Even and Yacobi, 1980). One way to implement fair exchange protocols is to use an active TTP (Bürk and Pfitzmann, 1990; Zhou and Gollmann, 1996; Franklin and Reiter, 1997). The TTP receives the two items, checks them, and forwards them to the respective parties. Since this renders the TTP a bottleneck, protocols have been devised (Asokan et al., 1997; Asokan et al., 1998; Zhou and Gollmann, 1997; Vogt et al., 1999) in which the two parties try to complete the exchange on their own and where an external TTP only comes into play if something went wrong. Such protocols are called *optimistic* (Asokan et al., 1997). They are efficient in the sense that no TTP is used in the faultless case. However, they require that

at least one of the two items is either revocable or generatable by the TTP. The fair exchange protocols presented in the Sections 3 and 4 achieve strong fairness and are optimistic in this sense.

Some exchange protocols accept a weaker notion of fairness and are thus able to process the exchange without a TTP. One kind of protocols is based on the idea of gradual exchange (Sandholm and Lesser, 1995; Zhou and Lam, 1999) which has the goal to reduce the advantage a misbehaving party can achieve. There the items are divided into small parts which are alternately exchanged. If these parts are sufficiently small, a communication disruption results in a loss that is still acceptable. However not all items are suitable for this kind of exchange, and splitting the exchange into several rounds causes significant communication. In a mobile environment with rather slow communication links this is not acceptable.

Other gradual exchange protocols used for the fair exchange of decryption keys (Blum, 1983; Brickell et al., 1987; Boneh and Naor, 2000) are based on the assumption that the participants have equal computing power. If one participant's computing power is significantly higher, then this party can compute the not yet transmitted bits of the key on its own while the other party cannot. This results in an unfair situation. Due to these shortcomings we will in the following not consider gradual exchange protocols.

A different approach, sometimes called *rational exchange* (Buttyàn and Hubaux, 2001; Buttyàn, 2001; Syverson, 1998), does not guarantee strong fairness but rather provides incentives to behave correctly. For example Jakobsson (1995) proposed a protocol which ensures that nobody can gain an advantage. However, the protocol cannot prevent parties from suffering a disadvantage: A cheating party can gain nothing, but it can behave in a way so that the other party loses its item. We will elaborate this idea further in Section 5 where we present a protocol establishing even stronger incentives for correct behavior than the one used by Jakobsson.

3. Fair Exchange Using Smart Cards

In this section, we describe our approach to perform fair exchange using smart cards (Vogt et al., 2001). We first describe general assumptions we make about the system and its communication. Then we sketch necessary requirements of the used smart card and present the basic protocol for fair exchange. Finally, we show how such a protocol can be implemented efficiently.

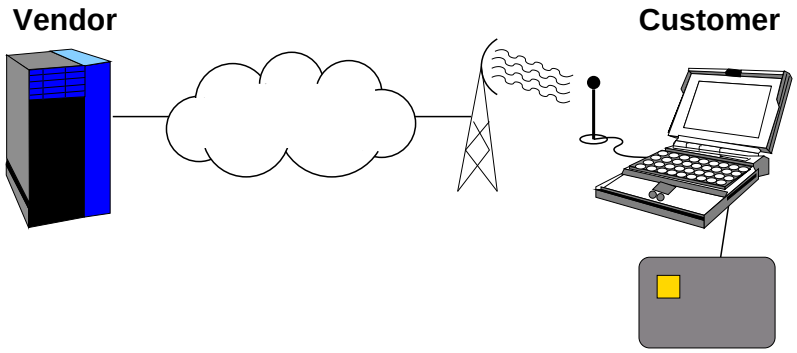

Figure 1. Smart card support for the exchange of money and intangible items.

3.1. System Assumptions

The system in which we consider fair exchange is sketched in Figure 1. We assume that both customer and vendor are connected through a communication network and that the customer's machine is equipped with a smart card as a trusted hardware device. The customer's machine might be a mobile phone, a personal digital assistant (PDA), a laptop computer or specialized hardware like a car navigation system. The smart card can be embedded into the above devices or it can also be attached through an external card reader. Smart cards are widely used for authentication, issuing digital signatures, or making payments. They are usually considered to be tamper-resistant, which means that they protects secret keys even from the owner of the smart card. Furthermore, smart cards are already available and in use for m-commerce (e.g. in GSM mobile phones). The card is assumed to have a distinguished interface, meaning that only predefined operations can be invoked with carefully chosen input and output parameters. Invoking these interface operations is only possible, while the smart card is connected to a regular computer. For the smart card this is also the only way to communicate with the vendor and vice versa.

The communication between all participants and the smart card is expected to be secure, i.e., cryptographic mechanisms are in place to protect the integrity and confidentiality of messages. We assume that an attacker and especially the owner of the smart card is limited to the following attacks: The attacker can disrupt or delay the communication with the smart card at any point of time. Data stored on the participants' computer may be deleted or modified. The smart card may permanently be disconnected or even be damaged by the owner, which destroys the smart card's stored information. In contrast, eavesdropping, replay, or forging of messages is assumed to be impossible.

A crucial part of fair exchange protocols is the ability to check whether the received item really is the expected one. The effort for this verification strongly depends on the exchanged item. Consider for example a software package for which a cryptographic hash value is publicly known. After receiving the package, the customer knows that it is the expected one only if it hashes to the published value. Other examples are items like digital concert tickets or electronic cheques that consist of a digital signature on certain messages. Such a signature can easily be verified by a smart card which has access to the public verification key. A thorough discussion about the verification of items is beyond the scope of this paper and the reader is referred to Pagnia et al. (2000).

3.2. Smart Card Requirements

If a smart card shall support the fair exchange of digital items, we have to presume that it fulfills the following requirements:

Tamper-resistance: The smart card must be protected so that it is impossible to read out secret data or to change the behavior of the card. This must even be true, if an attacker has physical access to the smart card.

Trust and authenticity of messages: Customer and vendor rely on the card correctly executing the algorithms, e.g., processing all requests correctly. As most people cannot verify the functionality of such a card, they must trust the issuer of the device that it always works as it is supposed to. To this end, anybody connecting to this card must be able to check the authenticity of all messages generated by it. This can be achieved, if the smart card contains a private key to generate digital signatures. The corresponding public key for the card's signature must be certified by some trusted authority, probably the issuer of the smart card. Any party communicating with the smart card can then verify whether a message is unaltered and has been created by the smart card itself.

Furthermore, a smart card should be able to identify the sender of messages by using digital signatures. Therefore the card must possess some built-in authentication information, e.g., the public key of a trusted certification authority. The card can then validate the customer's or the vendor's public key certificate. This prevents the smart card from being fooled by an attacker who claims to be somebody else.

Reliable state information: The smart card must be able to send signed messages describing the current state of the exchange software

Table I. The basic fair exchange protocol between the customer C and the vendor V using the customer's smart card S.

1a.	V $\rightarrow$ S	:	item, description
1b.	C $\rightarrow$ S	:	payment, description
2.	S	:	check item, check payment
3.	S $\rightarrow$ V	:	payment
4.	V	:	check payment
5.	V $\rightarrow$ S	:	acknowledge payment
6.	S $\rightarrow$ C	:	item

which it is executing. A party communicating with the card thus has a simple and reliable method to verify this state which is especially useful after communication has been disrupted and must be re-established. The smart card will provide trustworthy state information supporting the correct execution of the protocol.

Additionally, we assume that the state together with the currently processed data is stored on the card persistently, thus, shutting down the power supply will not cause any loss of information.

3.3. The Basic Protocol

Our basic fair exchange protocol is an optimistic protocol, as it does not require an external TTP in the faultless case. It enables the exchange of arbitrary items with any kind of electronic payment without any assumption about the item or the payment. In contrast, existing optimistic protocols *without* trusted hardware cannot exchange *arbitrary* items, as they make additional assumptions — namely generatability or revocability of items — to ensure fairness (Asokan, 1998, p. 25).

The protocol (shown in Table I) starts with the vendor V sending the item and the customer C delivering the payment to the smart card S (steps 1a and 1b). The card will abort the exchange in step 2, if the item or the payment does not match the description of what both parties expect. The verified payment is transferred to the vendor, who also checks its validity (steps 3 and 4). If an on-line payment system (e.g., Schoenmakers (1997)) is used, the vendor additionally has to deposit the payment at the bank to verify it. The vendor then notifies the smart card, whether he has accepted the payment (step 5). Finally, the smart card delivers the item (step 6), if the vendor has accepted the payment.

Table II. The customer starts this conflict resolution protocol, if he does not receive an acknowledgment from the vendor.

5'.	C $\rightarrow$ S	:	tell to resolve the exchange
6'.	S $\rightarrow$ B	:	payment
7'.	B	:	check and deposit payment
8'.	B $\rightarrow$ S	:	acknowledge payment
9'.	S$\rightarrow$ C	:	item

If the smart card does not receive an acknowledgment from the vendor or if the vendor falsely claims the payment to be invalid, the customer can instruct the smart card to resolve the exchange to ensure fairness. The smart card has three possibilities to react on such a customer request. If the smart card has not yet sent the payment to the vendor in step 3, it simply deletes the item and returns the payment to the customer. But if the item has been delivered to the customer in step 6, a resolve request will be ignored, as the exchange has already been finished successfully. In all other cases the help of an external TTP is needed. For simplicity we assume that the bank B offers the services of a trusted third party so that no additional communication between the bank and an external trusted third party is needed.

The protocol for resolving such a conflict utilizes the fact that the smart card stores the vendor's item after it passed the test in step 2. To ensure fairness the customer himself must not be allowed to instruct the smart card to release this item. Instead solely the bank which acts as a trusted third party has the power to do so.

The protocol for resolving an exchange is displayed in Table II. The smart card connects to the bank and sends the payment (step 6'). In step 7' the bank deposits the money on the vendor's account (if it has not already been deposited by the vendor) and notifies the smart card about this (step 8'). After receiving the bank's acknowledgment the smart card releases the item (step 9'). This guarantees that both parties obtain the expected items and that the exchange finishes achieving strong fairness.

3.4. Efficient Implementation with Smart Cards

Compared to other computing devices like a PC, smart cards have only limited storage and computing power. Therefore, items which allocate a huge amount of storage or which require an extensive computation for checking their correctness might cause problems. To overcome these

limitations, we suggest two methods, namely external storage of items and deferred item checking. Both can in principle be applied to any fair exchange protocol that relies on trusted hardware. The main idea is to "outsource" functionality and transfer duties from the smart card to the customer or the TTP.

External storage of items. The persistent storage of a smart card is limited to a few kilobytes today, which prevents it from storing larger items on the card. This can be circumvented if the smart card delegates the storage of the item in the following manner: The smart card receives and processes the item. It encrypts the item and outputs it to the customer's computer that simply saves this data to its hard disk. Later, when the item is needed again, the smart card can request it from the computer.

The smart card must remember the decryption key for this item, and additionally it must store a hash value of the item. This can be compared to the hash value of the data received from the customer's computer enabling the smart card to verify that it has received the original (encrypted) item.

Using the described mechanism the storage requirements can be reduced significantly. However, we have to take care that we do not create another bottleneck: The communication speed of the smart card is not very high and so we should avoid sending the same data back and forth. For example, if in step 1a of Table I the vendor sends the item, it is inefficient to first let the smart card read it and then output it to the customer's computer. Such a protocol step, in which the smart card receives but does not immediately process an item, should rather be implemented as follows: The vendor encrypts the item and transmits it to the customer's computer, where it is stored for later use. The decryption key is sent to the smart card, which stores this key. Now the smart card is not able to compute a hash value. We can compensate this, if both the customer and the vendor compute the hash value of the encrypted item and send it to the smart card, which only accepts if both parties sent the same hash value. Then neither the vendor nor the customer can cheat by sending a hash value that does not match the delivered encrypted item.

Deferred item checking. Another limitation of a smart card is its rather small computing power. Checking the item or the payment possibly exceeds the computing power of a smart card. In this case it will take unacceptably long to finish this computation. Thus, it is important to minimize the amount of computation which the smart card spends for verifying the items.

The main idea is to *defer* item checking, but proceed with the protocol as if the verification had been successful. In most cases the verification will succeed anyway and the outcome of the exchange will satisfy both parties. If the item is not the expected one, a party will detect this in one of the subsequent steps and react by starting a special resolution protocol. There are two possibilities of how to resolve the conflict: (1) The smart card is now provided with the item, which is then checked inside the card. Since we expect failures to occur infrequently, it should be acceptable to wait for the result of the checking process even if it takes some time. (2) If the smart card is not able to verify the item, we transfer this task to an external TTP. Verification is much easier for the TTP which is usually implemented using a high-speed server. The result of the verification is sent back to the smart card which now acts accordingly.

As an example we apply this technique to our basic exchange protocol (see Table I): We now skip the checking of the payment in step 2 because the vendor will detect any invalid payment later (step 4). If the vendor claims the payment to be invalid in step 5, the customer can send the payment either to the smart card, which resumes the exchange in step 2 with checking the payment, or to an external TTP. In the latter case we obtain a solution similar to the one in Table II, where the bank B (acting as TTP) checks the payment and forwards the result to the smart card. The smart card either aborts the exchange due to an invalid payment or otherwise reveals the item.

3.5. Discussion

Our approach shows that a smart card can partly provide services of a TTP. If the smart card checks the item, the resulting decision can be trusted by both, the customer and the vendor. But not only trust must be considered, also availability: In a mobile scenario connections are not durable and so the customer might become disconnected after having sent the payment. Although he can obtain the item as soon as he reconnects to the vendor or the bank, such a time delay is usually unacceptable for time-sensitive items. In the next section we investigate fair exchange protocols, which address this problem.

4. Fair Exchange of Time-Sensitive Data

Fair exchange of time-sensitive data is a difficult problem in unreliable networks. The main problem of solutions without secure hardware is proving *when* a message really arrived. A party can always claim that it has received the item after it has lost its value.

Table III. The extended protocol for fair exchange of time-sensitive items.

1a.	$V \rightarrow S$	:	item, description
1b.	$C \rightarrow S$	:	payment, description
2.	S	:	check item, check payment
3.	$S \rightarrow V$	:	payment
4.	V	:	check payment
5.	$V \rightarrow S$	:	acknowledge payment
6.	$S \rightarrow C$	:	ask, if the item is still valuable
7.	$C \rightarrow S$	:	tell to proceed the exchange
8.	$S \rightarrow C$	:	item

Our solution enables the customer to decide locally whether he wants to finish or abandon the exchange. As the decision is made locally, the customer is independent from communication delays. An obvious advantage of our solution is that we do not require synchronized clocks or trusted time-stamping services, which would significantly increase the implementation costs.

In this section we restrict ourself to the purchase of time-sensitive items. We propose a modification of our basic protocol from Section 3.3. Different to there, we now require payments to be revocable in order to resolve interrupted exchanges. Additionally we present an exchange protocol with a minimal delay between the vendor sending the item and the customer deciding whether he wants it to be delivered or not.

4.1. A Protocol Extension for Time-Sensitive Items

In our extended protocol (see Table III) we require revocability of payments (also called *payment cancellation*). The solution works for every payment system that supports revocation by an external TTP. Particularly, we are not restricted to payment systems based on secure hardware.

Steps 1 to 5 are identical for the basic and the extended protocol. Before revealing the item to the customer, the smart card now asks the customer, if he is still interested in the item (step 6). If the customer wants to proceed (step 7), the smart card releases the item (step 8). For the customer there is only a negligible delay between his decision in step 7 and the time he receives the item in step 8. Thus he can be sure that the time-sensitive item will still be valuable.

But if the customer does not want to receive the item anymore, he has to start the abort protocol (given in Table IV) in order regain

Table IV. The abort protocol for time-sensitive items.

7'.	C $\rightarrow$ S	:	tell to abort the exchange
8'.	S $\rightarrow$ B	:	state information
9'.	B	:	verify state, cancel payment
10'.	B $\rightarrow$ S	:	payment canceled
11'.	S	:	reset the state for this exchange

his payment. This protocol must also be used, if the customer does not receive a message in step 5 of Table III or if the merchant falsely claims a correct payment to be invalid.

Again, there are three alternatives how the smart card can react on an abort request from the customer in step 7'. (1) If the smart card has not yet sent the payment to the vendor in step 3, it simply discards the item and the payment. (2) If the item has been delivered to the customer in step 8, an abort request will be ignored, as the exchange has already finished successfully. (3) The smart card contacts the bank and asks for payment cancellation. The smart card sends its state information (step 8') which ensures to the bank that it has not yet revealed the item of the vendor. Thus, the exchange is guaranteed to be fair, after the payment has been canceled (step 9'). The bank notifies the smart card about the canceled payment (step 10') and the smart card resets its state for this exchange (step 11'), i.e., it discards the stored item and provides the customer with the necessary information about the payment revocation.

The exchange protocol guarantees fairness for the customer, as he will always get his money back, provided that he has not received the item or if he decided that the time-sensitive item became worthless to him. Because the smart card will only cancel the payment if the customer did not receive the item, the protocol also ensures fairness for the merchant.

4.2. A Minimal Delay Fair Exchange Protocol

In the previous exchange protocol of Table III the item is sent in step 1a and the customer has to wait with his purchase decision until step 7. However, for *highly time-sensitive items*, i.e., items which loose their value quickly (like "real-time" stock market information), the protocol should minimize the time delay, or otherwise the customer will often refuse to buy these items. We suggest a different fair exchange protocol (see Table V), which is designed especially for the exchange of highly time-sensitive items. This protocol has a minimal message complexity,

Table V. A minimal delay protocol for fair exchange of highly time-sensitive items.

1.	C $\rightarrow$ S	:	payment, description
2.	S $\rightarrow$ V	:	payment, description
3.	V	:	check payment
4.	V $\rightarrow$ S	:	item
5.	S $\rightarrow$ C	:	ask, if the item is still valuable
6.	C $\rightarrow$ S	:	tell to proceed the exchange
7.	S $\rightarrow$ C	:	item
8.	C	:	check item

as it only needs one message to the vendor and one back. Furthermore, it requires only the minimal number of steps between the vendor transmitting the item in step 4 and the customer deciding to take it or not in step 6.

Again we assume the payment to be revocable by the TTP. First the customer sends the payment to the smart card (step 1), which forwards it to the vendor for checking the payment (steps 2 and 3). The vendor responds to a valid payment by sending the desired item in step 4. After receiving the item, the smart card asks the customer *immediately* in step 5, if he still wants the time-sensitive item to be delivered. It is obvious that the time delay between sending the item in step 4 and asking the customer in step 5 is minimal, and thus the item will probably be still of value. If the customer wants the item to be delivered in step 6, the smart card locally transfers it to the customer's computer in step 7. Since no network communication is required for this, this can be performed without further delay. If the customer rejects the item he follows the abort protocol as described in Table IV.

In this protocol we exploit the idea of deferred item checking from Section 3.4, as the smart card checks neither the payment nor the item. Therefore the customer has to check the validity of the item in step 8. If it is not the correct item, the customer demands an additional verification from either the smart card or a TTP. After one of them assured that the item is incorrect, the bank cancels the payment which the vendor has received during this exchange. This finally re-establishes fairness for the customer. The fairness for the vendor is also enforced, as the item will only be delivered after payment and as revocation of a payment is only possible, if the item has not been delivered to the customer or if the item was incorrect.

Table VI. Exchange of time-sensitive data with incentives to behave correctly.

1a.	V $\rightarrow$ S	:	item, description
1b.	C $\rightarrow$ S	:	payment, description
2.	S	:	check item, check payment
3.	S $\rightarrow$ C	:	ask, if the item is still valuable
4.	C $\rightarrow$ S	:	tell to proceed the exchange
5a.	S $\rightarrow$ C	:	item
5b.	S $\rightarrow$ V	:	payment
6.	V	:	deposit payment
7.	V $\rightarrow$ S	:	payment OK
8.	S	:	reset the state for this exchange

5. Relying on Incentives to Achieve Fairness

In the protocols presented in Section 3 and 4 the smart card can only deliver the item to the customer after having assured that the payment was received by the vendor. This involves communication between the smart card and the vendor and thus the customer cannot access the item while being disconnected. In this section we present a protocol which enhances the customer's mobility but however does not achieve strong fairness. Instead we design our protocol in a way that it provides strong incentives for the customer to behave correctly.

The basic protocol steps (see Table VI) are as follows: The card receives the item and the payment (steps 1a and 1b), checks if both are correct (step 2), asks the customer if he wants to proceed (steps 3 and 4), and performs the exchange by immediately releasing the time-sensitive item and sending the payment to the vendor (steps 5a and 5b). This protocol guarantees that a customer gets a time-sensitive item just after having confirmed that he still wants it to be delivered.

The difference to the protocols for strong fair exchange in Section 4 is that the customer now obtains the item earlier. Furthermore, if the customer rejects the exchange in step 4, the smart card simply resets itself to ensure fairness. We do not require payments to be revocable, but assume that they can be released by the smart card and re-used in another purchase, after the customer decided to abort the exchange.

The protocol allows higher customer mobility, because the customer can disconnect after having performed step 1a. The subsequent steps including step 5a, in which the item is delivered to the customer, can be

performed off-line. When the customer reconnects later, the remaining steps will be performed.

However, in this case some problems exist concerning fairness. As the customer controls the communication with the smart card, he can block all messages to the vendor immediately after step 1a. In this case the vendor will never receive the payment in step 5b although the customer has obtained the item.

A malicious customer can cause the same effect by sending a "useless" payment. This is possible because for most payment systems the smart card will not be able to *fully* verify the payment without the help of the bank (e.g., in case of overspending). This means that the vendor might experience in step 6 that the bank refuses this payment. We solve this problem by designing the smart card in a way that it logs this exchange until it is notified by the vendor in step 7 that he received the payment. Now two possibilities exist to enforce a fair behavior of the customer which we elaborate in the following sections: (1) lock the smart card until the exchange is completed and (2) allow a certain amount of unfinished exchanges before the card is finally locked.

Obviously, both solutions cannot prevent the customer from permanently disconnecting or destroying his smart card, thereby preventing that the state of the exchange can be examined later. This attack is useless in practice if the item's value is far lower than the costs for buying a new smart card. Although strong fairness is not achieved, this is a strong incentive for the customer to correctly follow the protocol. However, for very expensive items exchange protocols achieving strong fairness should be used.

5.1. Strict Locking of the Smart Card

As a first possibility to deal with unfinished exchanges, the customer is forced to complete the exchange, if he wants to use the smart card for further exchanges. We call this *strict locking* of the smart card. This is a very strong incentive for the customer to complete the exchange, because the smart card will otherwise be useless to him. We now have to face the problem that a malicious merchant can refuse to send a confirmation message in step 7. In this case the customer needs the help of the bank to unlock his smart card. The required protocol is given in Table VII.

In order to unlock his smart card the customer can instruct the card to contact the bank (step 7'). The smart card then sends the payment to the bank (step 8'), which deposits it (step 9'). If the payment is valid the bank sends a confirmation to the smart card (step 10'), which is unlocked and which now can be used for further exchanges (step 11').

Table VII. Resolving the exchange of time-sensitive data without the help of the vendor.

7'.	C → S	:	tell to contact the bank for help
8'.	S → B	:	send exchange state and payment
9'.	B	:	verify state, deposit payment
10'.	B → S	:	payment OK
11.'	S	:	unlock and reset

5.2. Weak Locking of the Smart Card

Strict locking of the smart card might be too inconvenient to the customer. Consider the situation in which the vendor and the bank are unreachable for a longer time period and thus are not able to confirm the payment. Then the smart card remains locked and further purchases are impossible until the bank or vendor is available again.

As a concept to cope with this problem, we suggest *weak locking* of the smart card, i.e., we allow the customer to execute several exchanges before the card is locked. This can be enforced by the smart card, if it additionally checks in step 2 of Table VI whether unfinished exchanges are stored on the card. If too many exchanges are unfinished or if the total value of unfinished exchanges exceeds the amount necessary to buy a new smart card, the card rejects to support further exchanges and it remains locked until unfinished exchanges are completed or until it is unlocked using the protocol in Table VII.

The described solution is also very useful for a typical pay TV scenario: A customer wants to watch the live transmission of sports events using pay-per-view. He receives the encrypted transmission via a broadcast medium which does not support a backward channel to the service provider. The purchase of the decryption key to watch this transmission should then be supported by the weak locking version of our exchange protocol in Table VI. This ensures that the customer will only pay for watching the (highly time-sensitive) live TV event, if he receives the decryption key in time.

As we assume the application of weak locking, the purchase of the decryption key can be performed without connection to the service provider: The key is also broadcasted, but it must be encrypted for the smart card. If this key is received and forwarded to the smart card, the purchase is executed off-line and nevertheless in a fair manner. Due to weak locking the customer can pay for several TV events and still remain off-line. When he goes on-line again later, the payments will

be transferred to the service provider and the exchanges on the smart card are finished.

5.3. Discussion

The protocols for fair exchange of time-sensitive items in Section 4 require payment systems allowing a TTP to revoke payments. In contrast to this, the protocol described in the current section is able to purchase time-sensitive items even with non-revocable payments. The absence of revocability is a critical issue in anonymous payment systems (Chaum, 1983; Chaum et al., 1988). Transferring the payment back to the customer usually requires his deanonymization which conflicts with the goal of an anonymous payment system.

As we have shown time-sensitive items require immediate local delivery. Payments can only be sent to the vendor before or after item delivery. Paying before the local item delivery has been considered in Section 4 and requires revocability whereas the protocol described in the current section delivers the payment after the item. This has the drawback that a malicious customer might block the transmission of the payment resulting in unfairness. It is impossible to fully remedy this problem, but by providing strong incentives through smart card locking we can alleviate this.

6. On the Benefits of Hardware Support

Tamper-proof hardware is in common use today, e.g., in GSM mobile phones. For fair exchange it offers significant advantages which we name and investigate in this section.

There are many properties which an external TTP must provide in order to support fair exchange protocols. The properties concerning functionality consist of (1) atomicity in exchange, i.e., the basic exchange functionality, and (2) validating the items. The early protocols for fair exchange with active TTP (Bürk and Pfitzmann, 1990; Zhou and Gollmann, 1996; Franklin and Reiter, 1997) slightly obscured these different aspects by assuming dedicated computers that could be secured through standard mechanisms like firewalls. While exchange and checking functionality can potentially be offered by different servers, they remain a bottleneck, as they are needed for every single exchange. This explains why optimistic protocols (Asokan et al., 1997; Asokan et al., 1998; Zhou and Gollmann, 1997) are so useful. But even in optimistic protocols, the trust in the dedicated machine must be ensured by a strict security policy. Users must still have reason to trust that

the machine correctly follows the protocol. The amount of confidence in this machine can be reduced by providing forms of verifiability of protocol execution, i.e., a participating party can find out if the third party did not follow the protocol (Asokan, 1998).

One drawback inherent to all solutions using an external TTP is that they have no influence on the *timely delivery* of the exchanged items. Timely delivery means minimizing the amount of time between a party's final commitment to the exchange and the delivery of the corresponding item. Properties of timely delivery critically depend on the network. Of course, one can use system models with stronger assumptions in the design of fair exchange protocols, e.g., we can assume that message delivery delay is bounded. However, it is very dangerous to make such timing assumptions in practice, since the quality of network parameters is often rather uncertain and protocols may fail in unpredictable ways if their assumptions are not met. Another drawback is closely related to timing issues: Even in optimistic protocols it is assumed that the TTP is continuously available. This means that the TTP is responsive in situations where its help is needed.

Some of these drawbacks of an external TTP can be alleviated by partly placing its functionality "closer" to the participating parties, making the quality of communication more predictable. While this does not influence the communication properties of the network *per se*, this paper shows that the approach has immediate advantages for the party which has local access to a trusted hardware device: It increases the availability of TTP functionality to an extent where timely delivery local to a party is feasible. Furthermore, a trusted hardware device placed close to the customer is much less endangered to become a bottleneck since a customer does usually not engage in more than one transaction concurrently.

However, placing the TTP functionality close to a participating party has some disadvantages concerning security: Correct protocol execution has to be ensured for *both* parties, i.e., also for the remote party without local access to the trusted hardware device. This is where the additional properties of secure hardware are useful and where pure software-based solutions are insufficient. The assumptions about the security of the used hardware (e.g., that it is tamper-proof) ensure correct protocol execution and its verifiability for *both* participants. It can be argued that assuming tamper-proof hardware local to a participant reduced the exchange scenario to the "trivial" case where this participant never misbehaves. However, this is not the case for the types of tamper-proof hardware which we investigate in this paper, since parties can block the communication with the smart card or destroy it.

Additional means must be incorporated into protocols to handle this type of misbehavior.

A final point worth noting in this discussion is that in our protocols the bank operating the payment scheme often implicitly offers trusted third party services. For example, it checks the validity of the payment in the basic protocol of Section 3.3, it revokes the payment in the protocol for time-sensitive items in Section 4, and it can unlock the smart card in the protocols of Section 5. However note that these services are only necessary in the conflict resolution parts of the protocols.

7. Conclusion

We have presented fair exchange protocols in which the trusted third party is partly implemented using trusted hardware, namely a smart card, at the customer. The basic protocol presented in Section 3.3 ensures strong fairness and hence can be used to exchange arbitrary (even high priced) electronic items. We have shown that the basic protocol can be modified, resulting in a protocol which is very well suited for exchanging time-sensitive items, a property which previous fair exchange protocols lacked. Additionally, we designed a protocol for the exchange of extremely time-sensitive items that hardly tolerate a time-delay during the exchange.

Our protocols for time-sensitive items are especially well suited for exchanges in a mobile scenario, as an interrupted connection cannot prevent the smart card from timely delivering the item to the customer. In this paper we have adapted smart cards as trusted devices, but other devices like mobile phones or PDAs might also be used instead.

As an application area of our protocols providing a weakened type of fairness in Section 5 we envision the exchange of low-priced "push" services in mobile environments. With "push" based service we mean broadcast services like pay TV or traffic information where information is brought to the user's device in encrypted form and the user can *locally* engage in an exchange to receive the appropriate key for decryption. The protocol works in a timely fashion even for temporarily disconnected users. This makes our approach extremely suitable for mobile commerce.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. Holger Vogt was supported by the Deutsche Forschungsgemeinschaft (DFG) as

part of the PhD program (Graduiertenkolleg) "Enabling Technologies for Electronic Commerce" at Darmstadt University of Technology.

References

Asokan, N.: 1998, 'Fairness in electronic commerce'. Ph.D. thesis, University of Waterloo, Canada.

Asokan, N., M. Schunter, and M. Waidner: 1997, 'Optimistic Protocols for Fair Exchange'. In: T. Matsumoto (ed.): *4th ACM Conference on Computer and Communications Security*. Zürich, Switzerland, pp. 6–17, ACM Press.

Asokan, N., V. Shoup, and M. Waidner: 1998, 'Asynchronous protocols for optimistic fair exchange'. In: *Proceedings of the IEEE Symposium on Research in Security and Privacy*. pp. 86–99.

Blum, M.: 1983, 'How to Exchange (Secret) Keys'. *ACM Transactions on Computer Systems* **1**(2), 175–193.

Boneh, D. and M. Naor: 2000, 'Timed Commitments'. In: *Advances in Cryptology – CRYPTO '2000*, Vol. 1880 of *Lecture Notes in Computer Science*. pp. 236–254, Springer-Verlag.

Brickell, E. F., D. Chaum, I. B. Damgård, and J. van de Graaf: 1987, 'Gradual and Verifiable Release of a Secret'. In: *Advances in Cryptology – CRYPTO '87*, Vol. 293 of *Lecture Notes in Computer Science*. pp. 156–166, Springer-Verlag.

Bürk, H. and A. Pfitzmann: 1990, 'Value Exchange Systems Enabling Security and Unobservability'. *Computers & Security* **9**(8), 715–721.

Buttyàn, L.: 2001, 'Building blocks for secure services: Autenticated key transport and rational exchange protocols'. Ph.D. thesis, Swiss Federal Institute of Technology – Lausanne. No. 2511.

Buttyàn, L. and J.-P. Hubaux: 2001, 'Rational exchange – A formal model based on game theory'. In: *Electronic Commerce – WELCOM 2001*, Vol. 2232 of *Lecture Notes in Computer Science*. pp. 114–126, Springer-Verlag.

Chaum, D.: 1983, 'Blind Signatures for Untraceable Payments'. In: *Advances in Cryptology – CRYPTO '82*. pp. 199–203, Plenum.

Chaum, D., A. Fiat, and M. Naor: 1988, 'Untraceable Electronic Cash'. In: *Advances in Cryptology – CRYPTO '88*, Vol. 401 of *Lecture Notes in Computer Science*. pp. 319–327, Springer-Verlag.

Even, S. and Y. Yacobi: 1980, 'Relations amoung public key signature systems'. Technical Report 175, Computer Science Department, Technicon, Haifa, Israel.

Franklin, M. K. and M. K. Reiter: 1997, 'Fair Exchange with a Semi-Trusted Third Party'. In: T. Matsumoto (ed.): *4th ACM Conference on Computer and Communications Security*. Zürich, Switzerland, pp. 1–5, ACM Press.

IBM: 2002, 'The IBM 4758 PCI Cryptographic Coprocessor'. Homepage: http://www.ibm.com/security/cryptocards/.

Jakobsson, M.: 1995, 'Ripping Coins for Fair Exchange'. In: L. C. Guillou and J.-J. Quisquater (eds.): *Advances in Cryptology – EUROCRYPT '95*, Vol. 921 of *Lecture Notes in Computer Science*. pp. 220–230, Springer-Verlag.

Karjoth, G.: 2000, 'Secure Mobile Agent-Based Merchant Brokering in Distributed Marketplaces'. In: *Proceedings of the Second International Symposium on Agent Systems and Applications and Fourth International Symposium on Mobile Agents (ASA/MA2000)*, Vol. 1882 of *Lecture Notes in Computer Science*. Zurich, Switzerland, pp. 44–56, Springer-Verlag.

Pagnia, H., H. Vogt, F. C. Gärtner, and U. G. Wilhelm: 2000, 'Solving Fair Exchange with Mobile Agents'. In: *ASA/MA 2000*, Vol. 1882 of *Lecture Notes in Computer Science*. Zürich, Switzerland, pp. 57–72, Springer-Verlag.

Sandholm, T. W. and V. R. Lesser: 1995, 'Equilibrium Analysis of the Possibilities of Unenforced Exchange in Multiagent Systems'. In: C. S. Mellish (ed.): *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. San Mateo, pp. 694–703, Morgan Kaufmann.

Schoenmakers, B.: 1997, 'Security Aspects of the Ecash Payment System'. In: *COSIC '97 Course*, Vol. 1528 of *Lecture Notes in Computer Science*. pp. 338–352, Springer-Verlag.

Syverson, P.: 1998, 'Weakly Secret Bit Commitment: Applications to Lotteries and Fair Exchange'. In: *Proceedings of the 11th IEEE Computer Security Foundations Workshop (CSFW '98)*. Rockport, Massachusetts, pp. 2–13, IEEE.

Vogt, H., H. Pagnia, and F. C. Gärtner: 1999, 'Modular fair exchange protocols for electronic commerce'. In: *Proceedings of the 15th Annual Computer Security Applications Conference*. Phoenix, Arizona, pp. 3–11, IEEE Computer Society Press.

Vogt, H., H. Pagnia, and F. C. Gärtner: 2001, 'Using Smart Cards for Fair Exchange'. In: *Electronic Commerce – WELCOM 2001*, Vol. 2232 of *Lecture Notes in Computer Science*. pp. 101–113, Springer-Verlag.

Wilhelm, U. G.: 1999, 'A Technical Approach to Privacy based on Mobile Agents protected by Tamper-resistant Hardware'. Ph.D. thesis, École Polytechnique Fédérale de Lausanne, Switzerland.

Zhou, J. and D. Gollmann: 1996, 'A Fair Non-repudiation Protocol'. In: *Proceedings of the IEEE Symposium on Security and Privacy*. Oakland, CA, pp. 55–61, IEEE Computer Society Press.

Zhou, J. and D. Gollmann: 1997, 'An Efficient Non-repudiation Protocol'. In: *Proceedings of the 10th IEEE Computer Security Foundations Workshop*. pp. 126–132, IEEE Computer Society Press.

Zhou, J. and K.-Y. Lam: 1999, 'A Secure Pay-per-View Scheme for Web-Based Video Service'. In: *Public Key Cryptography – PKC '99*, Vol. 1560 of *Lecture Notes in Computer Science*. pp. 315–326, Springer-Verlag.