Evolution of Fair Non-repudiation with TTP

Jianying Zhou, Robert Deng, and Feng Bao

Kent Ridge Digital Labs
21 Heng Mui Keng Terrace
Singapore 119613
{jyzhou,deng,baofeng}@krdl.org.sg

Abstract. Non-repudiation turns out to be an increasingly important security service with the fast growth of electronic commerce on the Internet. Non-repudiation services protect the transacting parties against any false denial that a particular event or action has taken place, in which evidence will be generated, collected and maintained to enable dispute resolution. Meanwhile, fairness is a further desirable requirement such that neither party can gain an advantage by quitting prematurely or otherwise misbehaving during a transaction. In this paper, we survey the evolution of techniques and protocols that had been put forward to achieve fair non-repudiation with a (trusted) third party, and present a secure and efficient fair non-repudiation protocol.

Keywords: fair non-repudiation, trusted third party, secure electronic commerce

1 Introduction

Electronic transactions become a growing trend with the development of computer networks. On the other hand, dispute of transactions is a common problem that could jeopardise business. We imagine the following scenario.

A merchant A sells *electronic goods* M (e.g. softwares, videos, or digital publications) on the Internet. Suppose a customer B wants to buy M with his credit card. Typical disputes that may arise in such a transaction could be

- A claims that he has sent M to B while B denies receiving it;
- B claims that he received M (which is bogus or illegal) from A while A denies sending it.

In order to settle these disputes by a third party arbitrator, A and B need to present evidence to prove their own claims. Such evidence may be provided by non-repudiation services.

Non-repudiation services protect the transacting parties against any false denial that a particular event or action has taken place, in which evidence will be generated, collected and maintained to enable the settlement of disputes [21]. The basic non-repudiation services that address the above disputes are

J. Pieprzyk, R. Safavi-Naini, and J. Seberry (Eds.): ACISP'99, LNCS 1587, pp. 258–269, 1999.

- *Non-repudiation of Origin* (*NRO*) provides the recipient of a message with evidence of origin of the message which will protect against any attempt by the originator to falsely deny having sent the message.
- *Non-repudiation of Receipt* (*NRR*) provides the originator of a message with evidence of receipt of the message which will protect against any attempt by the recipient to falsely deny having received the message.

Generally speaking, non-repudiation can be achieved with basic security mechanisms such as digital signatures and notarisation. However, fairness may be a further desirable requirement. In the above transaction, the merchant A would like to get a receipt as evidence for payment claim when sending M to the customer B. On the other hand, the customer B will be reluctant to acknowledge the receipt before obtaining M. Fair non-repudiation was considered in the Draft International Standard ISO/IEC 13888 "Information technology - Security techniques - Non-repudiation". However, the mechanisms in the current version of this document [14,15,16] do not support fair non-repudiation and only have limited application [21].

A fair non-repudiation protocol should not give the originator of a message an advantage over the recipient, or vice versa. This paper investigates the evolution of techniques and protocols that had been put forward to achieve fair non-repudiation, and presents a secure and efficient fair non-repudiation protocol. The following general notation is used throughout the paper.

- X, Y: concatenation of two messages X and Y.
- $H(X)$: a one-way hash function of message X.
- $eK(X)$ and $dK(X)$: encryption and decryption of message X with key K.
- $sS_A(X)$: principal A's digital signature on message X with the private signature key S_A. The algorithm is assumed to be a 'signature with appendix', and the message is not recoverable from the signature.
- $A \rightarrow B : X$: principal A dispatches message X addressed to principal B.
- $A \leftrightarrow B : X$: principal A fetches message X from principal B using *"ftp get"* operation [18] or by some analogous means (e.g. using a Web browser).

2 Approaches for Fair Non-repudiation

The origin of a message will usually be verified by a digital signature appended by the originator. To obtain evidence of receipt, the originator requires the recipient to reply with some sort of acknowledgement. There are two possible reasons for such an acknowledgment not to arrive [20]:

- *The communication channel is unreliable*. Thus, a message may have been sent but failed to reach the recipient.
- *A communicating party does not play fair*. Thus, a dishonest party may abandon execution intentionally without following the rules of a protocol.

As a result, the recipient may repudiate receipt of a message even if it has received the message by falsely claiming the failure of the communication channel.

Definition 1. A non-repudiation protocol is *fair* if it provides the originator and the recipient with valid irrefutable evidence after completion of the protocol, without giving a party an advantage over the other party in any possible incomplete protocol runs [20].

Approaches for fair non-repudiation reported in the literature fall into two categories:

- *Gradual exchange protocols* [3,6,8,11,17] where two parties gradually disclose the expected items by many steps.
- *Third party protocols* [1,2,3,4,9,10,12,13,19,20,22] which make use of an on-line or off-line (trusted) third party.

The gradual exchange approach may have theoretical value but is too cumbersome for actual implementation because of the high computation and communication overheads. Moreover, fairness is based on the assumption of equal computational complexity, which makes sense only if the two parties have equal computing power, an often unrealistic and undesirable assumption [5]. Hence, recent research mainly focuses on the third party approach.

At the early stage, fair non-repudiation was achieved by the use of an *on-line* (trusted) third party *TTP*. As the use of *TTP* in fair non-repudiation protocols may cause the bottleneck problem, it is necessary to minimize the *TTP*'s involvement when designing efficient fair non-repudiation protocols. Such an attempt has been made in [20], where the *TTP* acts as a *light-weighted notary* rather than a delivery authority. However, the *TTP* still needs to be involved in each protocol run, though this might be necessary in some applications [22].

The *TTP*'s involvement is further reduced in [1,4,22], where transacting parties are willing to resolve communications problems between themselves and turn to the *TTP* only as a last recourse. However, only the risk-taking party (originator) is allowed to invoke the *TTP*, the responder may not know the final state of a protocol run in time. If a short time limit is imposed on a protocol run, the originator may not be quick enough to invoke the *TTP* for recovery thus the fairness will be destroyed.

The latest effort on fair non-repudiation was made by Asokan, Shoup and Waidner [2], which uses the *TTP* only in the case of exceptions and tolerates temporary failures in the communication channels to the *TTP*. In addition, it allows either party to unilaterally bring a protocol run to completion without losing fairness. However, some flaws and security weaknesses of their protocol have been pointed out in [23]:

- The protocol performance may degrade when transmitting large messages.

– The privacy of messages being transmitted may not be well protected.
– The non-repudiation evidence may not be publicly verifiable.

It is desirable to overcome these shortcomings while maintaining the merits of the protocol.

In this paper, we will use the protocols presented in [20,22] as examples to show the evolution of techniques for fair non-repudiation, and propose a secure and efficient fair non-repudiation protocol based on the ideas from [2,20].

3 Protocol A: Using Light-Weighted TTP

A fair non-repudiation protocol using light-weighted on-line *TTP* was proposed in [20], which supports non-repudiation of origin and non-repudiation of receipt while neither the originator nor the recipient can gain an advantage by quitting prematurely or otherwise misbehaving during a transaction. The main idea of this protocol is to split the definition of a message M into two parts, a commitment C and a key K. The commitment is sent from the originator A to the recipient B and then the key is lodged with the trusted third party *TTP*. Both A and B have to retrieve the confirmed key from the *TTP* as part of the non-repudiation evidence required in the settlement of a dispute. The notation below is used in the protocol description.

- M: message being sent from A to B.
- K: message key defined by A.
- $C = eK(M)$: commitment (ciphertext) for message M.
- $L = H(M, K)$: a unique label linking C and K.
- f_i $(i = 1, 2, \cdots)$: flags indicating the intended purpose of a signed message.
- $EOO\_C = sS_A(f_1, B, L, C)$: evidence of origin of C.
- $EOR\_C = sS_B(f_2, A, L, C)$: evidence of receipt of C.
- $sub\_K = sS_A(f_5, B, L, K)$: evidence of submission of K.
- $con\_K = sS_{TTP}(f_6, A, B, L, K)$: evidence of confirmation of K issued by the *TTP*.

The protocol is as follows.

$$
\begin{array}{ll}
1.\ A \rightarrow B: & f_1, B, L, C, EOO\_C \\
2.\ B \rightarrow A: & f_2, A, L, EOR\_C \\
3.\ A \rightarrow TTP: & f_5, B, L, K, sub\_K \\
4.\ B \leftrightarrow TTP: & f_6, A, B, L, K, con\_K \\
5.\ A \leftrightarrow TTP: & f_6, A, B, L, K, con\_K
\end{array}
$$

It is assumed that A, B, and the *TTP* either hold the relevant public key certificates, or are able to retrieve them from a X.509 directory service [7]. It is further assumed that the communication channels linking the *TTP* and each transacting party (A and B) are *resilient*.

Definition 2. A communication channel is *resilient* if a message inserted into such a channel will eventually be delivered.

We examine the protocol step by step.

1. A first sends C and EOO_C to B. There is no breach of fairness if the protocol stops at Step 1 since C is incomprehensible without K.

   B needs to verify EOO_C and save EOO_C as evidence of origin of C before proceeding to the next step.
2. B has to send EOR_C to A if B wants to get K and con_K from the TTP at Step 4. There is no breach of fairness if the protocol stops at Step 2 since EOR_C can only be used to prove receipt of C rather than receipt of M.

   A needs to verify EOR_C and save EOR_C as evidence of receipt of C before proceeding to the next step.
3. A has to send K and sub_K to the TTP if A wants to get con_K from the TTP at Step 5. B could obtain K by eavesdropping, and thereby the message M, before K is lodged with the TTP. As we assume that the communication channel between A and the TTP is resilient, A will eventually be able to send K and sub_K to the TTP in exchange for con_K.

   After receiving K and sub_K from A, the TTP will generate con_K and store the tuple (f_6, A, B, L, K, con_K) in a directory which is accessible (read only) to the public. The second component in the tuple indicates the key supplier which is authenticated by the TTP with sub_K. Intruders cannot mount a denial-of-service attack by sending bogus keys to the TTP as this will not generate entries $(f_6, A, \cdots)$ in the directory.
4. B fetches K and con_K from the TTP. B obtains M by computing $M = dK(C)$, and saves con_K as evidence to prove that K originated from A.

   As we assume that the communication channel between B and the TTP is resilient, B can therefore always retrieve K and con_K. B will lose the dispute over receipt of M even if B does not fetch K after it becomes publicly available.
5. A fetches con_K from the TTP, and saves it as evidence to prove that K is available to B.

The above analysis demonstrates that if and only if A has sent C to B and K to the TTP, will A have evidence (EOR_C, con_K) and B have evidence (EOO_C, con_K).

Label L plays an important role in the establishment of a unique link between C and K. Once L and C have been committed in EOO_C and EOR_C, it is computationally hard to find $K' \neq K$ satisfying $L = H(M, K) = H(M, K')$ while $M = dK(C) = dK'(C)$.

If A denies origin of M, B can present evidence EOO_C and con_K plus M, C, K to a third party arbitrator. The arbitrator will check

– A's signature $EOO\_C = sS_A(f_1, B, L, C)$

- *TTP*'s signature $con\_K = sS_{TTP}(f_6, A, B, L, K)$
- $L = H(M, K)$
- $M = dK(C)$

If the first two checks are positive, the arbitrator believes that C and K originated from A. If the last two checks are also positive, the arbitrator will conclude that C and K are uniquely linked by L, and M is the message represented by C and K from A.

If B denies receipt of M, A can present evidence EOR_C and con_K plus M, C, K to the arbitrator. The arbitrator will make similar checks as above.

Unlike other fair non-repudiation protocols which use the *on-line* trusted third party as a delivery authority, the trusted third party in this protocol acts as a *light-weighted notary* which only notarises message keys by request and provides directory services accessible to the public. This has two advantages.

- The trusted third party only deals with keys, which in general will be shorter than the full messages.
- The onus is now on the originator and the recipient to retrieve the key, while a delivery authority would have to keep resending messages until the receiver acknowledges the message.

4 Protocol B: Using Offline TTP

In Section 3, the trusted third party's work load has been significantly reduced in the protocol, where the *TTP* only needs to notarise message keys by request and provides directory services. Such a protocol is appropriate in applications where notarisation of keys is desirable [1], or where the participants and the communications infrastructure are so unreliable that participants prefer to rely on the *TTP* to facilitate transactions.

An efficient fair non-repudiation protocol was proposed in [22], which further reduced the trusted third party's active involvement when the two parties are willing to resolve communications problems between themselves and want to turn to the *TTP* only as a last recourse. In the normal case, the originator A and the recipient B will exchange messages and non-repudiation evidence directly. The *TTP* will be invoked only in the error-recovery phase initiated by A when A cannot get the expected evidence from B.

Besides the notation used in Section 3, the following additional notation is used in the protocol description.

[1] When disputes relate to the time of message transfer, the originator and the recipient may need evidence about the time of sending and receiving a message besides evidence of origin and receipt. The *TTP* can time-stamp evidence con_K to identify when the message key, and thus the message, was made available.

- $EOO\_K = sS_A(f_3, B, L, K)$: evidence of origin of K.
- $EOR\_K = sS_B(f_4, A, L, K)$: evidence of receipt of K.

The protocol in the normal case is as follows.

$$\begin{aligned}
&1.\ A \rightarrow B : f_1, B, L, C, EOO_C \\
&2.\ B \rightarrow A : f_2, A, L, EOR_C \\
&3.\ A \rightarrow B : f_3, B, L, K, EOO_K \\
&4.\ B \rightarrow A : f_4, A, L, EOR_K
\end{aligned}$$

If A does not send message 3, the protocol ends without disputes. If A cannot get message 4 from B after sending message 3 (either because B did not receive message 3 or because B does not want to acknowledge it), A may initiate the following recovery phase, which is the same as Steps 3 to 5 of the protocol in Section 3.

$$\begin{aligned}
&3'.\ A \rightarrow TTP : f_5, B, L, K, sub_K \\
&4'.\ B \leftrightarrow TTP : f_6, A, B, L, K, con_K \\
&5'.\ A \leftrightarrow TTP : f_6, A, B, L, K, con_K
\end{aligned}$$

If the protocol run is complete, the originator A will hold non-repudiation evidence EOR_C and EOR_K, and the recipient B will hold EOO_C and EOO_K. Otherwise, A needs to rectify the unfair situation by initiating the recovery phase so that non-repudiation evidence con_K will be available to both A and B.

If disputes arise, A can use (EOR_C, EOR_K) or (EOR_C, con_K) as non-repudiation evidence to prove that B received M; B can use (EOO_C, EOO_K) or (EOO_C, con_K) as non-repudiation evidence to prove that M originated from A.

This protocol will be efficient in an environment where two parties usually play fair in a protocol run. Although the recipient B is temporarily in an advantageous position after Step 3, fairness can be retained by ensuring the success of the recovery phase, which relies on the assumption that the communication channels between the TTP and the participants A, B are resilient.

In practice, however, a time limit for a protocol run may have to be set so that both parties can terminate an expired protocol run safely. Then the choice of time limit in the above protocol becomes critical because that may affect the protocol fairness. If A cannot get message 4 from B, A has to rely on a successful recovery phase to rectify the unfair situation. A needs to submit the message key to the TTP in time since the TTP will not confirm A's submission once the protocol run expires. However, as we only assume that the communication channels are not permanently broken, A may not be sure that the TTP can receive its submission in time. Therefore, A has to choose the time limit big enough. This means that B may not know the final state of a protocol run in

time, which is obviously unfavourable to B [2]. If B does not receive the message key from A by the deadline, B has to retrieve it from the *TTP*, or abandon the protocol run with a notification from the *TTP*.

5 Protocol C: Autonomous with Offline TTP

Here we present an autonomous fair non-repudiation protocol using off-line *TTP*, which is mainly based on the ideas from [2,20].

Definition 3. A fair non-repudiation protocol is *autonomous* if either transacting party can unilaterally bring a transaction to completion without losing fairness.

We split the definition of a message M into two parts, a commitment C and a key K. In the normal case, the originator A sends (C, K) (plus evidence of origin) to the recipient B in exchange for evidence of receipt without any involvement of the *TTP*. If there is something wrong in the middle of a transaction, either A or B can unilaterally bring the transaction to completion with the help from the *TTP*. The *TTP* only needs to notarise and/or deliver the message key K by request, which is usually much shorter than the whole message M. The notation below is used in the description of our protocol.

- M: message being sent from A to B.
- K: message key defined by A.
- $C = eK(M)$: commitment (cipher text) for message M.
- $L = H(M, K)$: a unique label linking C and K.
- f_i $(i = 1, 2, \cdots)$: flags indicating the intended purpose of a signed message.
- $EOO\_C = sS_A(f_1, B, L, C)$: evidence of origin of C.
- $EOR\_C = sS_B(f_2, A, L, EOO\_C)$: evidence of receipt of C.
- $EOO\_K = sS_A(f_3, B, L, K)$: evidence of origin of K.
- $EOR\_K = sS_B(f_4, A, L, EOO\_K)$: evidence of receipt of K.
- $sub\_K = sS_A(f_5, B, L, K, TTP, EOO\_C)$: evidence of submission of K to the *TTP*.
- $con\_K = sS_{TTP}(f_6, A, B, L, K)$: evidence of confirmation of K issued by the *TTP*.
- $abort = sS_{TTP}(f_8, A, B, L)$: evidence of abortion.
- P_{TTP}: the *TTP*'s public encryption key.

Our protocol has three sub-protocols: *exchange*, *abort*, and *resolve*. We assume that the communication channels between the *TTP* and each transacting party (A and B) are *resilient*. We also assume that the communication channel between A and B is *confidential* if the two parties want to exchange messages

[2] This problem does not exist in the protocol described in Section 3. An arbitrary length of time limit can be set for a protocol run as long as the message key is protected from disclosure to B when A submits it to the *TTP* for confirmation.

secretly. The *exchange* sub-protocol is as follows.

1. $A \rightarrow B : f_1, f_5, B, L, C, TTP, eP_{TTP}(K), EOO\_C, sub\_K$
   IF B gives up **THEN** quit **ELSE**
2. $B \rightarrow A : f_2, A, L, EOR\_C$
   IF A gives up **THEN** *abort* **ELSE**
3. $A \rightarrow B : f_3, B, L, K, EOO\_K$
   IF B gives up **THEN** *resolve* **ELSE**
4. $B \rightarrow A : f_4, A, L, EOR\_K$
   IF A gives up **THEN** *resolve*

The *abort* sub-protocol is as follows.

1. $A \rightarrow TTP : f_7, B, L, sS_A(f_7, B, L)$
   IF resolved **THEN**
2. $TTP \rightarrow A : f_2, f_6, A, B, L, K, con\_K, EOR\_C$
   ELSE
3. $TTP \rightarrow A : f_8, A, B, L, abort$

The *resolve* sub-protocol is as follows, where the initiator U is either A or B.

1. $U \rightarrow TTP : f_2, f_5, A, B, L, TTP, eP_{TTP}(K), sub\_K, EOO\_C, EOR\_C$
   IF aborted **THEN**
2. $TTP \rightarrow U : f_8, A, B, L, abort$
   ELSE
3. $TTP \rightarrow U : f_2, f_6, A, B, L, K, con\_K, EOR\_C$

If the *exchange* sub-protocol is executed successfully, B will receive C and K and thus $M = dK(C)$ together with evidence of origin (EOO_C, EOO_K). Meanwhile, A will receive evidence of receipt (EOR_C, EOR_K).

B can simply quit the transaction without losing fairness before sending EOR_C to A. Otherwise, B has to run the *resolve* sub-protocol to force a successful termination. Similarly, A can run the *abort* sub-protocol to quit the transaction without losing fairness before sending K and EOO_K to B. Otherwise, A has to run the *resolve* sub-protocol to force a successful termination.

The *resolve* sub-protocol can be initiated either by A or by B. When the TTP receives such a request, the TTP will first check the status of a transaction identified by (A, B, L) uniquely. If the transaction has been aborted by A, the TTP will return the *abort* token. If the transaction has already been resolved, the TTP will deliver the tuple $(f_2, f_6, A, B, L, K, con\_K, EOR\_C)$ to the current initiator of the *resolve* sub-protocol. Otherwise, the TTP will

- check that EOR_C is consistent with sub_K in terms of L and EOO_C,
- generate evidence con_K,
- deliver the tuple $(f_2, f_6, A, B, L, K, con\_K, EOR\_C)$ to the current initiator,
- set the status of the transaction *resolved*.

The third component in the tuple indicates the key supplier which is authenticated by the *TTP* with *sub_K*. Evidence *con_K* can be used to prove that

- a transaction identified by (A, B, L) has been resolved successfully,
- the message key K originated from A, and
- the message key is available from the *TTP* by request.

The time limit on maintaining the status of a transaction (*resolved* or *aborted*) by the *TTP* will be defined in the non-repudiation policy, which can be reasonably long enough (mainly depending on the *TTP*'s storage capability) so that both transacting parties are deemed to be able to consult the *TTP* within such a time limit to force a successful termination of a transaction when it is necessary.

If disputes arise, A can use evidence (*EOR_C*, *EOR_K*) or (*EOR_C*, *con_K*) to prove that B received the message M, B can use evidence (*EOO_C*, *EOO_K*) or (*EOO_C*, *con_K*) to prove that A sent the message M.

In comparison with the protocol in [2], our protocol has the following merits.

- The *TTP*'s overhead will not increase when transmitting a large message M.
- The content of the message M need not be disclosed to any outsiders including the *TTP*.
- The evidence is publicly verifiable without any restrictions on the types of signature and encryption algorithms.

Therefore, our protocol is more secure and efficient both at the stage of exchange and at the stage of dispute resolution.

6 Conclusion

Fair non-repudiation protocols can be constructed in two ways, by gradual exchange of the expected items, or by invoking the services of a (trusted) third party. The major defects of the first approach are

- high computation and communication overheads, and
- strong assumption on transacting partys' equal computing power for fairness.

Hence, recent research mainly focuses on the second approach. As the trusted third party may become a system bottleneck, a critical issue is how to minimize the trusted third party's involvement in fair non-repudiation protocols.

There are three major advances on the research along this direction. Early efforts were to make use of a light-weighted on-line trusted third party (e.g. a fair non-repudiation protocol in [20]). Later on, an off-line trusted third party was employed in fair non-repudiation protocols (e.g. in [1,4,22]) but fairness may be destroyed when a time limit is imposed on a protocol run. The most recent

advance is to allow either transacting party to unilaterally bring a protocol run to completion without losing fairness with the assistance of an off-line trusted third party [2]. This paper presented a more secure and efficient fair non-repudiation protocol based on the ideas from [2,20]. An open problem is how to achieve fair non-repudiation without relying on the assumption of *resilient* communication channels between an off-line trusted third party and each transacting party.

Acknowledgements

We thank the anonymous referees for valuable comments on the draft of this paper.

References

1. N. Asokan, M. Schunter and M. Waidner. *Optimistic protocols for fair exchange*. Proceedings of 4th ACM Conference on Computer and Communications Security, pages 7–17, Zurich, Switzerland, April 1997.
2. N. Asokan, V. Shoup and M. Waidner. *Asynchronous protocols for optimistic fair exchange*. Proceedings of 1998 IEEE Symposium on Security and Privacy, pages 86–99, Oakland, California, May 1998.
3. A. Bahreman and J. D. Tygar. *Certified electronic mail*. Proceedings of the Internet Society Symposium on Network and Distributed System Security, pages 3–19, San Diego, California, February 1994.
4. F. Bao, R. H. Deng and W. Mao. *Efficient and practical fair exchange protocols with off-line TTP*. Proceedings of 1998 IEEE Symposium on Security and Privacy, pages 77–85, Oakland, California, May 1998.
5. M. Ben-Or, O. Goldreich, S. Micali and R. Rivest. *A fair protocol for signing contracts*. IEEE Transactions on Information Theory, IT-36(1):40–46, January 1990.
6. E. F. Brickell, D. Chaum, I. B. Damgard and J. van de Graaf. *Gradual and verifiable release of a secret*. Lecture Notes in Computer Science 293, Advances in Cryptology: Proceedings of Crypto'87, pages 156–166, Santa Barbara, California, August 1987.
7. CCITT. *Recommendation X.509: The directory – Authentication framework*. November 1988.
8. R. Cleve. *Controlled gradual disclosure schemes for random bits and their applications*. Lecture Notes in Computer Science 435, Advances in Cryptology: Proceedings of Crypto'89, pages 573–588, Santa Barbara, California, August 1989.
9. T. Coffey and P. Saidha. *Non-repudiation with mandatory proof of receipt*. Computer Communication Review, 26(1):6–17, January 1996.
10. B. Cox, J. D. Tygar and M. Sirbu. *NetBill security and transaction protocol*. Proceedings of the First USENIX Workshop on Electronic Commerce, pages 77–88, July 1995.
11. I. B. Damgard. *Practical and provably secure release of a secret and exchange of signatures*. Lecture Notes in Computer Science 765, Advances in Cryptology: Proceedings of Eurocrypt'93, pages 200–217, Lofthus, Norway, May 1993.
12. R. H. Deng, L. Gong, A. A. Lazar and W. Wang. *Practical protocols for certified electronic mail*. Journal of Network and Systems Management, 4(3):279–297, 1996.

13. M. Franklin and M. Reiter. *Fair exchange with a semi-trusted third party*. Proceedings of 4th ACM Conference on Computer and Communications Security, pages 1–6, Zurich, Switzerland, April 1997.
14. ISO/IEC 13888-1. *Information technology - Security techniques - Non-repudiation - Part 1: General*. ISO/IEC, 1997.
15. ISO/IEC 13888-2. *Information technology - Security techniques - Non-repudiation - Part 2: Mechanisms using symmetric techniques*. ISO/IEC, 1998.
16. ISO/IEC 13888-3. *Information technology - Security techniques - Non-repudiation - Part 3: Mechanisms using asymmetric techniques*. ISO/IEC, 1997.
17. T. Okamoto and K. Ohta. *How to simultaneously exchange secrets by general assumptions*. Proceedings of 2nd ACM Conference on Computer and Communications Security, pages 184–192, Fairfax, Virginia, November 1994.
18. J. B. Postel and J. K. Reynolds. *File transfer protocol*. RFC 959, October 1985.
19. N. Zhang and Q. Shi. *Achieving non-repudiation of receipt*. The Computer Journal, 39(10):844–853, 1996.
20. J. Zhou and D. Gollmann. *A fair non-repudiation protocol*. Proceedings of 1996 IEEE Symposium on Security and Privacy, pages 55–61, Oakland, California, May 1996.
21. J. Zhou. *Non-repudiation*. PhD Thesis, University of London, December 1996.
22. J. Zhou and D. Gollmann. *An efficient non-repudiation protocol*. Proceedings of 10th IEEE Computer Security Foundations Workshop, pages 126–132, Rockport, Massachusetts, June 1997.
23. J. Zhou, R. H. Deng and F. Bao. *Some remarks on a fair exchange protocol*. (manuscript)