A Non-Repudiation Message Transfer Protocol for E-commerce

Seokwon Yang, Stanley Y. W. Su, and Herman Lam
Database Systems R&D Center, University of Florida, Gainesville, Florida 32611
{seyang,su,hlam}@cise.ufl.edu

Abstract

In the business world, exchange of signatures or receipts is a common practice in case of future dispute. Likewise, it is critical in E-commerce applications to have the security service that generates, distributes, validates, and maintains the evidence of an electronic transaction. Quite of number of non-repudiation protocols have been proposed in distributed systems and evaluated based on some evaluation criteria. However, in the context of e-commerce, there are additional evaluation criteria to be considered: Fairness to both the message sender and the message receiver with respective to their control over the completion of a transaction, the degree of trust on a third party, and existence dependency on a third-party for dispute settlement on a committed transaction. In this paper, we identify the set of requirements for a message transfer protocol in E-commerce, and propose a new non-repudiation message transfer protocol that meets these additional criteria. Our protocol protects the confidentiality of message contents such that no unauthorized intermediary is able to see the contents. And, the protocol is superior to other protocols in that continuous existence of the third-party authority is not needed beyond the completion of a message transfer. Furthermore, with respect to the control over the commitment of a transaction, our protocol is fair to both the message sender and the receiver.

1. Introduction

In B2C or B2B e-commerce, organizations/people exchange resource requests, data, business documents, agreements, payments, contracts, acknowledgements, etc. These exchanges can be abstracted as message transfers among members (users or automated systems) of a virtual community. Non-repudiation in message transfers is a key security issue. A sender or a receiver should not be able to deny that a message has been sent or received if the message transfer actually took place. Non-repudiation is a security service, which creates, collects, validates, and maintains cryptographic evidence of an electronic transaction to support the settlement of a possible dispute [1].

Quite a number of non-repudiation protocols have been proposed in the literature and some criteria for evaluating these types of protocols have been proposed [1, 2, 3, 4, 5, 6]. Generally, these protocols encrypt the message with a secret key and deliver them (key and cipher-text) through different network routes (or orders) to satisfy non-repudiation from both the sender and the recipient. In his book, Zhou [1] compares the merits and weaknesses of eleven non-repudiation protocols qualitatively, in terms of the third party involvement (e.g., inline, online, or offline), communication overhead (high, medium, or low), privacy protection (good, average, or poor), and timely termination (yes, possible, or no). In the context of e-applications (for instance, internet-based e-check systems), additional evaluation criteria need to be considered when choosing the messaging protocol. These are:

- **Fairness:** Depending on who could control the execution of a messaging protocol, the protocol can be biased to either the sender or the receiver, or could be fair to both. For example, in order to protect a message sender from the receiver's repudiation of the receipt, a protocol can be designed in such a way that the message sender can control the commitment of the messaging protocol by not releasing the encryption key until he/she gets a receipt from the receiver. Such a protocol is in favor of the sender. Conversely, a protocol can be designed in such a way that task of releasing a encryption key is entrusted to a third party and no longer under control of the sender after sending a message, which makes it fair to the receiver. The protocol can be further refined so that the third-party releases the encryption key when it collects evidence of the recipient's non-repudiation. As for the case of e-check systems, we need a protocol that is in favor of the check receiver so that he can clear the check as soon as he receives.
- **Trust dependency on a third party:** Different messaging protocols can exhibit different degrees of trust dependency on a third party authority (TPA). For example, a protocol may allow a TPA to have a key to an encrypted message and the message itself, thus trusting the TPA with the contents of the message (i.e., a high degree of trust dependency). Another protocol may use a TPA's service to accomplish the message transfer, but does not allow the TPA to see the message contents. Such a protocol can be said to

Proceedings of the IEEE International Conference on E-Commerce (CEC'03)
0-7695-1969-5/03 $17.00 © 2003 IEEE

have a lesser degree dependency on the TPA, which is preferable for e-check systems because of confidentiality of transactions.

- **Existence dependency:** A protocol may depend on the existence of a TPA to settle a dispute long after the message transfer has been completed. Another protocol may produce enough digital evidence for both the sender and the receiver so that a subsequent dispute settlement does not depend on the existence or availability of the TPA.

If we take the above three evaluation criteria into consideration, we may find that some existing protocols do not satisfy these criteria. For example, the protocol proposed by Zhou [2] is not fair to the message receiver, because it requires that the receiver keeps on pulling for the encryption key to decrypt a received message. Also, the protocol has a high degree of trust dependency on the third party because the third party is entrusted with the encryption key. It can potentially use it to decrypt the sensitive information transmitted in a message. Furthermore, the presence of the third party is required for dispute settlement even long after the transaction has been committed. Ideally, at the end of a protocol, each party involved in a transaction should have a receipt from each other instead of a delivery signature of a third party whose business may no longer exist at the time of dispute resolution. A non-repudiation protocol that is fair to both parties, requires a lesser degree of trust dependency on the part of the third party, and does not rely on the existence of the third party to settle disputes, would be a better protocol. In this work, we developed such a non-repudiation protocol.

The remainder of the paper is organized as follows. After we survey some related works in Section 2, we describe the non-repudiation protocol requirements in Section 3, and present our approach to non-repudiation for collaborative E-commerce in Section 4. An informal analysis and discussion on the protocol is given in Section 5. Finally, a summary and some concluding remarks are given in Section 6.

2. Related work

The existing studies on handling digital signature or evidence in electronic transactions have been reported in the context of non-repudiation [1]. For different application areas (messaging systems, certified mail systems, electronic software distributions, payment systems, and etc.), researchers have proposed different non-repudiation protocols. Louridas and Ray give the comprehensive survey on these protocols in their report [10, 11]. For space limitation, we will mention only a couple of the major protocols using an on-line third party authority to motivate the need for a new non-repudiation protocol.

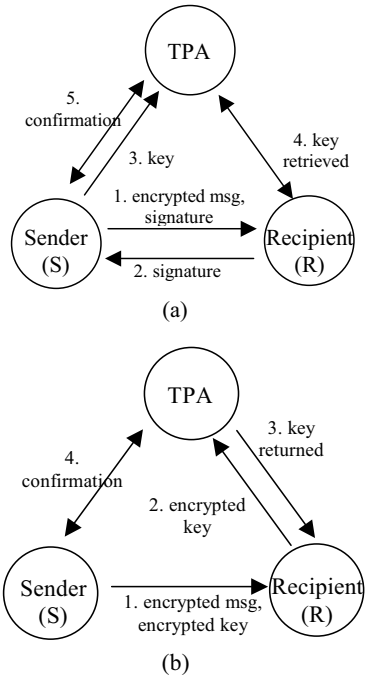


Figure 1. Third Party Authority (TPA)-based non-repudiation protocols

Zhou and Gollmann proposed a "Fair Non-Repudiation Protocol" [2]. The protocol is fair in the sense that the partial evidence generated during the execution of the protocol does not give any advantage to anyone. The sequence of actions is shown in Figure 1(a). In step 1, a message sender S creates a cipher text C by encrypting a plaintext M with an encryption key K. Then, it sends the ciphered text C to a recipient R with its digital signature. R, then, is supposed to acknowledge its receipt of the ciphered text C by returning a digital receipt to S in step 2. After receiving the receipt, S publishes the key K to TPA in step 3, where R retrieves the key in step 4 and S retrieves a confirmation ticket in step 5. The soundness of the protocol was discussed in terms of dispute resolution for each repudiation case. However, as pointed out in [3], the protocol has some drawbacks. First, it is advantageous to the sender because the successful execution of the protocol depends on whether the sender submits the key K to TPA as expected. The recipient has to keep on pulling to check if the key is available at TPA. In terms of the control over the commitment of a transaction, the protocol is not fair to message recipients. In the internet-based applications, especially e-commerce, we believe that the fairness with respect to the control over the commitment of a transaction needs to be considered. Secondly, the encryption key K is visible to TPA, thus, there is a risk of

violation of message security/privacy. Anyone who can access the key K at TPA potentially can read the content of the message M.

Kim reported an extension of Zhou's protocol to address the above problem [3]. The sender set the time limit *t1* and included the information in step 1 of Figure 1 (a). And, the recipient also set the time limit *t2* (where, current time < *t2* < *t1*) to let the sender know the deadline to submit the key. The protocol assumes global time synchronization among senders, recipients, and TPA. In order to transfer secretly decryption keys, his protocol uses the Diff-Hellman algorithm. However, his protocol still requires the recipient to pull the decryption key from TPA until *t2*, which may incur several rounds of communication overhead. Furthermore, it needs the existence of the third party authority for dispute resolution long after a transaction has been committed.

Abadi proposed another protocol shown in Figure 1(b), targeting its application at certified email systems [7]. E-Mail systems require sending messages in a send-and-forget manner. Moreover, mail senders need digital evidences of delivery to make sure that the mail is actually delivered. The protocol was designed to meet these requirements. The protocol works in the following way. In step 1, the sender encrypts the message, encrypts the key with the Third Party Authority (TPA)'s public key, and sends them together to the recipient. The recipient then forwards the encrypted key to TPA to retrieve the key, in step 2. TPA returns the key after decrypting the encrypted key with its private key in step 3 and sends a confirmation of the key delivery in step 4. The above protocol has the following drawbacks. First, the protocol allows TPA to have access to the encryption key. It assumes that TPA is totally trustworthy and will not intentionally violate the privacy policy. The protocol has a high degree of trust dependency on TPA. Second, from the non-repudiation perspective, the protocol is not secure because there is no evidence exchanged except the receipt of key delivery from TPA. The sender can repudiate the sending of a message because the protocol does not require the sender to write his/her signature. And, TPA's confirmation of the key delivery cannot be accepted as proof of a recipient's receipt of the message because the sender can intentionally send an encrypted key that cannot decrypt the message. We argue that TPA's confirmation of key delivery is not equal to the evidence of message delivery.

Ray proposed a non-repudiation protocol that does not use TPA, avoiding possible the single-point-of-failure and availability issues [6]. However, the e-applications in a collaborative computing environment are characterized by not just a single TPA, but any number of TPAs that can be replicated over the Internet. Replication techniques (i.e. transparent request distribution, and policy-based server selection) introduced in [8] can be used to replicate TPA's services in the e-commerce environment. Also, communication between collaborating organizations may go through multiple intermediaries rather than direct communication between message senders and recipients. Our protocol is designed for such an environment.

3. Non-repudiation Protocol Requirements for the Security of Collaborative E-business

Like other protocols [2, 7], we assume that the communication channel between parties involved in message transfer is reliable (i.e. messages are not lost). In addition, we assume that there is no single-point-of-failure or the availability issue with respect to the service provided by TPA, possibly using replication techniques.

Based on these assumptions, which eliminate the problems in executing the protocol correctly, we identify the following requirements regarding non-repudiation in e-commerce. We will show that our protocol satisfies these requirements in Section 5.

- The protocol must protect both parties (i.e. the sender and the recipient) from security threats such as message interception, modification, and replay attacks. This principle could be easily compromised in collaborative e-business because communication channel may go through multiple intermediaries rather than through direct communication. To achieve resource sharing, messages for making resource access requests, sending purchasing orders or requests-for-quote, reporting the status, transmitting data, etc., may go through third parties at multiple network sites. There is a higher risk of security threats.
- The protocol must ensure the confidentiality of transactions so that except the intended receiver, no one else including the third party authority (TPA) involved in the protocol is able to see any part of the transmitted messages. Although TPA collects transactional evidence for settling future disputes, it should not misuse its authority to monitor and collect transactional details.
- The protocol must prevent the message recipient from reading the content of message until he/she has confirmed that the message has been received correctly.
- The protocol must prevent the message sender from sending an invalid message or denying the sending of a message. The protocol should require the digital signature of the message sender not only for message authentication but also for message integrity.
- The protocol must ensure that no communicating party can gain any advantage for having some partial evidence. The result of the protocol must be either the recipient having obtained the message with the

sender's signature and the sender having obtained digital evidence, or neither of them obtained any useful information.

- The settlement of a dispute for a committed transaction should be based solely on the digital signatures of transaction parties. For a committed transaction, the involved parties should not rely on the existence of a third party for dispute settlement because the third party's business may be transient. The third party's responsibility should be limited to facilitating a fair transaction to take place but should not have any further responsibility after the transaction commitments.
- The protocol should be able to satisfy above requirements without causing too much overhead with respect to the number of communication channel needed, transaction delay, and scalability.

4. Non-repudiation Message Transfer Protocol for Collaborative E-business

In this section, we explain our approach to address the requirements identified in Section 3. The following notation adopted from Zhou's paper [2] will be used in the remaining part of this chapter to present our non-repudiation protocol.

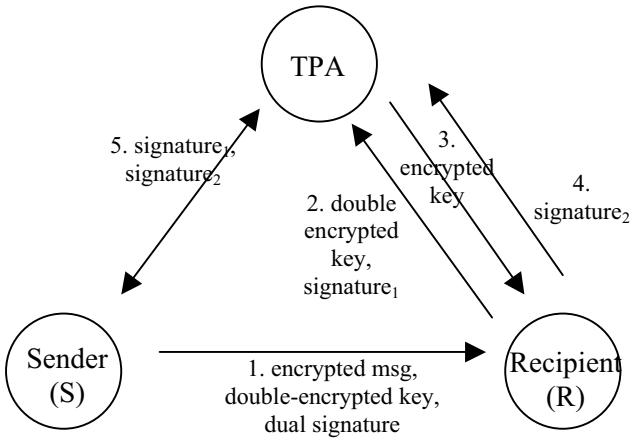
X | Y : concatenation of two messages X and Y.
MD (X) : message digest value of message X.
eK(X) : encryption of message X with key K.
dK(X) : decryption of message X with key K
sK(X) : digital signature of message X with the private key K
$\mathbf{P_A, S_A}$: the public and private key of A.
$\mathbf{A \rightarrow B : X}$: A sends message X to B.

Note that the message is encrypted with a '*secret key*', which are generated at run time and different from the key pair of '*public/private*' keys in PKI. In the following discussion, by an "encrypted key", we mean a secret key that is encrypted with the message recipient's public key. The sender does this encryption to make sure that only the recipient can use the key. The recipient will decrypt the encrypted key using its private key and decrypt the content of a message using the secret key. We also use the term "double encrypted key" to mean a twice-encrypted secret key that is encrypted with the recipient's public key first and then with the public key of a third party authority (TPA) involved in the protocol. The sender creates the double encrypted key to ensure that if and only if the recipient performs an obligation, he/she is entitled to access the secret key. The TPA will be responsible for monitoring the fulfillment of the recipient's obligation (i.e., acknowledgement).

Another technique we use in designing our protocol is 'dual signature'. The dual signature is a verification technique used in Secure Electronic Transaction (SET) to link a purchase order and the purchase authorization with a credit card [9]. In SET protocol, a purchase order message from a customer to a merchant consists of two parts: one is the main content containing details of the purchase order, and the other is the authorization code containing the credit card number of the customer. The latter is usually sealed to protect the customer's credit card number from the merchant. The merchant then gets the main content of the purchase, whereas the credit card service provider receives the authorization code. The protocol needs a way to prove that two parts are actually linked for the settlement of possible future disputes. For instance, the authorization for purchasing product M should not be used for purchasing product N. The dual signature was proposed to prevent the two parts of a transactional message from being used separately.

We use the same idea to make a link between an encrypted message and a sealed decrypting key. The message sender certifies the linkage by providing the dual signature to the recipient. The dual signature in our protocol serves the following three purposes. First, the recipient can use the signature to check the integrity of the received message because it contains the message digests of both the message content and the key information. Second, it is the sender's certification about the linkage between the encrypted contents and the secret key information. This is needed to prevent the sender's misbehavior, i.e. sending the incorrect decryption key information so that the recipient will not be able to read the message. The sender cannot deny having sent the wrong key, because he/she is the only one who can generate the signature. The recipient can later use the signature to prove his/her case. Third, the recipient will have the proof that the message has come from the particular sender because only the sender can generate the signature.

Figure 2 gives a high-level sketch of the new non-repudiation protocol without going into details. To simplify the figure, we omit the transaction ID. By 'message type *i*' we mean the contents of the message exchanged in step *i*.In step 1, the sender generates a secret key randomly and uses it to encrypt the message. It then double-encrypts the secret key (encrypted with the recipient's public key and then with the third party authority's public key). The secret key is encrypted twice because the sender depends on the third party authority to check the key releasing policy before releasing the key but does not want the authority to access the key. The dual signature is also generated on concatenation of the message digest of the ciphered text (the encrypted content), and the message digest of the double encrypted secret key. All of this information is sent to the recipient in step 1.



msg type 1. S → R : ***tid*** || ***S*** || ***em*** || ***dek*** || ***dual_signature***
where K : a symmetric key generated by A
tid : transaction id or nonce
em = eK(msg),
ek_from_S= $e\mathrm{P}_R$(K), dek = $e\mathrm{P}_{TTP}$(ek_from_S),
md1 = MD(em), md2 =MD(dek),
dual_signature = tid || md1 || md2 || $s\mathrm{S}_S$(tid||md1||md2).

msg type 2. R → TPA : ***tid*** || ***S*** || ***R*** || ***md1*** || ***dek*** || $s\mathrm{S}_R$***(tid||md1)***

msg type 3. TPA → R : ***tid*** || ***ek_from_TTP***
where ***ek_from_TPA*** = $d\mathrm{S}_{TTP}$(***dek***).

msg type 4. R → TPA : ***tid*** || $s\mathrm{S}_R$**(MD(*ek_from_TTP*))**.

msg type 5. TPA → A : ***tid*** || $s\mathrm{S}_R$***(tid||md1)*** || $s\mathrm{S}_R$**(MD(*ek_from_TTP*))**.

Figure 2. Secure Message Transfer Protocol for E-commerce

When receiving the message of step 1 (that is, tid || A || *em* || *dek* || *dual_signature*), the recipient checks the integrity of both the encrypted main content *em* and the double-encrypted key *dek* by comparing with the dual signature. Note that only when the integrity is preserved, the recipient initiates the second step. The progress to the second step implies the recipient's confirmation of receiving both the encrypted content and the double-encrypted key correctly, so the recipient cannot claim later that he/she had received the wrong encrypted message content.

In step 2, the recipient forwards the double-encrypted key to the third party authority (TPA), along with its signature to acknowledge the correct receipt of the encrypted message. The recipient is required to send his/her digital signature on the cipher text *em*, in order to have access to the key. The recipient's signature provides significant digital evidence that the recipient had attempted to access the secret key. TPA will store the signature temporarily for signature distribution at the end of the protocol. Note that the recipient cannot write a signature on a cipher text *em'* (where *em* is actually what the sender had sent and *em'* is not equal to *em*) because he/she cannot construct the sender's dual signature that contains *em'* which is needed if there is a lawsuit.

In step 3, the third party authority (TPA) decrypts the double-encrypted key and releases the encrypted key to the recipient. Note that TPA is still unable to access the secret key because it is still sealed by the recipient's public key. Only the recipient can access the secret key. TPA will log the execution of step 3 in an auditing system. Then, TPA waits for the acknowledgement from R. In case TPA does not receive the acknowledgement within a certain timeout (even after redoing step 3 several times and contacting the recipient off-line), TPA detects the

recipient's misbehavior. Even in such a case, the sender S is protected because TPA has the first signature in step 2 and log of the execution of step 3, which constitutes the undeniable evidence that the recipient receives the message correctly.

In step 4, the recipient sends to the third party authority (TPA) the confirmation of receiving the key. The recipient creates the signature on the digested secret key. Note that the key is digested before being signed with the recipient's private key. Message digest uses the one-way hash function, which make it impossible to reconstruct the original content from the digested data. Therefore, TPA cannot access any key information from the signature.

Lastly, the protocol ends with TPA forwarding two signatures in step 2 and 4 from the recipient to the original sender. These two signatures represent the recipient's acknowledgments of the receipts of the encrypted ciphertext and the secret key, respectively. TPA collects and forwards these signatures so that the sender does not need the existence of TPA after the transaction is completed. The sender checks if the recipient returns the digital receipts correctly. This can be done because the sender knows what the ciphertext and the secret key he/she had sent. If the sender detects a mismatch with received signatures, it retrieves the execution records of step 2 and step 3 as evidence. The sender can prove the recipient's misbehavior by demonstrating that his secret key is not matched with the key signed by the recipient but is matched with the double-encrypted key *dek* in step 2 and the key *ek_from_TPA* in step 3. Here, matching means that the double encryption of the sender's secret key is equal to dek in step 2 and the encryption of the secret key with the TPA's public key produces *ek_from_TPA* in step 3.

5. Discussion

In this section, we give an informal analysis on how our protocol satisfies the requirements identified in Section 3. By describing this analysis, we want to clarify the implicit logic and resolution scheme, which was not described in Section 4.

Requirement 1: *The protocol protects the involved parties from well-known message security threats such as message interception and modification, and replay attacks.*

Argument: To protect from message interception and modification, we use message digest and encryption techniques. The integrity of the message can be checked with the message digest value and the confidentiality of the message is protected through encryption. No one but the recipient can read the message content. To protect from replay attacks, the protocol generates a fresh transaction id (TID) every time.

Requirement 2: *The protocol ensures the confidentiality of transactions so that, except the recipient, no one else including the third party authority (TPA) involved in the protocol is able to know the contents of a transmitted message.*

Argument: The only way to see the message between the sender and the recipient is through the secret key that encrypts the message. The secret key is encrypted twice to prevent the third party authority (TPA) as well as other intermediaries from getting access to the key. And, in step 4, the recipient signs on the message digest of the secret key, but not on the secret key itself. Thus, TPA does not know the key, even though he facilitates the key exchange. Note that a message digest is one-way so that it is impossible to reconstruct the original content from a message digest.

Requirement 3: *The protocol must prevent the message recipient from reading the content of a message until he/she has confirmed that the message has been received correctly.*

Argument: It is after step 3 that recipient B can read the entire message; B cannot read the message without receiving the encrypted message *em* in step 1, and, B cannot read *em* without receiving the encrypted key from TPA in step 3. In between, step 2 forces recipient B to sign that he has received the cipher text correctly. B's signature and the execution record of step 3 constitute undeniable digital evidence (in a sense, fingerprint in forensics) that the recipient was able to access the message content in the transaction. There are two cases in which the recipient B may misbehave after step 3. Either way, the sender A is protected from the recipient's misbehavior.

- Recipient B does not take step 4. In this case, the protocol detects the recipient's misbehavior when timeout has been reached and the TPA does not receive the recipient's acknowledgement of an encrypted key. Sender A proves the recipient's misbehavior by presenting *tid*, *em* and *dek* and showing that *tid* and *em* are matched to $sS_B(tid||md1)$ and *dek* is matched to *ek_from_TPA* of step 3.
- The recipient B purposely signs on a fake key in step 4 to deny the transaction later: The sender A can show B's dishonesty by showing that key A sent at step 1 corresponds to *ek_from_TPA* at step 3 but not to the second signature of the recipient B in step 4.

Either way, the protocol detects the recipient's misbehavior. The protocol ends properly when the recipient confirms that he has received both the encrypted content and the key. The protocol either ends normally or detects the recipient's misbehavior.

Requirement 4: *The protocol prevents the message sender from sending an invalid message or denying sending a message.*
Argument: Only when step 5 is reached, sender A can obtain a receipt. However, step 5 cannot be reached if the sender A has sent an invalid message. The recipient B would not give the first signature at step 2 if he did not receive the encrypted message correctly. B can check this with the sender's dual signature, and B would not give the second signature at step 4 if he cannot read the encrypted message with the key received from TPA. In court, B can demonstrate his position by showing that key K cannot decrypt the message and key K corresponds to the key part of the dual signature received at step 1.

Sender A cannot deny having sent a message M (containing *em*, *dek* and *dual signature*) because of the dual signature. It is only the sender who can generate the signature. If the sender denies having sent either *em* or *dek* to the recipient B and claims having sent a different message *em'* (where *em'* is not equal to *em*) or *dek'* (again, *dek'* is not equal to *dek*), B can refute that claim by showing the sender's ***dual signature*** on *em* and *dek* that has been received.

Requirement 5: *The protocol must ensure that no communicating party can gain any advantage for having some partial evidence.*
Argument: If protocol ends at step 1, even if recipient B has the sender's dual signature, B cannot take any advantage because he/she has no way to access the message content. If it ends at step 2, the sender A cannot claim anything because recipient B has yet to sign the receipt of the secret key. Even if the protocol ends at step 3, the sender A is protected because the recipient's signature over the encrypted message and log information over the key are left to the third party authority (TPA) when executing step 3. If it ends in step 4, A can contact TPA later to retrieve the signature.

Requirement 6: *Any dispute for a committed transaction must be resolved solely based on the digital signatures of transaction participants. For a committed transaction, both parties should not rely on the existence of a third party for dispute resolution.*
Argument: At the end of the protocol, the recipient ends up having senders' dual signature and the sender having the recipient's confirmation. They do not need the third party's presence in court. Signatures of both parties are enough to resolve any dispute.

Requirement 7: *The protocol should be able to satisfy above requirements without causing too much overhead with respect to the number of communication channels needed, transaction delay, and scalability.*
Argument: The implementation of our protocol requires only 3 network connections to be established because message 2, 3, and 4 will be exchanged in a channel (or through a session of a communication connection). This is better than Zhou's protocol (at least 4 connections) and has same overhead as Abdi's. For reducing transaction delay and maintaining scalability, our approach requires multiple replicated services of TPAs, which are the always true when the third party provides a trust service.

The proposed protocol is based on the assumption of the reliable communication channel. The recipient can not claim to never receive the encryption key in step 3 since TPA will repeat step 3 until it receives a signature in step 4 or contacts the recipient off-line as well. Note, however, that the sender is still protected from non-repudiation of the recipient because of the first signature in step 2 and log information in step 3.

Another way to cope with the correct termination of the protocol is through trusted implementation. For instance, the recipient-side implementation, especially the code dealing with step 3 and 4, would be a third-party's one or at least would be inspected, certified by a trusted third party. This requires TPA to check trustworthiness of the implementation from the certificate of the code before releasing the encrypted key.

Our protocol implementation consists of the sender-side SOAP handler, the receiver-side SOAP handler, and the TPA-side SOAP handler. These handlers intercept the SOAP messages and apply the cryptographic operations. They are distributed and deployed in the web service environment. We assume, to prevent recipient's denial of receiving key in step 3, that these handlers are developed or at least certified by a trusted third party.

One might argue that the problems we point out in this paper can be solved easily by modifying the Zhou's protocol, i.e. the sender sending a double encrypted key to the third party in Step 3 and having the key pushed to the recipient in the next step. This certainly solves the drawback of Zhou's: It does not require the recipient to keep pulling the key from the third party and guarantees the secrecy of the message content. However, it makes the recipient dependent on the third party seriously and vulnerable if the third party does not play fair to the recipient. Consider the case when the third party and the sender team up to get the recipient in trouble in the following way. Once receiving the double encrypted key, the third party does not push the encrypted key to the recipient and claims it did. There is no way to protect the recipient from this trust dependency in the modified version of Zhou's protocol, where the recipient would be accused of the default. Our protocol is safe from this situation in the following sense. In step 1, our protocol allows the recipient to have the dual signature so that he can have partial evidence on the secret key information.

And, TPA can not claim that it gave the encrypted key without really giving it to the recipient, because the recipient would not give the second signature in step 4. TPA has to give the encrypted key to get the second signature.

6. Summary

In e-commerce, interactions and exchanges of business documents and data between customers and merchants in B2C e-commerce or between business partners in B2B e-commerce can be abstracted as message transfers between them. The confidentiality of message contents and the non-repudiation of a message transfer are essential requirements. Transaction details between customers and merchants should remain confidential. Customers and merchants on the Internet should not deny any committed transactions. There are several published papers that address confidentiality and non-repudiation problems using various levels of trust and dependency on a third party authority (TPA) and with different features and weaknesses. Collaborative e-business poses several new challenges, which is not supported by the existing approach. In this paper, we have presented a new security protocol to address the new requirements. Unlike the existing approach, the delivery signature of the third party is not accepted as evidence. Instead, the role of the third party authority is reduced as a facilitator for the exchange of business end-entities' digital signatures and as a watchman to detect their misbehavior. Once the protocol is committed, any dispute can be resolved without the presence of the third party. Our approach also addresses confidentiality of transactions by hiding the decryption key from the third party authority. Moreover, our proposed protocol protects message recipients from message senders' misbehavior so that it would not allow the message senders an opportunity to intentionally delay the commitment of transactions for financial gains. In subsequent research we will investigate how our approach can be applied to various Internet-based applications such as electronic payment systems, auction systems, Internet-based software distribution, ticket sale applications, etc. And, we are planning to investigate how to formally verify that our approach is safe.

7. References

[1] Zhou J., "Non-repudiation in Electronic Commerce," Artech House, Computer Security Series, 2001.

[2] Zhou J. and Gollmann D., "A Fair Non-Repudiation Protocol," In Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy, May 1996, pages 55--61.

[3] Kim K., Park S., and Baek J., "Improving Fairness and Privacy of Zhou-Gollmann's Fair Non-Repudiation Protocol," In Proceedings of 1999 ICPP Workshops on Security (IWSEC), pp. 140-145, IEEE Computer Society, Sep. 21-22, 1999.

[4] Asokan, N., Schunter, M. and Waider, M., "Optimistic Protocols for Fair Exchange," In Proceedings of the 4th ACM Conference on Computer and Communication Security, pp. 4-17, April 1997.

[5] Bao, F., Deng R., And Mao, W., "Efficient and Practical Fair Exchange Protocols with Off-line TTP," In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, California, 1998.

[6] Ray, I., and Ray, I., "An optimistic Fair exchange e-commerce Protocol with Automated Dispute Resolution," In Proceedings of the 1st International Conference on Electronic Commerce and Web Technologies, London, UK, 2000.

[7] Abadi M., Glew N., Horne B., and Pinkas B., "Certified Email with a Light On-line Trusted Third Party: Design and Implementation", the eleventh International World Wide Web Conference, Honolulu, Hawaii, USA, 2002.

[8] Rabinovick M., and Spatscheck, "Web caching and replication," Part III Web Replication, Addison Wesley, 2002.

[9] Lewis P., Bernstein A., and Kifer M., "Databases and Transaction Processing: An Application-oriented Approach," Chapter 27. Security and Internet Commerce, pp. 915-949, Addison Wesley, 2002.

[10] Louridas P., "Some guidelines for non-repudiation protocols", ACM SIGCOMM Computer Communication Review, v.30 n.5, October 2000.

[11] Ray, I., and Ray, I., "Non-repudiation in Electronic Commerce," Artech House, Computer Security Series, 2001.