Non-Repudiation in An Agent-Based Electronic Commerce System

C.-C. Liew W.-K. Ng E.-P. Lim B.-S. Tan K.-L. Ong
Centre for Advanced Information Systems, School of Applied Science
Nanyang Technological University, Singapore 639798, SINGAPORE
{awkng,aseplim}@ntu.edu.sg

Abstract

Abecos is an agent-based e-commerce system under development at the Nanyang Technological University. A key factor in making this system usable in practice is strict security controls. One aspect of security is the provision of *non-repudiation* services. As protocols for non-repudiation have focused on -message non-repudiation, its adaptation to afford non-repudiation in a communication session for two agents in Abecos is inefficient. In this work, we investigate and propose a protocol for enforcing non-repudiation in a session. The protocol is believed to be applicable in any e-commerce system; agent- or not agent-based.

1 Introduction

Electronic commerce is an emerging paradigm of business on the fast growing Internet which holds great potential and opportunities for business organizations. Besides access to new and bigger markets, e-commerce can help to bring about reduced costs and faster turnaround times by streamlining and integrating processes along the entire business value chain. However, as in any other modes of business, we cannot assume that participating parties in an e-commerce activity play fair and abide by code of moral conduct. The mere fact that business is being performed online over an unsecured media is enough to entice criminal activity to the Internet. Hence, the key factor in making e-commerce flourish lies in enforcing security in a commerce system within an unsecured environment. One aspect of security is the provision of *non-repudiation* services.

Many non-repudiation protocols have been proposed. These protocols are mostly described in a *wrapper* context: A party A wants to send a message M to B; to enforce non-repudiation of M, these protocols introduce additional messages so that collectively, sufficient evidence is gathered and the sending and receipt of M cannot be repudiated. Hence, non-repudiation is enforced for *one* message (M) or *an* exchange of a message.

In an agent-based e-commerce system, software agents engage in commerce activities on behalf of buyers and sellers, which can be business organizations or individuals. In Abecos, we divide the above process of buying and selling into five phases—directory search, product information enquiry, negotiation, and finally, payment and delivery. In each phase, a sequence of messages are exchanged between two agents. That is, each phase is a communication session between two parties. For consumers to rely on software agents in a system such as Abecos, we need security and accountability in each of the sessions. Having a non-repudiation service enforced prevents any communicating party from denying any action it has committed.

Direct adaptation of known non-repudiation protocols result in message and traffic overheads because they are designed for the tranmission of *one* message only. In this paper, we propose a non-repudiation protocol for a communication session in an agent-based e-commerce system. This paper is a documentation of our preliminary and ongoing work in enforcing security in an e-commerce system.

2 Related Work

Non-repudiation protocols are either based on a trusted third party or on the gradual exchange of information. For the former, third party communication is a bottleneck while the latter places a heavy load on communication resources. There are some non-repudiation protocols which use a (non-trusted) third party either to deliver the message or to generate, extract or keep the required evidences. In this section, we briefly review three of these protocols.

2.1 Fair Non-Repudiation Protocol (FNP)

One important key in a non-repudiation protocol is to ensure that both communicating parties are neither at any advantageous position at any instance of the communicating phases. A Fair Non-repudiation Protocol (FNP) was proposed in [23]. Its objective is to provide the originator and recipient with valid irrefutable evidences after the completion of the protocol, without giving any party an advantage at any point during the execution of the protocol.

Assuming that each entity is equipped with its own pair of public keys for digital signature and verification, the protocol is best summarized formally as follows.:

1.	$A \rightarrow B$	:	f_{EOO}, B, L, C, EOO
2.	$B \rightarrow A$	:	f_{EOR}, A, L, EOR
3.	$A \rightarrow TTP$	:	f_{SUB}, B, L, K, sub_K
4.	$B \leftrightarrow TTP$	:	$f_{CON}, A, B, L, K, con_K$
5.	$A \leftrightarrow TTP$	:	$f_{CON}, A, B, L, K, con_K$

It should also be noted that in this fair non-repudiation protocol, the ciphertext C (encrypted by key K), is not used for confidentiality purposes. The explaination of the notation used and a more elaborate discussion of this protocol can be found in [23].

2.2 Certified Electronic Mail Protocol (CMP)

A Certified electronic Mail Protocol (CMP) was proposed in [2]. The protocol uses the same idea as FNP and involves three parties: the mail sender A, the mail recipient B, and the Trusted Third Party TTP. There are two variations of the CMP model: CMP1 and CMP2. The difference between the two is that CMP2 provides content confidentiality protection; in CMP1, the encrypted message and the key is passed to the third party for decryption while in CMP2, only the key, which is encrypted under the third party's public key, is sent. The responsibility of decrypting the ciphertext lies with B. Since the size of the key is usually smaller than the size of the message, it is more efficient to use CMP2 than CMP1. Protocol CMP2 is described as follows:

1.	$A \rightarrow B$	:	$A, B, TTP, H(M), e(V_P, K), e(K, M + EOO)$
2.	$B \rightarrow TTP$	:	$A, B, TTP, H(M), e(V_{TTP}, K), e(K, M + EOO), EOR$
3.	$TTP \rightarrow B$	:	$A, B, TTP, H(M), e(V_B, K), EOO_{TTP}$
4.	$TTP \rightarrow A$	:	$A, B, TTP, H(M), EOR, EOD$

As pointed out in [2], the problem of selective receipt of the decrypted session key may happen at the last step. Three approaches to solve the problem are proposed in the paper [2]. The CMP protocol also suffers from one major drawback. The mail is transmitted at least twice in the protocol. Although the protocol reaches the lower bounds on the number of messages and rounds, it is not efficient when the mail item becomes very large.

2.3 Optimistic Fair Exchange

Asokan *et al.* proposed a protocol for fair exchange [9] where the third party is involved only when a player cheats or plays unfairly, or in the event of a fault in the communication. The main idea underlying the protocol is that both principals start by promising each other the exchange of information. The information is then exhanged for non-repudiation evidences during the course of the protocol. If anything goes wrong during the run of the protocol, an error recovery phase is initiated. The party initiating the recovery will then have to sent all the messages exchanged previously to the third party to solve the dispute.

3 Non-Repudiation in A Session

The motivation underlying the design of a non-repudiation protocol for use in a session is that in an e-commerce system such as ABECOS, non-repudiation service is invoked several times and across different phases. In ABECOS, we need non-repudiation service in the enquiry, negotiation and payment phases. As such, we would like to view a series of message transfers that span across different phases of ABECOS as a non-repudiation session.

In this section, we propose a protocol NRSP (Non-Repudiation Session Protocol) for to provide fair non-repudiation in a session. We define a session as a series of *request-response* messages between two parties A and B: A sends the first message; B processes the received message and formulates the response and sends it back to A; A processes the reply and formulates the next message for B; and so on. The process continues for a consecutive sequence of messages until one party calls for termination of the session. Ideally, the protocol should incur less message overheads than direct application of FNP and CMP on each of the messages in the session.

3.1 Symbols and Notations

In this protocol, we attempt to make the non-repudiation session a fair one. Let us define the following notations to be used in this protocol:

- f: A flag denoting the purpose of the message.
- K: Symmetric key used for encrypting a message.
- V: Public key of a principal identified in the subscript.
- $token_n$: An *nth* token consisting of the session identification number, $SessId$, and the message sequence number n.
- $e(K_{An}, M_{An})$: Encryption of nth message send by party A. using symmetric key K_{An}.
- N_A and N_B: A secret number between A and TTP or B and TTP. This number is known to them only when the parties requests for opening of a session.
- key_info_{An}: Data items consisting of K_{An}, N_A, $sS_B(token_n)$.
- $e(V_{TTP}, key\_info_{An})$: Encryption of key information using TTP's public key, V_{TTP}.
- $sS_A(data\ items)$ or $sS_B(data\ items)$: Signature of A or B on the particular data items.
- $P_1 \leftrightarrow P_2$: P_1 performs a *ftp get* operation on P_2 in order to retrieve some data items.
- $EOR_n = sS_B(A, token_n, M_n)$: Evidence of receipt of the nth message.

For generality purposes, parties A and B are used to denote *sender* and *recipient* respectively in the explanatory of the notations above. In the following sections, we will use A and B to mean two individual parties in communication. Both parties are capable of sending and receiving messages.

3.2 Opening of A Session

When party A and B request for opening of a session, a secret number N_A, N_B is given to them respectively. These secret numbers are known only to A or B themselves and also to TTP. It is used so that TTP is able to ascertain the identity of the party sending the encrypted key to the receiver. These secret numbers replaces the digital signatures that would otherwise be needed for the purpose. Verification of digital signatures is generally more time consuming than just performing a table lookup on the secret number.

The exchanges of messages in the session starts off by having the party whom is expecting to receive the first message to send out the first *token* of the session. Suppose A is the first sender, we have the following flow of data:

- $B \rightarrow A : f_{token}, token_1, sS_B(f_{token}, token_1)$

The token is a string of numbers consisting of the *SessId* and the sequence number of the message. It provides the control over the flow of messages in sequence within the session. As an analogy, we can view a *token* as a numbered return envelope. Without the return envelope from the intended recipient, the sender cannot send the message. The use of *token* is further described in the following sections.

3.3 Exchanging of Messages

The main idea of our protocol is to encrypt the intended message using a symmetric key. The key is further encrypted using the third party's public key. The encrypted key and the cipher text are then sent to the recipient. The recipient, upon receiving the message in encrypted form, will not be able to read the content and hence he sends the key encrypted with TTP's public key to the TTP for decryption. The non-repudiation evidence of origin is generated when the sender signs the encrypted message digitally using his private key and TTP published the symmetric key. The evidence for non-repudiation of receipt is generated when the recipient signs the message to request for the key.

The steps below shows the use of the protocol in a session starting from nth message: (We assume that A has obtained $token_n$)

1. $A \rightarrow B \quad : \quad f_{SMsg}, B, token_{n+1}, e(V_{TTP}, K_{An} + N_A + sS_B(token_n)), e(K_{An}, M_n), sS_A(token_{n+1}), signature\_of\_A, EOR_{n-1}.$

   where $signature\_of\_A = sS_A(f_{SMsg}, B, e(V_{TTP}, K_{An} + N_A + sS_B(token_n)), e(K_{An}, M_n))$.

A sends the nth encrypted message to B. The key information is in turn encrypted using TTP's public key. The key information that A has to include consists of his secret number with TTP, B's signature on the $token_n$, as well as the key used to encrypt or decrypt the message. In addition, A will also send the $(n+1)th$ token and the evidence of receipt of the previous message to B. B at this point in time, will check the validity of the message by verifying the digital signature of A on the message, on the $token_{n+1}$ and on the previous message. Thus, B would have obtained the evidence of receipt for the previous, $n - 1th$ message.

2. $B \rightarrow TTP \quad : \quad f_{GKey}, A, token_n, e(V_{TTP}, K_{An} + N_A + sS_B(token_n)), sS_B(f_{GKey}, A, token_n, e(V_{TTP}, K_{An} + N_A + sS_B(token_n))).$

When TTP receives the request from B, he will verify the signature of B. And if it is valid, TTP will proceed to decrypt the key information using his own private key. A table lookup will be performed on N_A to check that it matches the secret number he has with A. Next, TTP uses same public key of B to verify that B has signed $token_n$. In addition, TTP checks with his database entry to ensure that the $token_n$ is a valid sequence. Once verified, TTP is ascertained that A and B are communicating with the right party and the message sequence number is matched and valid.

The TTP will retain the signature of B and store in its database for a designated period of time. TTP will also store the decrypted key in a public directory where B can retrieve it. We note that the key used for encryption of the message is not to provide confidentiality of the message but to support non-repudiation service.

3. $B \leftrightarrow TTP \quad : \quad f_{Key}, A, B, token_n, K_{An}, sS_{TTP}(f_{Key}, A, B, token_n, K_{An})$

B will retrieve the encrypted key which he sends to TTP for decryption after an agreed time duration. Thus, it is the recipient's responsibility to get the key from TTP; this avoids the problem of *selective receipt.* Consider the case if TTP is to send the decrypted key back to the recipient. The recipient can choose not to acknowledge the receipt of the key even if he has received it; thus the sender is at a disadvantaged position. This is the problem of selective receipt. The mode of retrieval is proposed to be a *ftp get* operation.

4. $B \rightarrow A \quad : \quad f_{SMsg}, A, token_{n+2}, e(V_{TTP}, K_{B(n+1)} + N_B + sS_A(token_{n+1})), e(K_{B(n+1)}, M_{n+1}), sS_B(token_{n+2}), signature\_of\_B, EOR_n.$

   where $signature\_of\_B = sS_B(f_{SMsg}, A, e(V_{TTP}, K_{B(n+1)} + N_B + sS_A(token_{n+1})), e(K_{B(n+1)}, M_{n+1})))$.

After B has processed the message and formulated a response, he then sends a reply using the same protocol again.

In the event that B does not send the evidence of receipt to the sender after obtaining the decryption key from TTP, the sender A may then start a recovery phase.

1. $A \leftrightarrow TTP$: $f_{Key}, A, B, token_n, K_{An}, sS_{TTP}(f_{Key}, A, B, token_n, K_{An})$

(Party A retrieves the evidence of publication of key from TTP.)

2. $A \rightarrow TTP$: $f_{GSub}, B, token_n, sS_A(f_{GSub}, B, token_n)$

(Party A further request that TTP send him the evidence that B submitted the encrypted key to TTP.)

3. $TTP \rightarrow A$: $f_{KSub}, A, B, sS_B(f_{GKey}, A, token_n, e(V_{TTP}, K_{An} + N_A + sS_B(token_n)), signature\_of\_TTP.$

where $signature\_of\_TTP = sS_{TTP}(f_{KSub}, A, B, sS_B(f_{GKey}, A, token_n, e(V_{TTP}, K_{An} + N_A + sS_B(token_n)))$

(TTP grants A's request and send him the evidence.)

In the recovery phase, A simply obtains from the third party, the evidence that the B has sent the key for decryption and that the TTP has published the key. Using these evidences, A can prove that B has indeed received his message.

4 Verification of Fairness

Let us examine the examine the fairness of this protocol by looking at the most critical step in this protocol. When B has retrieved the key from TTP, B can choose to continue with the session and send EOR to A or to play unfairly by ending this protocol abruptly.

If B does not play fair such that A will not receive the EOR_n of the *nth* message (B stops at step 3), A will then have to get the evidences from TTP. From the evidences he gathered, he can then prove to the judge that B has indeed sent a request to TTP to decrypt the key for the *nth* message in the session. And also prove that TTP has verified the validity of the request and published the key publicly.

At this point in time, the judge is convinced that B must have received the encrypted key and some form of the *nth* encrypted message from A. To be fair to B, the judge request B to forward the *nth* message in the session which B received from A. If A tries to fake an *nth* message different from the one he actually sent to B and claims that B has actually received the one which he made up, the data items and evidence provided by B would be able to tell.

The judge can verify A's signature on the encrypted key and encrypted message of the *nth* message which is provided by B. And then, the judge compares the encrypted message which B receives and the one A claims to have sent. If they are the same, the judge concludes that B has received the *nth* message. If not, the judge concludes that A's claim should be rejected.

5 Efficiency Analysis

Now consider a simple session where a command is sent to a party and a response was returned. CMP and FNP each takes 8 and 10 messages respectively to accomplish the session. In the case of NRSP, we find that we need 6 messages plus 1 message for obtaining the evidence of receipt for the reply. Therefore, a total of 7 messages are needed. (We have omitted the overhead incurred when establishing and terminating the session.)

Clearly, when we extrapolate the simple session to consist of n consecutive command-response messages, the overhead becomes insignificant. Thus, we have:

Result 1 *For n consecutive command-response messages, CMP, FNP and NRSP incur $8n$, $10n$ and $6n+1$ messages respectively. Therefore, the savings in terms of the number of messages is at least $2n-1$, or approximately 12.5%.*

It can be shown that if we assume the following size:

- key used for DES is 56-bits in length,
- identifiers and token are 32-bits in size,
- flag used in the protocols are negligible in size,
- mail identifier (L), digital signatures, output of the hash function are 128-bits,
- actual message size is 1,024 bytes,
- secret number is 16-bit,

Then, for $n > 12$, NRSP is more efficient.

Result 2 *The amount of byte overhead saved when using NRSP over FNP for a session consisting of n two-message exchanges in $O(n)$.*

6 Conclusions

Non-repudiation service protects a party involved in a transaction against the other party denying that a particular event or action took place. In this paper, we presented a non-repudiation protocol NRSP that can be used in a session where messages are being exchanged consecutively. The need for this protocol arises in the ABECOS research system where buyer and seller agents representing businesses and individual engage in commerce activities. As agents progress from one phase of activity to another, NRSP can be used to provide and enforce non-repudiation within a phase. We believe this protocol is applicable in any e-commerce system; agent- or not agent-based.

In this paper, we have shown that NRSP is more efficient than CMP and FNP in terms of the message overhead. We compare and show that it is more efficient than simple adaptations of existing protocols. As part of our ongoing work on the ABECOS research project, we shall be looking into improving NRSP further. In particular, we will attempt to improve the protocol so that there is less reliance on the third party.

References

[1] Martin Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. In *Proceedings of 1994 IEEE Computer Symposium on Research in Security and Privacy*, Oakland, California, 1994.

[2] Robert H. Deng, Li Gong, Aurel A. Lazar, and Weiguo Wang. Practical protocols for certified electronic mail. *Journal of Network Systems Management*, 4(3), 1996.

[3] William M. Farmer, Joshua D. Guttman, and Vipin Swarup. Security for mobile agents: Authentication and state appraisal. In *Proceedings of 4th European Symposium on Research in Computer Security*, Rome, Italy, Sept 1996.

[4] Warwick Ford and Michael S. Baud. *Secure Electronic Commerce: Building the Infrastructure for Digital Signatures and Encryption.* Prentice Hall, 1997.

[5] VISA International and MASTERCARD International. Secure electronic transaction (set) specification. version 1.0. May 1997.

[6] R. Kailar. Reasoning about accountability in procotols for e-commerce. In *Proceedings of 1995 IEEE Computer Symposium on Research in Security and Privacy*, Oakland, California, 1995.

[7] Gary McGraw and Edward Felten. *Java Security.* Wiley Computer Publishing, 1997.

[8] Alexandros Moukas, Robert Guttman, and Pattie Maes. Agent-mediated electronic commerce: An mit media laboratory perspective. In *Proceedings of International Conference on Electronic Commerce 98*, Seoul, Korea, April 1998.

[9] M. Schunter N. Asokan and M. Waidner. Optimistic protocols for fair exchange. In *Proceedings of 4th ACM Conference on Computer and Communications Security*, Zurich, Switzerland, April 1997.

[10] Charles P. Pfleeger. *Security in Computing.* Prentice Hall, 1989.

[11] Michael Reiter, Kenneth Birman, and Li Gong. Integrating security in a group oriented distributed system. In *Proceedings of the 1992 IEEE Symposium on Research in Security and Privacy*, Oakland, California, 1992.

[12] Artur Romo and Miguel Mira da Silva. An agent-based secure internet payment system for mobile computing. In *Proceedings of IFIP International Conference on Trends in E-commerce '98*, Hamsburg, Germany, 1998.

[13] Phyllis A. Schneck and karsten Schwan. Authenticast: An adaptive protocol for high-performance, secure network applications.

[14] William Stallings. *Data And Computer Communications.* Prentice Hall, 5th edition, 1997.

[15] P. F. Syverson and P. C. van Oorschot. On unifying some cryptographic protocol logics. In *Proceedings of 1994 IEEE Symposium on Research in Security and Privacy*, Oakland, California, 1994.

[16] Chelliah Thirunavukkarasu, Tim Finn, and James Mayfield. Secret agents—a security architecture for the kqml agent communication language. In *Proceedings of International Conference on Information and knowledge Management '95 Intelligent Information Agents Workshop*, Baltimore, December 1995.

[17] Bill Venners. Java platform's new security framework — javaone today. *http://www.javaworld.com/ javaworld/ javaone98/ j1-98-security.html, JavaWorld.*

[18] M. Waidner. Development of a secure electronic marketplace for europe. In *Proceedings of Computer Security ESORICS '96*, Springer, Berlin, 1996.

[19] G. H. Yao, W. K. Ng, and E. P. Lim. Toolkits for a distributed, agent-based web commerce system. In *Proceedings of the International IFIP Working Conference on Trends in Distributed Systems for Electronic Commerce '98*, Hamburg, Germany, June 1998.

[20] X. Yi, X. F. Wang, and K. Y. Lam. A secure intelligent trade agent system. In *Proceedings of IFIP International Conference on Trends in E-commerce '98*, Hamsburg, Germany, 1998.

[21] Michael Zapf, Helge Muller, and Krt Geihs. Security requirements for mobile agents in electronic markets. In *Proceedings of IFIP International Conference on Trends in E-commerce '98*, Hamsburg, Germany, 1998.

[22] J. ZHOU. *Non-Repudiation.* PhD thesis, University of London, 1997.

[23] J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *Proceedings of 1996 IEEE Symposium on Security and Privacy*, Oakland, California, May 1996.

[24] J. Zhou and D. Gollmann. Observations on non-repudiation. In *Asiacrypt'96*, Kyongju, Korea, November 1996. Lecture Notes in Computer Science, Advances in Cryptology.

[25] J. Zhou and D. Gollmann. Evidence and non-repudiation. *Journal of Network and Compter Applications*, 1997.