Multi-party Non-repudiation Protocol with Different Message Exchanged

Li Yanping  
College of Mathematics and Information Science  
Shaanxi Normal University  
Xi’an, China  
Email: lypdzp@163.com

Pang Liaojun  
State Key Laboratory of Integrated Service Networks  
Xidian University  
Xi’an, China  
Email: lj pang@mail.xidian.edu.cn

Abstract—Based on the hardness assumption of discrete logarithm problem and a group oriented encryption scheme, a new multi-party non-repudiation protocol is presented. The features of this protocol are the following: In the B2C or B2B E-commerce application, it is necessary for an organisation/people to deal with all kinds of requests, business documents, agreements, payments, contracts, acknowledgments as a batch. The protocol allows such organisations/people to send different messages to multiple different recipients, eliminates the previous restriction on the exchange of the same message. The protocol also utilizes an off-line third trusted party (TTP) for short and has higher efficiency than those protocols with an on-line TTP.

Keywords—Non-repudiation; Multicast; Resilient channel

I. INTRODUCTION

The non-repudiation problem is one of important security problems in E-commerce data exchange. Generally, non-repudiation of origin can easily be achieved by signing the sent information. But non-repudiation of receipt is more difficult to provide. Usually protocols providing both non-repudiation of origin and non-repudiation of receipt relies on a trusted third party (TTP) who has to intervene during each protocol run [1,2]. Such a TTP is said be on-line and may create a communication bottleneck. Therefore Kremer and Markowitch presented a protocol [3] based on the optimistic idea: he assumes that in general Alice and Bob are honest, i.e. they correctly follow the protocol, and that the TTP only intervenes, by the mean of a recovery protocol, when a problem arises. A TTP that dose not necessarily intervene in each protocol run is said be off-line. Obviously off-line TTP is more practical than on-line TTP.

Most of well-known non-repudiation protocols are two-party protocol[1,2,3]. With the development of B2C or B2B E-commerce, the multi-party non-repudiation protocols are also important. The known multi-party non-repudiation protocols have been presented in [4,5,6]. All these protocols have one simple sender, with many recipients, for exchange of the same message. But many E-commerce applications, organisations/people always wants to exchange resource requests, data, business documents, agreements, payments, contracts, acknowledgments, and so forth to many different recipients at one time. These exchanges can be abstracted as different message transfers among members (users or automated systems) of a virtual community.

Our aim is to design a multi-party non-repudiation protocol, where the sender can exchange for different messages with multiple recipients. Suppose the proposed protocol needs an off-line TTP, which has a open directory (DIR) in our proposed protocol.

Structure of the rest of paper is organized as follows. In section 2, we give a brief review of the preliminaries and properties required by non-repudiation protocols. In section 3, we present a novel optimistic multi-party non-repudiation protocol achieving strong fairness for exchanging different messages and give its security analysis. Conclusions are given in the last section.

II. PRELIMINARIES

A. The Communication Channels

In the messages exchange protocols, the communication channels are usually divided into three classes: unreliable channels, resilient channels, and operational channels [7]. No assumptions have to be made about unreliable channels: data may be lost. A resilient channel delivers data after a finite, but unknown amount of time. Data may be delay, but will eventually arrive. Operational channels are rather unrealistic in heterogenous networks for the data inserted in the operational channels must arrive after a known, constant amount of time.

B. Basic Requirements on Non-repudiation Protocol

A primary property non-repudiation protocol must respect is non-repudiation, which includes non-repudiation of receipt and non-repudiation of origin. The second property is fairness. Fairness comes in three flavors: weak fairness and strong fairness, whose security enhances gradually. A non-repudiation protocol respects fairness if and only if it can provide one of the above three fairness. The third property is timeliness: a protocol must be finished after a finite amount of time for each participant who is behaving correctly while preserving fairness. This property avoids situations where a party does not know whether he can stop the protocol or not without losing fairness [7]. It makes a protocol practicable. We will show that our protocol respects
non-reputation, strong fairness and timeliness in the following sections.

C. Group Encryption Scheme

There are many group oriented encryption schemes[8,9].The scheme presented in [8] is based on a public key encryption scheme and on the Chinese Remainder Theorem(CRT).The protocol is generic as it can use any secure public-key cryptosystem. Let n be the number of recipients and A be the message m sender. We shall denote the public-key encryption operation as $PE_R(.)$ and the decryption operation as $PD_R(.)$ for participant R. The public values for each recipient are his public key and a large random integer $N_i$.All $N_i$ are required pairwise relatively prime $N_i > PE_R(m)$. A computes $X \equiv PE_R(m)(modN_i)\ (i = 1,...,n)$.As all $N_i$ are pairwise relatively prime,using the CRT, an unique solution is obtained from these equations. Once receiving the solution $E_R(m)$,each recipient $R_i \in R$ can read $m$ by computing $E_R(m)(mod N_i) \equiv PE_R$ and $PD_R(PE_R(m)) = m$. The scheme presented in [9] easily apply to our protocol with small changes.

III. THE PROPOSED NON-REPUTATION PROTOCOLS

This protocol is composed of three sub-protocols:Main protocol,Recovery protocol and Abort protocol. We suppose each transmission is associated to some maximum time-out,which is decided according to networks condition and the business urgency.Once time exceeds the time-out, the recipient considers that the transmission will not arrive any more and initiates either a recovery or an abort protocol. We also suppose the channel between the TTP and each participants is resilient and the channel between the sender and the receivers maybe unreliable. We first give some notations which are used to describe our protocol.

A. Definitions and Notations

$R = (R_1,R_2, ..., R_n)$: set of intended recipients.The symbol $R_i$ can be $R_i$’s unique ID:
$R' \triangleright R$: the set of recipients having sent the sender a valid signature evidences of receipt,$R' \subseteq R$;
$A \Rightarrow R$: multicast messages from A to the set of R.
$E_R(m)$:denotes the result of group oriented encryption,Only entity $R_i \in R$ set can decrypt $m$ when receive $E_R(m)$.
$A \rightarrow B$: transmission from entity A to entity B;
$M_i$: the message which the sender A will send to $R_i$;
$|M| = |M_i|$, $|x|$ denotes the binary length of x;
$c_i = E_k(M_i)$: the cipher resulting from symmetric-key encryption of $M_i$;
$c = PE_T(M)$:the cipher resulting from TTP public-key encryption of M;
$t$: a time-out chosen by the sender,after which a Recoveryprotocol may be performed;
$H(.) : \{0,1\}^* \rightarrow Z_q^*$ and $H_i(.) : \{0,1\}^* \rightarrow \{0,1\}^{128}$ are one-way hash function.

System chooses a big prime p,q such that $p-1$ have a big prime factor q,p,q. The public parameters are \{p,q,g, H,H_1,E_k() \}. The sender A randomly chooses $x \in Z_q^*$ and computes $k_i = H_1(y_i^{x}modp), i = 1,2, ..., n,k_T = H_1(y_T^{x}modp), c = PE_T(M), m = H(M), r = H(g^{c}modp,c,m), r_i = H(g^{c_i}modp,c_i,M_i)s = x/(y_Tr_1r_2...r_n + xA)modp$. The sender A takes $l_i = H(M_i,M,c_i)$ as label of message $M_i$.

B. Main Protocol

(1) $A \Rightarrow R : MSG_1 = A,TTP,t,M,c,r,s,\{R_1,l_1,r_1, c_1\}, i = 1,2, ..., n$;
(2) $R_i \rightarrow A : Sig_{R_i}(MSG_2), MSG_2 = \{R_i,TTP,A,l_i, \bar{m}_i\}$.If A time out then abort;
(3) $A \Rightarrow R' : Sig_A(\bar{M}_3), \bar{M}_3 = \{A,TTP, E_R(\bar{M})\}$. If $R_i$ time out after (after $t$) then recovery;
(4) $R_i \rightarrow A : Sig_{R_i}(MSG_4), MSG_4 = \{R_i',TTP,A,l_i, m_i\}$. If $A$ time out (after $t$) then recovery.

The non-reputation evidence of origin is $NRO = Sig_{R_i}(MSG_2) \cup Sig_{R_i}(MSG_4)(or Sig_{TTP}(MSG_6))$. The non-reputation evidence of receiver is $NRR = MSG_1 \cup Sig_A(\bar{M}_3)(or Sig_{TTP}(MSG_6))$.

step1: Each recipient in R gets the cipher $c_i$ , the committed evidence $MSG_1$ and the key message M ciphered by TTP’s public-key (this information is used by the TTP in case of a recovery).Any recipient who does not accept the time-out t can fairly quit the protocol by replying no message to A. Message label $l_i$ also can be used to check the correctness of the message $M_i$,M and the cipher $c_i$ .If $R_i$ wants to continue the protocol,he computes $y = (yo^{g^{r_1}...r_n})^{x}modp, k_i = H_1(y_i^{x}modp), M_i = D_k(c_i)$ and verifies $r_i = H(y,c_i,M_i)$.If verification is true,$R_i$ sends $MSG_2$ to A.

step2: After having sent $MSG_1$,A decides of a future moment by an time-estimated model. All committed receipts $MSG_2,Sig_{R_i}(MSG_2)$ arriving after the moment are not considered any more. Then A sends M to those recipients who reply valid committed receipts.

step3: Only those entities belonging to $R' \triangleright R$ will be able to decrypt and extract $M$ by compute $M_i = M \bigoplus M_i$.They also verify the correction of $M_i$ and $M$ by computing $l_i = H(M_i,M,c_i)$.If some $R_i$ do not receive $MSG_3$ and $Sig_{R_i}(MSG_3)$ ,they can launch Recovery protocol.

C. Recovery Protocol

At any moment during the protocol A and $X \in R$ have the possibility to launch a Recovery protocol with the offline TTP. The Recovery protocol as follow.

1.$R_1 \rightarrow TTP : MSG_1, MSG_2, Sig_{R_i}(MSG_2)$;
Recovery
No matter which needs the initiator provides the valid sufficient evidences.
can assure the fairness between
A
not get any advantages, as the TTP sends
R
Abort
by running an
D. Abort Protocol
The key is to prevent some malicious eavesdroppers to intercept
key is ciphered under the
is TTP’s valid signature on
\bar{y}, \bar{c}, m\}. Else recovered = true.

TTT \rightarrow R_i ∪ A: SigTTT(MSG_6), MSG_6 = A, TTT, R_i, l_i, PE_{R_i}(M);

When a Recovery protocol is initiated after \( t \), the TTP needs the initiator provides the valid sufficient evidences. No matter which Recovery protocol is performed, the TTP can assure the fairness between \( A \) and \( R_i \). Even a malicious \( R_i \) successfully execute a Recovery protocol, he still can not get any advantages, as the TTP sends \( \text{Sig}_{TTT}(MSG_6) \) to \( A \). That the TTP verifies \( m_i = H(M), r_i = H(y, c_i, M_i) \) can prevent the sender \( A \) from replacing the encrypting \( M \) to cheat the recipients. It also avoids disputes which may arise at the end of the protocol about the \( M \) (such problem exists in [10]). That \( M \) is ciphered under the TTP’s or \( R_i \) public key is to prevent some malicious eavesdroppers to intercept the message.

D. Abort Protocol
Since any recipient can quit the protocol by replying no message to \( A \)’s first massage, \( A \) also can stop the protocol by running an Abort protocol. If neither a Recovery nor an Abort protocol has yet been initiated, the TTP accepts \( A \)’s abort request.

1. \( A \rightarrow TTT: A, R', L, \text{Abort}; R' ⊆ R, L = l_i' \);

If a Recovery has been performed successfully, then TPT \( \rightarrow A: MSG_6, \text{Sig}_{TTT}(MSG_6) \), else aborted = true;

TTT publishes \( \text{Sig}_{TTT}(MSG_7), MSG_7 = A, TTT, R'', L, \text{Abort} \), on his open DIR;

E. Disputes Resolved
At the end of the protocol execution, each recipient \( R_i' \in R' \) and \( A \) receive the non-repudiation evidences. And the non-repudiation of origin for message \( M_i \) are: \( NRR = \text{Sig}_{R_i}(MSG_3) ∪ \text{Sig}_{G}(MSG_4) \) or \( \text{Sig}_{RR}(MSG_6) \). The non-repudiation of receiver \( R_i \) are \( NRR = MSG_1 ∪ \text{Sig}_{A}(MSG_3) \) or \( \text{Sig}_{TTT}(MSG_6) \). Two kinds of repudiation disputes may arises as follows.

1) Repudiation of Origin: When \( A \) denies the origin of the message \( M_i \), \( R_i \) presents to a judge: \( MSG_1, MSG_3, M, \bar{M}_i, \text{Sig}_{G}(MSG_3) \) or \( MSG_1, MSG_6, M, \bar{M}_i, \text{Sig}_{TTT}(MSG_6) \). The judge
a. Computes \( y = (y_0 g^{t_1} \cdots t_n)^{modp}; \)

b. Verifies \( r_i = H(y, c_i, \bar{M}_i), m = H(M), r' = H(y, c, M_i), l_i = H(M, M, c_i), M_i = M \bigoplus \bar{M}_i; \)

c. If all yes, he verifies \( MSG_1 \) and \( \text{Sig}_{G}(MSG_3) \) is valid signature on the key message \( \{O, TTT, t, PE_T(M), M, \bar{M}_i, l_i, M_i, r_i \} \) and \( \text{Sig}_{TTT}(MSG_6) \) is TTP’s valid signature on \( MSG_6 \).

If \( R_i \) provides the required information and all the checks hold, the judge accepts the claim that \( A \) is the origin of the message \( M_i \).

2) Repudiation of Receipt: When \( R_i' \in R' \) denies having received the message \( M_i \), \( A \) presents to the judge: \( MSG_1, MSG_4, M, \bar{M}_i, \text{Sig}_{RR}(MSG_2), \text{Sig}_{RR}(MSG_4) \) or \( MSG_1, MSG_6, M, \bar{M}_i, \text{Sig}_{RR}(MSG_2), \text{Sig}_{TTT}(MSG_6) \). The presented \( M \) may be distinct from which randomly chosen by \( A \) at the beginning of the protocol. The judge verifies:

a. Computes \( y = (y_0 g^{t_1} \cdots t_n)^{modp}; \)

b. Verifies \( r_i = H(y, c_i, \bar{M}_i), m = H(M), r' = H(y, c, M_i), l_i = H(M, M, c_i), M_i = M \bigoplus \bar{M}_i; \)

c. If all yes, he verifies \( \text{Sig}_{RR}(MSG_2) \) and \( \text{Sig}_{RR}(MSG_4) \) is \( R_i \) valid signature and \( \text{Sig}_{TTT}(MSG_6) \) is TTP’s valid signature on \( MSG_6 \).

If \( A \) provides all needed information and all the checks hold, the judge concludes that \( R_i' \) received \( M_i \).

F. Fairness, Timeliness and Confidentiality
We start showing our protocol provides fairness. If some recipients stop the protocol after having received the first message, \( A \) can run the Abort protocol to prevent \( R_i \) to initiates a Recovery protocol later. Even some malicious \( R_i \) initiates a Recovery protocol successfully, he still can not gain any advantages. So the protocol remains fair.

If \( A \) stop the protocol after having received the second message, the recipients who reply to \( A \) valid receipts can initiates a Recovery protocol. If \( A \) has aborted the protocol, the protocol also ends in a fair way with no valid evidences exchanged. If the protocol has not been aborted, a Recovery protocol which is successfully performed assures that the fairness between \( A \) and \( R_i \).

If \( R_i \) stops the protocol after having received the third message or \( A \) receives an invalid evidence in the fourth message, they can launch a Recovery protocol to keep the protocol fair. Due to the resilience of the channels between the TTP and \( A ∪ R_i \), all data sent by the TTP to \( R_i \),
eventually arrive. In those cases all entities received expected messages and valid evidences. If the protocol is finished normally, each participant gets what he expects. Even in case of problems, one can initiate a Recovery protocol in time to force the protocol finished successfully and gets the messages he expects. The protocol finished in a fair way and the protocol provides strong fairness[7].

Now we shall show that timeliness is still respected. A can either finish the main protocol or initiate a recovery or an abort protocol. Any recipient R_i can quit the protocol by replying no message to A’s first massage, and either finish the main protocol or launch a recovery protocol. If R_i successfully initiates the recovery protocol, the TTP will send him and A the messages they respectively expect. As the channels between the TTP and R_i are resilient, the protocol finishes in a finite time preserving timeliness.

Now we shall show that confidentiality of transferred message is still respected. Whether a non-repudiation protocol must ensure confidentiality of transferred message or not is a controversy for a long time[10]. Recently[11] points out this should be decided according to the specific applications in E-commerce, and confidentiality is a basic requirement in most E-commerce applications. In our protocol, the confidentiality of transferred message is respected if private and public encryption algorithms are used. We suggest the M_i take the necessary measures to prevent duplicated and collude attacks[9].

G. Efficiency

Our aim is to design a non-repudiation protocol, which allows organisations/people to deal with all kinds of requests, business documents, agreements, payments, contracts, acknowledgments as a batch. With the development of B2C or B2B e-commerce, it is necessary for a merchant to deal with them at one time. Obviously, our protocol can realize this goal and can save the operation numbers and the sender’s time. So our approach has higher efficiency than that of executing n-instance of a two-party protocol.

IV. Conclusion

In this paper, we first introduce the state-of-art of two-party and multi-party non-repudiation protocols, and analyze the previous works and point out their disadvantages. The aim of this paper is to design a multi-party non-repudiation protocol, which allows the sender exchanges for different messages with multiple recipients for non-repudiation evidences. In our protocol, each participant can get the messages and evidences what he expects if and only if the protocol is finished. From security analysis, the protocol always keep fair even if the protocol is interrupted abnormally and the proposed protocol can provide strong fairness. The protocol utilizes an off-line TTP and has higher efficiency than those protocols with an on-line TTP. Therefore the protocol has great potential value in the future E-commerce applications.