An Optimistic Multi-party Fair Exchange Protocol with Reduced Trust Requirements

Nicolás González-Deleito and Olivier Markowitch

Université Libre de Bruxelles
Bd. du Triomphe – CP212
1050 Bruxelles
Belgium

{ngonzale,omarkow}@ulb.ac.be

Abstract. In 1999, Bao et al. proposed [5] a multi-party fair exchange protocol of electronic items with an offline trusted third party. In this protocol, a coalition including the initiator of the exchange can succeed in excluding a group of parties without the consent of the remaining entities. We show that every participant must trust the initiator of the protocol for not becoming a passive conspirator. We propose a new protocol in which the participants only need to trust the trusted third party. Moreover, under certain circumstances, if there are participants excluded from the exchange, they can prove that a problem occurred to an external adjudicator.

1 Introduction

During the last decade, the important growth of open networks such as the Internet has lead to the study of related security problems. Fair exchange of electronic information (contract signing, certified mail, ...) is one of these security challenges. An exchange protocol is said to be *fair* if it allows two or more parties to exchange electronic information in such a way that, at the end of the protocol, no honest party has sent anything valuable unless he has received everything he expected. Those protocols often use a trusted third party (TTP) helping the participants to successfully realize the exchange.

Depending on his level of involvement in a protocol, a TTP can be said *inline*, *online* or *o ine*. Inline and online trusted third parties are both involved in each instance of a protocol, but the first one acts as a mandatory intermediary between the participants. An offline TTP is used when the participants in a protocol are supposed to be honest enough to not need external help in order to achieve fairness; the TTP will only be involved if some problem emerges. Protocols with such a TTP are called *optimistic*.

Fair exchange between two parties has been extensively studied and several solutions have been proposed in the online [8,12] as in the offline case [4,3,6,11].

The interest of [11], where an item is exchanged against a digital signature, is that the TTP produces the same signatures as those that would have been produced by the participants in a faultless scenario.

Fair exchange of electronic information between more than two parties may have applications in electronic commerce. For example, we can consider a common and generic scenario with four parties describing a ring. Let one of these parties be a customer who wants to purchase an electronic item offered by a provider; the payment is realized through the customer's and the provider's banks. This is how each participant views the exchange:

- the provider provides the expected electronic item to the customer in exchange of having his bank crediting his account;
- the customer sends a payment authorization to his bank in exchange of the desired electronic information offered by the provider;
- the customer's bank carries out the payment to the provider's bank in exchange of the payment authorization sent by the customer;
- finally, the provider's bank credits the provider's account in exchange of the payment carried out by the customer's bank.

More general topologies than the ring described above are also possible. For example, we could consider an exchange in which each participant offers items to a set of parties in exchange of items offered by another set of participants. Franklin and Tsudik gave [7] a classification of multi-party exchanges, based on the two following properties: the number of items that a participant can exchange (one or several), and the disposition of the participants (describing a ring or a more general topology).

In the multi-party case, Asokan et al. described [2] a generic optimistic protocol with a general topology. This protocol, during which a participant may receive an affidavit from the TTP instead of the expected item, achieves *weak fairness*. Franklin and Tsudik presented [7] two protocols with an online TTP, and later Bao et al. proposed [5] a protocol where the TTP is offline. Both works supposed the exchange topology as being a ring, where each participant P_i offers to participant P_{i+1} message m_i in exchange of message m_{i-1} offered by participant P_{i-1}. Of course, all subscripts are $\bmod n$, where n is the number of participants in the exchange. We will omit this hereafter.

With regard to optimistic multi-party fair exchange, it could be unrealistic to think that all the participants in a protocol execution will be honest. Designing optimistic protocols for this kind of exchanges might not have, at a first glance, much sense. However, we think that even in a scenario with a dishonest participant, a protocol with an offline TTP remains more efficient than a protocol with an online TTP.

In this paper we focus on optimistic multi-party fair exchange with a ring topology. Section 3 describes the protocol proposed by Bao et al. in [5], and its trust requirements are discussed. In section 4 we propose a variant of that protocol, where trust needs are reduced.

The next section describes the concept of verifiable encryption schemes. This technique, used by Bao et al. in their multi-party fair exchange protocol [5], will also be used in section 4.

2 Verifiable Encryption Schemes

Suppose a scenario with two participants, Alice and Bob, and a trusted third party. Let E and D be the encryption and the corresponding decryption algorithms of a public-key cryptosystem. The TTP owns a public encryption key e and a secret decryption key d of this cryptosystem. Moreover, let h be a homomorphic one-way function. Alice knows a secret message m, with $h(m)$ being public.

Alice enciphers m to $c = E_e(m)$, generates a certificate $cert_A = certify(m, c, e)$ using a public algorithm $certify()$, and sends c and $cert_A$ to Bob.

Bob checks that $cert_A$ is a correct certificate by using a public algorithm $verify()$ such that $verify(c, cert_A, h(m), e) = yes$ if and only if $h(D_d(c)) = h(m)$. Bob is then convinced that c is indeed the cipher of m under key e. Later, Bob will be able to obtain m by asking the TTP to decipher c.

A verifiable encryption scheme must satisfy [5] these two properties:

- it is computationally unfeasible for Alice to generate a certificate $cert_A$ such that $verify(c, cert_A, h(m), e) = yes$, while $h(D_d(c)) \neq h(m)$;
- and it is computationally unfeasible for Bob to get m from c without knowing d.

Asokan et al. gave [4] some examples of verifiable encryption schemes implementations. Bao et al. used [5] an implementation of a non-interactive scheme, corresponding to the description above.

3 A Multi-party Fair Exchange Protocol

In this section we briefly describe the optimistic multi-party fair exchange protocol proposed by Bao et al. [5]. (We use notations as close as possible to the ones used in the original paper.) The exchange topology is a ring.

Let P_0 be the participant that initiates the protocol. The status of P_0 is known by the TTP, who also knows all $h(m_i)$, for $0 \leq i \leq n-1$.

3.1 Main Protocol

P_0 begins the main protocol by sending to P_1 the cipher c_0 of the message m_0 along with a certificate $cert_0$ proving that c_0 is indeed the cipher of m_0, as described in the previous section.

After receiving $(c_0, cert_0)$, P_1 checks $cert_0$; if this certificate is valid, he enciphers m_1 and sends $(c_1, cert_1)$ to P_2. For $i = 2, 3, ..., n-1$, participant P_i does similarly.

When P_0 receives $(c_{n-1}, cert_{n-1})$, he checks the certificate $cert_{n-1}$ and if it is valid, he sends the message m_0 to P_1. For $i = 1, 2, ..., n-1$, after receiving m_{i-1} from participant P_{i-1}, P_i sends m_i to P_{i+1}.

These are the two rounds of the main protocol:

1. $P_i \rightarrow P_{i+1} : c_i, cert_i \quad \text{for } i = 0, ..., n-1.$
2. $P_i \rightarrow P_{i+1} : m_i \quad \text{for } i = 0, ..., n-1.$

3.2 Recovery Protocol

If all parties behave correctly, after the main protocol execution each participant should obtain his expected message. However, if some P_i does not receive m_{i-1}, he has to run the recovery protocol.

P_i begins this protocol by sending $(c_{i-1}, cert_{i-1})$ to the TTP. The latter checks the certificate $cert_{i-1}$, and, if it is valid, waits for the time[1] it takes to P_0 to obtain m_{n-1} if no problem occurs, before asking P_0 if he has received a valid $(c_{n-1}, cert_{n-1})$ during the first round of the main protocol. If P_0 answers *yes*, the TTP will accept to decipher the corresponding c_{i-1}.

The TTP does not contact P_0 each time that some other participant runs the recovery protocol: if P_0 answered *yes* in the first recovery execution then the TTP will accept further recovery requests; otherwise the TTP will reply with an *abort* message.

This call to P_0 prevents of having party P_i getting c_{i-1} deciphered without having sent $(c_i, cert_i)$.

Here are the four steps of the recovery protocol, initiated by some P_i having not received m_{i-1}. Steps 2 and 3 are only executed the first time that this protocol is invoked.

1. $P_i \rightarrow TTP : c_{i-1}, cert_{i-1}$.
2. $TTP \rightarrow P_0 : call$.
3. $P_0 \rightarrow TTP$: *yes* or *abort*.
4. $TTP \rightarrow P_i : m_{i-1}$ or *abort*.

3.3 Analysis

Bao et al. gave [5] the following definition of fairness: *an exchange protocol is called a* fair exchange protocol *if after the protocol execution no participant* P_i

[1] This time is estimated by the TTP.

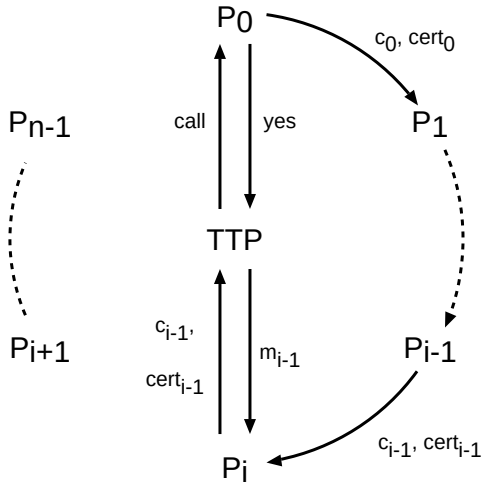

Fig. 1. An exclusion scenario

following properly the protocol is in a state where his c_i has been deciphered without having him received m_{i-1}.

A participant P_j $(1 \leq j \leq n-1)$ sends m_j to P_{j+1} only if he has received m_{j-1}. P_{j+1} can however ask the TTP to decipher c_j; he obtains the expected m_j only if P_0 received $(c_{n-1}, cert_{n-1})$ in the first round of the main protocol. If so, P_j can also ask the TTP to decipher c_{j-1} in order to get m_{j-1}.

Since P_0 sends m_0 to P_1 after receiving a valid $(c_{n-1}, cert_{n-1})$ from P_{n-1}, it is then also possible for P_0 to recover m_{n-1} if he ever does not receive it later.

Assuming that the TTP behaves as described, after the protocol execution (both, the main and, possibly, the recovery protocol) each honest P_i either has obtained m_{i-1}, either his c_i has not been deciphered.

About Trust and Passive Conspiracies As pointed out by Bao et al., in that protocol two or more parties can collude in order to exclude some other participants from the exchange. This happens only if P_0 belongs to this coalition.

Consider that P_0 colludes with P_i (figure 1). When P_i receives $(c_{i-1}, cert_{i-1})$ from P_{i-1} in the first round of the main protocol, instead of sending $(c_i, cert_i)$ to P_{i+1}, P_i could run directly the recovery protocol, asking the TTP to decipher c_{i-1}. The TTP asks then P_0 if he has received $(c_{n-1}, cert_{n-1})$, P_0 answers with a false *yes*, and the TTP deciphers c_{i-1}.

P_i will obtain m_{i-1} without having sent $(c_i, cert_i)$. This causes no harm as long as the TTP follows the protocol properly: P_1 to P_{i-1} will be able to run the

recovery protocol and obtain respectively m_0 to m_{i-2}, and P_{i+1} to P_{n-1} will not receive nor send anything: they will be simply excluded from the exchange.

By the above definition, fairness is still guaranteed for any party following properly the protocol, including those who are excluded, even if there is a participant P_i having received m_{i-1} without having sent c_i. There is here a difference between this definition of fairness and the one given by Asokan et al. [2], where *all* the participants must be in the same state at the end of the protocol. A similar approach of the former definition of fairness can be found in [9,10].

We define a passive conspirator who takes part in a coalition excluding certain participants from the exchange as someone who cannot prevent this coalition from being done and who by his idleness contributes to keep the excluded participants in ignorance of the exchange which takes place.

Even if the fairness property is respected, in the coalition described above, P_1 to P_{i-1} become passive conspirators. They must trust P_0 for not answering with a false *yes* to the TTP during the recovery protocol.

Otherwise, even if P_0 decides to send a false *abort* to the TTP during the recovery protocol, fairness is still preserved. Once the first round of the main protocol has been successfully completed, P_0 sends m_0 to P_1 and participants P_1 to P_{i-1} do similarly. If P_i decides to not send m_i, after a certain amount of time P_{i+1} will realize that he must run the recovery protocol in order to obtain m_i. However, if P_0 sends a false *abort* to the TTP, the protocol will be terminated without P_{i+1} receiving m_i. If the TTP behaves properly, P_0 will be the only participant in a non-fair state at the end of the protocol.

It must be pointed out that every participant has to not only trust the TTP for behaving properly, but must also trust P_0 for not sending a false *yes* to the TTP during the recovery protocol.

4 A Protocol with Reduced Trust Requirements

We now present a variant of the multi-party fair exchange protocol described in the previous section. In this protocol an offline trusted third party is also used. The exchange topology is a ring and the communication channels between the participants and the TTP are supposed to be resilient (data is delivered after a finite, but unknown, amount of time).

Through this section we will use the following notations:

- $\mathcal{P}$ is the set $\{P_0, P_1, ..., P_{n-1}\}$ of all the participants in the exchange.
- $A \Rightarrow \beta$: denotes participant A multicasting a message to the set of participants β.
- f_x is a flag indicating the purpose of a message in a given protocol; x is composed of a letter and a number corresponding respectively to the protocol and the message number in this protocol.

– *label* is an information identifying a protocol run, that depends, among others, on $\mathcal{P}$.
– $S_A(m)$ denotes the digital signature of participant A over the message m.
– in a protocol message, $S_A(\star)$ denotes the digital signature of A over all information preceding this signature.
– m'_i denotes the concatenation of P_i's identity and the message m_i expected by P_{i+1}. $h(m'_i)$ is supposed to be public. m'_i and $c_i = E_e(m'_i)$ are used to generate the certificate $cert_i$.

Let P_0 be the participant that initiates the protocol. We suppose that this is known by all the participants and the TTP. Moreover, the set $\mathcal{P}$ is supposed to be known by all the participants in the exchange.

4.1 Main Protocol

1. $P_i \rightarrow P_{i+1} : f_{m1}, P_{i+1}, label, c_i, cert_i, S_{P_i}(\star)$ for $i = 0, ..., n-1$.

P_0 begins the main protocol by sending to P_1 the cipher c_0 of the message m'_0, along with a certificate $cert_0$ proving that c_0 is indeed the cipher of m'_0.

After receiving $(c_0, cert_0)$ from P_0, if $cert_0$ is valid, P_1 enciphers m'_1 and sends $(c_1, cert_1)$ to P_2. For $i = 2, 3, ..., n-1$, every P_i does similarly.

2. $P_0 \Rightarrow \mathcal{P} \setminus P_0 : S_{P_0}(label)$.
   $P_0 \rightarrow P_1 : \quad f_{m2}, P_1, label, m_0, S_{P_0}(\star)$.

Upon P_0 receiving $(c_{n-1}, cert_{n-1})$, if the certificate sent by P_{n-1} is valid, P_0 multicasts his signature over the *label*, $S_{P_0}(label)$, to the set $\mathcal{P} \setminus P_0$ of participants, and sends the message m_0 to P_1.

3. $\left.\begin{array}{l} P_i \Rightarrow \mathcal{P} \setminus P_i : S_{P_0}(label). \\ P_i \rightarrow P_{i+1} : \quad f_{m2}, P_{i+1}, label, m_i, S_{P_i}(\star). \end{array}\right\}$ for $i = 1, ..., n-1$.

When P_i $(1 \leq i \leq n-1)$ receives a valid $S_{P_0}(label)$ for the first time, he multicasts this signature to $\mathcal{P} \setminus P_i$. Upon receiving such a signature and m_{i-1} from P_{i-1}, P_i sends m_i to P_{i+1}.

4.2 Recovery Protocol

If some P_i does not receive m_{i-1} during the main protocol, he has to run the recovery protocol.

1. $P_i \rightarrow TTP : \quad f_{r1}, TTP, label, h(m'_{i-1}), S_{P_0}(label), c_{i-1}, cert_{i-1}, S_{P_i}(\star)$.
2. $TTP \rightarrow P_{i-1} : S_{P_0}(label)$.
3. $TTP \rightarrow P_i : \quad f_{r2}, P_i, label, m_{i-1}, S_{TTP}(\star)$.

In order to get c_{i-1} deciphered by the TTP, P_i sends to the latter ($S_{P_0}(label)$, c_{i-1}, $cert_{i-1}$). If $S_{P_0}(label)$ is a valid signature of P_0 over the *label* and if the certificate $cert_{i-1}$ is correct, the TTP deciphers c_{i-1} and obtains m'_{i-1}, the

concatenation of P_{i-1}'s identity and m_{i-1}. He forwards $S_{P_0}(label)$ to P_{i-1} and sends m_{i-1} to P_i.

4.3 Analysis

Before that P_0 multicasts $S_{P_0}(label)$ to $\mathcal{P} \setminus P_0$, no participant belonging to this set is able to run the recovery protocol. If the TTP behaves as described, none of them will get c_{i-1} deciphered.

P_0 could do a recovery instead of multicasting his signature over the *label*. In this case, the TTP will forward that signature to P_{n-1} and the exchange will be able to continue.

If during the main protocol some P_i does not receive the expected m_{i-1}, he can run the recovery protocol only if he provides $S_{P_0}(label)$ to the TTP. As no particular assumption has been made over the communication channels between participants, P_i could not receive the signature of P_0 over the *label*. Even in this case, P_i would remain in a fair state: if P_{i+1} contacts the TTP, this one will send $S_{P_0}(label)$ to P_i in the second step of the recovery protocol and P_i will also be able to run this protocol.

Following the definition given by Bao et al. [5], the protocol presented above is fair: at the end there will be no honest participant in a state where his c_i has been deciphered without having him received m_{i-1}.

About Trust and Passive Conspiracies In the main protocol, m_j ($1 \leq j \leq n-1$) is sent as soon as a valid $S_{P_0}(label)$ has been received. Otherwise, a coalition between, for example, P_0 and a participant P_i, excluding participants P_{i+1} to P_{n-1} from the exchange without the consent of participants P_1 to P_{i-1}, could exist: P_0 decides to not multicast his signature and P_i stops the exchange after receiving m_{i-1}, while P_1 to P_{i-1} are unable to inform the excluded participants that the last round of the main protocol has begun. P_1 to P_{i-1} would become passive conspirators; this situation is avoided by sending m_j after having received a valid $S_{P_0}(label)$.

During the main protocol $S_{P_0}(label)$ is multicasted to all the participants in the exchange. Suppose that some P_i refuses to realize this step of the protocol. If there exists a coalition (not including P_i) willing to exclude some participants from the exchange, P_i will not send $S_{P_0}(label)$ to the excluded participants and will become a passive conspirator. Multicasting $S_{P_0}(label)$ to *all* the others participants in the exchange prevents such a situation.

Passive conspiracies in order to exclude participants can be avoided. Therefore, the participants in the exchange do not longer need to trust P_0.

Complaint Protocol As described above, if a participant having received the cipher and the corresponding certificate during the first round of the main pro-

tocol receives $S_{P_0}(label)$, it will be possible for him to run the recovery protocol. Otherwise, if he receives $S_{P_0}(label)$ without having received the cipher and the corresponding certificate, he will be able to prove to an external party, by executing the following *complaint protocol* with the TTP, that something went wrong during the exchange.

1. $P_i \rightarrow TTP: \quad f_{c1}, TTP, label, \mathcal{P}, S_{P_0}(label), S_{P_i}(\star)$.
2. $TTP \Rightarrow \mathcal{P} \setminus P_i : f_{c2}, \mathcal{P}, label, S_{P_0}(label), S_{TTP}(\star)$.

A participant P_i $(1 \leq j \leq n-1)$ begins that protocol by sending ($label$, $\mathcal{P}$, $S_{P_0}(label)$) to the TTP. The latter checks if $\mathcal{P}$ is consistent with the $label$, if P_i belongs to the set $\mathcal{P}$ and if $S_{P_0}(label)$ is a valid signature. If so, the TTP multicasts the signature of P_0 over the $label$ to the remaining participants and asks them for the signatures obtained during the first round of the main protocol.

At this moment all the participants have received $S_{P_0}(label)$, and other excluded participants (having possibly not received this signature before) can also invoke the complaint protocol.

3. For $j = 0, ..., i-1, i+1, ..., n-1$:
   $P_j \rightarrow TTP : f_{c3}, TTP, f_{m1}, label, c_{j-1}, cert_{j-1},$
   $S_{P_{j-1}}(f_{m1}, P_j, label, c_{j-1}, cert_{j-1}), S_{P_j}(\star)$.
4. $TTP \rightarrow P_i : f_{c4}, P_i, label, \mathcal{P}, S_{TTP}(\star)$.

If after a deadline chosen by the TTP none of the remaining participants is able to present P_i's first round signature, the TTP will issue an affidavit attesting that something wrong happened during the protocol. Two cases are possible: either an honest entity was excluded from the exchange or a dishonest entity ran the complaint protocol with the help of the next participant in the ring.

$\mathcal{P}$ has been defined as a set of participants. If all the participants in $\mathcal{P} \setminus P_i$ reply to the second message of the complaint protocol, the TTP will know their disposition in the ring. However, the TTP will not be able to determine where the excluded participant P_i should be.

Otherwise, if $\mathcal{P}$ is defined as an ordered set according to the agreed topology, it will be possible for the TTP to determine the identity of the nearest participant, actively involved in the exchange, who follows P_i. At least this participant has contributed to the coalition. The TTP may not be able to identify the dishonest participant who precedes P_i because this dishonest entity may also realize a complaint protocol.

5 Conclusion

We have shown that every participant in the fair exchange protocol proposed by Bao et al. [5] must not only trust the TTP, but has to also trust P_0, the participant that initiates the protocol, for not sending a false *yes* to the TTP during the recovery protocol. Such a behavior from P_0 can lead to have a set

of entities to participate, without their consent, in a coalition excluding the remaining participants. (Bao et al. also proposed [5] two modified versions of their protocol. Participants in these modified protocols have to also trust the initiator of the exchange.)

We have presented a new protocol in which participants must only trust the TTP, and where passive conspiracies in order to exclude a set of participants can be avoided. Trust requirements are reduced by increasing communication needs. It is not easy to compare this reduction of trust requirements with the resulting increase of communications. However, this communication increase is measurable, unlike trust aspects.

Moreover, the proposed protocol allows excluded honest participants having received $S_{P_0}(label)$ to prove to an external adjudicator that something went wrong during the protocol.

References

1. Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996.
2. N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for multi-party fair exchange. Research Report RZ 2892 (# 90840), IBM Research, December 1996.
3. N. Asokan and Victor Shoup. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the 1998 Security and Privacy Symposium*. IEEE, 1998.
4. N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. Research Report RZ 2973 (#93019), IBM Research, November 1997.
5. Feng Bao, Robert Deng, Kanh Quoc Nguyen, and Vijay Vardharajan. Multi-party fair exchange with an off-line trusted neutral party. In *DEXA'99 Workshop on Electronic Commerce and Security*, Firenze, Italy, September 1999.
6. Feng Bao, Robert H. Deng, and Wenbo Mao. Efficient and practical fair exchange protocols with off-line TTP. In *RSP: 19th IEEE Computer Society Symposium on Research in Security and Privacy*, Washington - Brussels - Tokyo, May 1998.
7. Matt Franklin and Gene Tsudik. Secure group barter: Multi-party fair exchange with semi-trusted neutral parties. *Lecture Notes in Computer Science*, 1465, 1998.
8. Matthew K. Franklin and Michael K. Reiter. Fair exchange with a semi-trusted third party. In *4th ACM Conference on Computer and Communications Security*, Zurich, Switzerland, April 1997.
9. Steve Kremer and Olivier Markowitch. A multi-party non-repudiation protocol. In *Proceedings of SEC2000 conference*, Beijing, China, August 2000.
10. Olivier Markowitch and Steve Kremer. A multi-party optimistic non-repudiation protocol. Technical Report 443, ULB, 2001. Published in the proceedings of The 3rd International Conference on Information Security and Cryptology (ICISC 2000).
11. Olivier Markowitch and Shahrokh Saeednia. Optimistic fair-exchange with transparent signature recovery. Technical Report 452, ULB, 2001. Published in the proceedings of the 5th International Conference: Financial Cryptography 2001 (FC01).
12. Jianying Zhou and Dieter Gollmann. An efficient non-repudiation protocol. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.