An Improved Protocol for Optimistic Multi-party Fair Exchange

Yi Liu[1,2,a] ,Hongli Hu[3,b]
1.Faculty of Computer
Guangdong University of Technology
Guangzhou, 510006, China
2.State Key Laboratory of Software Engineering
Wuhan University
Wuhan, 430072,China
3.Guangzhou GRG Banking Equipment Co.,Ltd.
Guangzhou, 510663, China
a.E-mail:liuyi_xd@126.com;b.E-mail:crystal_hu@126.com

***Abstract*—An exchange protocol is a protocol that allows two or more parties to exchange electronic information. Bao et al. proposed a multi-party fair exchange protocol based on an offline TTP. Participants in this protocol have to trust the initiator of the exchange in addition of the TTP. In this paper we present an improved optimistic multi-party fair exchange protocol in which participants do not have to trust anyone except for the TTP. Moreover, the communication overhead of our protocol is not increased.**

Keywords- optimistic; multi-party; fair exchange protocol

I. INTRODUCTION

An exchange protocol is a protocol that allows two or more parties to exchange electronic information. It is said to be fair if the exchange is realized in such a way that, at the end of the protocol, any honest participant has received all the expected items corresponding to the items that he has provided; or he has received nothing without losing anything. Fair exchange protocols often use a trusted third party (TTP) helping the participants to successfully realize the exchange.

Depending on its level of involvement in a protocol, a TTP can be said inline, online or offline [4]. 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. Much more efficient are optimistic protocols in which the offline TTP only has to participate if a conflict has to be resolved.

In [1] a generic optimistic protocol with a general topology (each entity can communicate with the set of entities of his choice) is described. Practically, many proposed multi-party fair exchange protocols [2,3,4,5] suppose that each party exchanges an item against another and that the exchange's topology is a ring, where each participant Pi offers to participant Pi_{+1} message mi in exchange of message mi_{-1} offered by participant Pi_{-1}. Of course, all subscripts are mod n, where n is the number of participants in the exchange. We will omit this hereafter.

Based on an offline TTP, a multi-party fair exchange protocol is presented in [2]. Participants in this protocol described in [2] have to trust the initiator of the exchange in addition of the TTP. A modified version of this protocol, in which participants do not longer need to trust that initiator, was proposed in [5]. While the protocol in [5] requires that some information is broadcasted many times, the communication overhead is increased significantly.

In this paper we propose an improved optimistic multi-party fair exchange protocol in which participants do not have to trust that initiator. Moreover, the communication overhead of our protocol is the same as the protocol in [2].

For simplicity, we assume that the communication channels between each participant and the TTP are confidential or messages sent are encrypted with the public key of the receiver and that a unique transaction identifier should be incorporated in every message to prevent replay attacks. Nevertheless, we omit these in our protocol description in order to ease presentation. We also assume that all channels are resilient (the messages inserted into the channel are delivered within a finite, but unknown, amount of time).

The remaining of this paper is organized as follows. Section 2 describes the protocol proposed by Bao et al. in [2] and the analysis on it. In section 3 we present our improved optimistic multi-party fair exchange protocol. Finally, we conclude this paper in Section 4.

II. A MULTI-PARTY FAIR EXCHANGE PROTOCOL WITH AN OFFLINE *TTP*

In this section we briefly describe the optimistic multi-party fair exchange protocol proposed by Bao et al. [2]. The exchange topology is a ring.

This exchange protocol consists of a setup phase and an exchange phase. During the setup phase, entities willing to participate in a given exchange's execution agree on the set of entities who will take part in this exchange, on the items to be exchanged and on how these items will be exchanged during the exchange phase. After completion of that setup phase, the exchange phase is performed by the means of a fair exchange protocol.

Let P_0 be the participant that initiates the protocol and the status of P0 is known by the TTP. Let E and D be the encryption and the corresponding decryption algorithms of a public-key cryptosystem. The TTP owns a public key e and a

secret decryption key d of this cryptosystem. Let $h(\cdot)$ be a public one-way hash function.

Each participant P_i enciphers mi to $c_i = E_e(m_i)$, generates a certificate $cert_i = certify(m,c,e)$ using a public algorithm $certify(\,)$ (details in [6]). The $cert_i$ can be verified by using a public algorithm $vertify(\,)$ such that $verify(c_i, cert_i, h(m_i), e) = yes$ if and only if $h(D_d(c_i)) = h(m_i)$. Then it is convinced that ci is indeed the cipher of mi under key e, $i \in [0, n-1]$.

There are two properties are satisfied [2]:

It is computationally unfeasible for P_i to generate a certificate $cert_i$ such that $verify\ (c_i, cert_i, h(m_i), e) = yes$, while $h(D_d(c_i)) \neq h(m_i)$;

It is computationally unfeasible to get mi from ci without knowing d.

At the end of the setup phase:

— each participant knows the identity of the remaining participants in the exchange;

— the participants in the exchange have agreed on the identity of the TTP that will be contacted during the protocol's execution and on the hash function h(.) that will be used;

— the descriptions of the messages to be exchanged, $h(m_i), i \in [0, n-1]$, are made public.

A. Main Protocol

P_0 begins the main protocol by sending to P1 the cipher c0 and cert0. After receiving (c_0, $cert_0$), P_1 checks $cert_0$; if it is valid, he enciphers m1 and sends (c_1, $cert_1$) to P_2. For $i = 2, \ldots, n-1$, participant P_i does similarly.

When P_0 receives (cn_{-1}, $certn_{-1}$), he checks the $certn_{-1}$; if it is valid, he sends m0 to P1. For $i = 1,2,\ldots,n-1$, after receiving mi_{-1} from P_{i-1}, Pi sends mi to P_{i+1}.

These are the two rounds of the main protocol:

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

B. Recovery Protocol

If all participants behave correctly, after the main protocol execution, each party should obtain his expected message. However, if some P_i does not receive his respected 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 TTP firstly checks the $cert_{i-1}$, if it is valid, the TTP does not ask P_0 if he has received a valid (c_{n-1}, $cert_{n-1}$) until he estimates P_0 has obtained it if no problem occurs. If P_0 answers yes, the TTP will decipher c_{i-1} to m_{i-1} and send m_{i-1} to P_i.

The TTP does not contact P_0 each time that some other party runs the recovery protocol. If P_0 answers yes in the first recovery execution then the TTP will accept further recovery request; otherwise the TTP will reply with an abort message.

These are the four steps of the recovery protocol. 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. $TTP \rightarrow P_i : m_{i-1} \ \ or \ \ abort$.

C. Analysis

Here we introduce the definition of fairness: an exchange protocol is called a fair exchange protocol if after the protocol execution no participant P_i following properly that he has received m_{i-1} without sending c_i.

In the protocol, a participant $P_j (1 \le j \le n-1)$ sends m_j to P_{j+1} only if he has received m_{j-1}. P_{j+1} can obtain the expected m_j only if P_0 received valid (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}. After the protocol execution (including 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.

But as pointed out by [2], P_0 must be honest. Otherwise, let's consider that P_0 colludes with P_i. When P_i receives (c_{i-1}, $cert_{i-1}$) from P_{i-1} in the first round of the main protocol, P_i run directly the recovery protocol without sending (c_i, $cert_i$) to P_{i+1}. The TTP asks then P_0 if he has received (c_{n-1}, $cert_{n-1}$), P_0 answers with a yes (in fact P_0 has not receive it), the TTP will decipher c_{i-1}. So, P_i can gain m_{i-1} without losing m_i.

III. An Improved Optimistic Multi-party Fair Exchange Protocol

Before presenting our protocol, we introduce some background about the theory of cross validation on which the protocol is based. For details the reader can refer to [3,4].

We present an improved optimistic multi-party fair exchange protocol in this section. The exchange topology is still a ring. In this protocol, we use a homomorphic one-way function f, as in [3], which satisfies f(a) · f(b) = f(ab). The authors propose that $f(y) = y^2 \bmod N$, where N is the product of two large distinct primes. Let E and D be the encryption and the corresponding decryption algorithms of a public-key cryptosystem. The TTP owns a public key e and a secret decryption key d of this cryptosystem. Let $h(\cdot)$ be a public one-way hash function.

Each participant enciphers mi to $c_i = E_e(m_i)$, generates a certificate $cert_i = certify(m,c,e)$ using a public algorithm $certify(\,)$. After that, Pi chooses a random value Ri and calculates f(Ri), $i \in [0, n-1]$.

At the end of the setup phase:

— each participant knows the identity of the remaining participants in the exchange;

— the participants in the exchange have agreed on the identity of the TTP that will be contacted during the protocol's execution and on the function $h(\cdot)$, $f(\cdot)$ that will be used;

—the descriptions of the messages to be exchanged, $h(m_i), f(R_i), i \in [0, n-1]$, are made public.

A. Main Protocol

P_0 begins the main protocol by sending to P_1 the cipher c_0, $cert_0$, R_0 and the signature of P_0 over them ($S_{P_i}(*)$ denotes the digital signature of P_i over all information preceding this signature). After receiving the information from P_0, P_1 checks $cert_0$; if it is valid, he checks if R_0 is the pre-image of $f(R_0)$ and verifies the signature $S_{P_0}(*)$. If all the checks pass, P_1 sends to P_2 the cipher c_1, $cert_1$, $(R_0 \cdot R_1)$ and the signature of P_1 over them. After receiving the information from P_1, P_2 checks cert1; if it is valid, he checks if $(R_0 \cdot R_1)$ is the pre-image of $f(R_0) \cdot f(R_1)$ (where function f satisfies $f(a) \cdot f(b) = f(ab)$) and verifies the signature $S_{P_1}(*)$. If all the checks pass, P_2 sends to P_3 the cipher c_2, $cert_2$, $(R_0 \cdot R_1 \cdot R_2)$ and the signature of P_2 over them. For $i = 3, \ldots, n-1$, participant Pi does similarly.

When P_0 receives the information sending by P_{n-1}, he firstly checks $cert_{n-1}$; if it is valid, he checks if $(R_0 \cdot R_1 \cdot \ldots \cdot R_{n-1})$ is the pre-image of $f(R_0) \cdot f(R_1) \cdot \ldots \cdot f(R_{n-1})$ and verifies the signature $S_{P_{n-1}}(*)$. If all are valid, he sends the message m_0 to P_1. For $i = 1, \ldots, n-1$, after receiving mi-1 from P_{i-1}, P_i sends mi to P_{i+1}.

These are the two rounds of the main protocol:

1. $P_i \rightarrow P_{i+1}$: $P_i, P_{i+1}, c_i, cert_i, (R_0 \cdot R_1 \cdot \ldots \cdot R_i), S_{P_i}(*)$ for $i = 0, \cdots, n-1$.
2. $P_i \rightarrow P_{i+1} : m_i$ for $i = 0, \cdots, n-1$.

B. Recovery Protocol

If all participants behave correctly, after the main protocol execution, each party should obtain his expected message. However, if some Pi does not receive his respected mi-1, he has to run the recovery protocol.

Pi begins this protocol by sending (ci-1, certi-1) to the TTP. The TTP firstly checks the certi-1. If it is valid, the TTP does not asks P0 if he has received $(R_0 \cdot R_1 \cdot \ldots \cdot R_{n-1})$ until he estimates P0 has obtained it if no problem occurs. If P0 answers yes and sends a valid $(R_0 \cdot R_1 \cdot \ldots \cdot R_{n-1})$ to the TTP, the TTP will decipher ci-1 to mi-1 and send mi-1 to Pi.

The TTP does not contact P0 each time that some other party runs the recovery protocol. If P0 answers yes in the first recovery execution then the TTP will accept further recovery request; otherwise the TTP will reply with an abort message.

These are the four steps of the recovery protocol, initiated by some Pi having not received mi-1. Steps 2 and 3 are only executed the first time that this protocol is invoked.

1. $P_i \rightarrow TTP : c_{i-1}, cert_i$.
2. $TTP \rightarrow P_0 : call$.
3. $P_0 \rightarrow TTP : (yes, R_0 \cdot R_1 \cdot \ldots \cdot R_{n-1})$ or $abort$.
4. $TTP \rightarrow P_i : m_{i-1}$ or $abort$.

C. Analysis

If the first round of the main protocol has not been completed:

Any participant Pi cannot get mi-1 from the TTP. Because the TTP will ask P0 for $(R_0 \cdot R_1 \cdot \ldots \cdot R_{n-1})$ when Pi runs the recovery protocol. However, whether P0 is honest or not, he cannot provide $(R_0 \cdot R_1 \cdot \ldots \cdot R_{n-1})$ for the TTP and the latter may not decipher ci-1.

If the first round of the main protocol has been completed:

In this scenario, each party Pi receives the valid (ci-1, certi-1). If some Pj does not receive his expected mj-1 in the second round of the main protocol, he has to run the recovery protocol. If P0 is honest, he will provide $(R_0 \cdot R_1 \cdot \ldots \cdot R_{n-1})$ for the TTP and finally Pj can receive mj-1 after the recovery protocol completes; If P0 misbehaves, he will not provide $(R_0 \cdot R_1 \cdot \ldots \cdot R_{n-1})$ for the TTP and send a abort to TTP. After that, the TTP will not decipher any ci. Though he Pj has not received mj-1, he has not lost cj. Our protocol is still fair.

IV. CONCLUSION

The exchange of valuable items like electronic payments and digital goods exposes the participating parties to the risk that the Internet connection is accidentally or even maliciously disrupted. This may lead to an unfair situation. To overcome it, fair exchange protocols have been proposed.

An optimistic multi-party fair exchange protocol proposed by [2] shows that every participant in the protocol trusts not only the TTP, but also the initiator of the protocol. In this paper, we present an improved fair exchange protocol in which participants must only trust the TTP and this protocol does not increase communication needed.

ACKNOWLEDGMENT

The work described in this paper is supported by State Key Laboratory of Software Engineering (SKLSE2010-08-17).

REFERENCES

[1] N. Asokan, M. Schunter, and M. Waidner, “Optimistic protocols for multi-party fair exchange,” Research Report RZ 2892 (# 90840), IBM Research, Dec. 1996.

[2] F. Bao, R. Deng, K. Q. Nguyen, and V. Vardharajan, “Multi-party fair exchange with an off-line trusted neutral party,” in DEXA’99 Workshop on Electronic Commerce and Security, 858-863, Sep. 1999, pp. 858-863.

[3] M. Franklin and G. Tsudik. Secure group barter, “Multi-party fair exchange with semi-trusted neutral parties,” in Proceedings of the International Conference on Financial Cryptography, LNCS 1465, Springer-Verlag, 1998, pp. 90-102.

[4] N. González-Deleito and O. Markowitch, “Exclusion-freeness in multi-party exchange protocols,” in 5th International Conference on Information Security, LNCS 2433, Springer-Verlag, 2002, pp. 200-209.

[5] N. González-Deleito and O. Markowitch, “An optimistic multi-party fair exchange protocol with reduced trust requirements,” in proceedings of the 4th International Conference on Information Security and Cryptology, LNCS 2288, Springer-Verlag, Dec. 2001, pp. 258-267.

[6] F. Bao, R. H. Deng, and W. Mao, “Efficient and practical fair exchange protocols with off-line TTP,” in Proceedings of the 19th IEEE Computer Society Symposium on Research in Security and Privacy, May 1998, pp. 77-85.