An Efficient Non-repudiation Protocol

Jianying Zhou
Dept. of Information Systems & Computer Science
National University of Singapore
10 Kent Ridge Crescent
Singapore 119260
email: zhoujy@iscs.nus.edu.sg

Dieter Gollmann
Department of Computer Science
Royal Holloway, University of London
Egham, Surrey TW20 0EX
United Kingdom
email: D.Gollmann@rhbnc.ac.uk

Abstract

Fairness may be a desirable property of a non-repudiation service. Protocols can achieve fairness through the involvement of a trusted third party but the extent of the trusted third party's involvement can vary between protocols. Hence, one of the goals of designing an efficient non-repudiation protocol is to reduce the work load of the trusted third party. In this paper we present a variant of our fair non-repudiation protocol [15], where the trusted third party will be involved only in case that one party cannot obtain the expected non-repudiation evidence from the other party. This variant is efficient in an environment where the two parties are likely to resolve communications problems between themselves.

1. Introduction

Computer networks are an efficient means for providing electronic services but reliance on electronic communications makes information also more vulnerable. Network security requirements have emerged in virtually all network application environments, including banking, electronic trading, government, public telecommunications carriers, and corporate/private networks. The objective of network security is to protect the confidentiality, integrity, authenticity, accountability, availability, and legitimate use of information according to application requirements. Non-repudiation is one of the essential security services in computer networks, being mainly applied in message handling systems and electronic commerce [17, 18]. The standardisation of non-repudiation mechanisms is in progress [8, 9, 10], and some problems have been pointed out in [16, 18].

Non-repudiation services protect the parties involved in a transaction against the other party denying that a particular event or action took place. They collect irrefutable evidence to support the resolution of any such disagreement. The basic non-repudiation services required to establish the accountability of parties involved in the transaction are non-repudiation of origin and non-repudiation of receipt.

- *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.

Fairness may be a desirable property of a non-repudiation service. Protocols can achieve fairness through the involvement of a trusted third party but the extent of the trusted third party's involvement can vary between protocols. Hence, one of the goals of designing an efficient non-repudiation protocol is to reduce the work load of the trusted third party. In this paper we present a variant of our fair non-repudiation protocol [15], where the trusted third party will be involved only in case that one party cannot obtain the expected non-repudiation evidence from the other party.

2. Fair Non-repudiation

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

- The communication channel is *unreliable*. The message may have been sent, but sending a message does not imply delivery of the message.

- A communicating party does *not play fair*. A communicating party plays fair if it follows the rules of the protocol and does not abandon execution intentionally.

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. We define a non-repudiation protocol to be *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 at any stage of the protocol run [15].

Generally speaking, fair non-repudiation protocols can be constructed in the following two ways:

- The originator and the recipient exchange the message and/or non-repudiation evidence simultaneously;
- Trusted third parties assist in fair exchange of the message and/or non-repudiation evidence.

In the first approach, e.g. [6, 12], fairness is achieved by the *gradual release of secrets* over many rounds: during each round, some knowledge about the message and/or evidence is revealed. If either party stops before the protocol run is complete, both parties are left with comparable knowledge. This approach seems to be too cumbersome and inefficient for actual implementation. 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 [3].

A few fair non-repudiation protocols based on the active involvement of trusted third parties exist [2, 4, 5, 7, 15]. From the investigation in [18], we found that they differ in

- the degree to which a trusted third party has to be involved in a protocol run, and
- the assumptions on the availability of communications links.

We can classify the involvement of trusted third parties into three levels [17]:

- *Non-repudiation with an in-line trusted third party* – A trusted third party acts as an intermediary between the originator and the recipient and intervenes directly in a non-repudiation service, e.g. a delivery authority.
- *Non-repudiation with an on-line trusted third party* – A trusted third party is actively involved in every instance of a non-repudiation service, e.g. a notary.
- *Non-repudiation with an off-line trusted third party* – A trusted third party supports non-repudiation without being involved in each instance of a service, e.g. a certification authority.

The fair non-repudiation protocol presented in [15] only needs an on-line low weight notary and a weak assumption on the availability of communications links. In this paper, we try to improve its efficiency for the situation where two parties usually play fair and do not attempt to cheat by further reducing the active involvement of the trusted third party. We employ the following notation to represent messages and protocols.

- X, Y: concatenation of two messages X and Y, *in the order specified*.
- $H(X)$: a one-way hash function of message X.
- $eK(X)$: encryption of message X with key K.
- V_A and S_A: the public and private key of principal A.
- $sK(X)$: digital signature of message X with the private key K.
- $A \rightarrow B : X$: principal A sends message X to principal B.
- $A \leftrightarrow B : X$: principal A fetches message X from principal B using *"ftp get"* operation [13].

3. A Low-weight Notary

A fair non-repudiation protocol is proposed in [15], where the originator and the recipient communicate directly with significantly reduced involvement of a *TTP* acting as a `low weight notary'. The main idea of this protocol is to split the definition of the message 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 *TTP*. Both A and B have to retrieve the confirmed key from the *TTP* as part of the non-repudiation evidence required in a dispute. We assume that even in case of network failures, both parties will eventually be able to retrieve the key from the *TTP*. The notation in the protocol description is as follows.

- M: message sent from A to B.
- C: commitment (ciphertext) for message M, e.g. M encrypted under a key K.
- K: message key defined by A.
- L: a unique label chosen by A to link all messages of a particular protocol run.

- f: a flag indicating the intended purpose of a (signed) message.
- *EOO* $= sS_A(f_{EOO}, B, L, C)$: evidence of origin of C.
- *EOR* $= sS_B(f_{EOR}, A, L, C)$: evidence of receipt of C.
- *sub_K* $= sS_A(f_{SUB}, B, L, K)$: evidence of submission of K.
- *con_K* $= sS_{TTP}(f_{CON}, A, B, L, K)$: evidence of confirmation of K issued by *TTP*.

The protocol is as follows:

$$
\begin{array}{ll}
1.\ A \rightarrow B: & f_{EOO}, B, L, C, \textit{EOO} \\
2.\ B \rightarrow A: & f_{EOR}, A, L, \textit{EOR} \\
3.\ A \rightarrow \textit{TTP}: & f_{SUB}, B, L, K, \textit{sub\_K} \\
4.\ B \leftrightarrow \textit{TTP}: & f_{CON}, A, B, L, K, \textit{con\_K} \\
5.\ A \leftrightarrow \textit{TTP}: & f_{CON}, A, B, L, K, \textit{con\_K}
\end{array}
$$

In the above protocol, B could obtain K by eavesdropping, and thereby the message M by deciphering C under K, before K is lodged with the *TTP*. We assume that B is not in a position to block *sub_K* permanently and thus prevent A from obtaining evidence of receipt.

After receiving *sub_K* and K from A, the *TTP* will generate *con_K* and store the tuple

$$(f_{CON}, A, B, L, K, \textit{con\_K})$$

in a directory which is accessible (read only) to the public. The second component in the tuple, identifying the key supplier, corresponds to the entity associated with the public verification key V_A. The link between A and B, L, K is authenticated by A's signature. The key supplier will be regarded as the originator of this protocol run. Intruders cannot mount a denial-of-service attack by sending bogus keys to the *TTP* as this will not generate entries for A in the directory. To facilitate a unique interpretation of message M, A is not allowed to submit more than one key with the same label and recipient. So the *TTP* needs to check existing labels for entries $(f_{CON}, A, B, \cdots)$ before confirming A's submission, and a new key K' for A, B, L will not be accepted by the *TTP*.

This protocol has the following properties:

- The protocol does not depend on the reliability of a communication channel or on the communicating parties playing fair. It only needs a weak assumption that the communication channels are not permanently broken.
- At no point of the protocol run does either participant have an advantage. Any party can abort the protocol run without disputes before the message key is notarized by the *TTP*; no party can repudiate the message transfer after the message key has been notarized by the *TTP*.
- The trusted third party's involvement is significantly reduced. The *TTP* only notarizes message keys by request and provides directory services.

In this protocol, the trusted third party has to be involved in every protocol run. Such a protocol is appropriate in applications where notarisation of keys is desirable, where strict fairness is essential, or where the participants and the communications infrastructure are so unreliable that participants prefer to rely on a *TTP* rather than sort out communications problems between themselves.

4. Reducing the TTP's Involvement

4.1. A Variant Protocol

We can further reduce the trusted third party's active involvement in the above non-repudiation protocol when the two parties are willing to resolve communications problems between themselves and want to turn to a *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. *TTP* will be invoked only in the error-recovery phase initiated by A when A cannot get the expected evidence from B. The protocol can be changed as follows [17].

$$
\begin{array}{l}
\textit{EOO} = sS_A(f_{EOO}, B, L, C) \\
\textit{EOR} = sS_B(f_{EOR}, A, L, C) \\
\textit{EOO\_K} = sS_A(f_{EOO\_K}, B, L, K) \\
\textit{EOR\_K} = sS_B(f_{EOR\_K}, A, L, K)
\end{array}
$$

$$
\begin{array}{ll}
1.\ A \rightarrow B: & f_{EOO}, B, L, C, \textit{EOO} \\
2.\ B \rightarrow A: & f_{EOR}, A, L, \textit{EOR} \\
3.\ A \rightarrow B: & f_{EOO\_K}, B, L, K, \textit{EOO\_K} \\
4.\ B \rightarrow A: & f_{EOR\_K}, A, L, \textit{EOR\_K}
\end{array}
$$

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.

$$
\begin{array}{ll}
3'.\ A \rightarrow \textit{TTP}: & f_{SUB}, B, L, K, \textit{sub\_K} \\
4'.\ B \leftrightarrow \textit{TTP}: & f_{CON}, A, B, L, K, \textit{con\_K} \\
5'.\ A \leftrightarrow \textit{TTP}: & f_{CON}, A, B, L, K, \textit{con\_K}
\end{array}
$$

If the protocol run is complete, the originator A will hold non-repudiation evidence *EOR* and *EOR_K*, and the recipient B will hold *EOO* 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 A and B.

This variant protocol could be efficient in an environment where two parties usually play fair in a protocol run. However, as B will be in an advantageous position after Step 3, this requires guaranteed success of the recovery phase if A cannot get message 4 from B, which relies on the assumption that the communication channels between the *TTP* and the participants A, B are not permanently broken.

4.2. Weak Fairness

In the variant protocol, after the recipient B acknowledges A's commitment C at Step 2, B has to wait for A's message key K. B may receive it from A directly, or B has to retrieve it from the *TTP* if A has submitted it to the *TTP*. It is desirable that B needs to know when it should retrieve K from the *TTP* if B did not receive K from A, i.e. there should be a deadline after which the state of message transfer can be determined.

In the protocol in Section 3, it is easy to achieve the above goal without affecting the protocol's fairness. Suppose T is the deadline that the message key K should be available to the public, and K will never appear in the directory after the deadline. The protocol can be changed as follows:

$$
\begin{aligned}
\textit{EOO} &= sS_A(f_{EOO}, B, L, T, C)\\
\textit{EOR} &= sS_B(f_{EOR}, A, L, T, C)\\
\textit{sub\_K} &= sS_A(f_{SUB}, B, L, T, K)\\
\textit{con\_K} &= sS_{TTP}(f_{CON}, A, B, L, T, K)
\end{aligned}
$$

$$
\begin{aligned}
&1.\ A \rightarrow B: && f_{EOO}, B, L, T, C, \textit{EOO}\\
&2.\ B \rightarrow A: && f_{EOR}, A, L, \textit{EOR}\\
&3.\ A \rightarrow \textit{TTP}: && f_{SUB}, B, L, T, K, \textit{sub\_K}\\
&4.\ B \leftrightarrow \textit{TTP}: && f_{CON}, A, B, L, K, \textit{con\_K}\\
&5.\ A \leftrightarrow \textit{TTP}: && f_{CON}, A, B, L, K, \textit{con\_K}
\end{aligned}
$$

The deadline T could be chosen by A but refers to *TTP*'s clock. If B does not agree with this deadline, B can end the protocol at Step 2. If there is a discrepancy in the deadline T appearing in non-repudiation evidence *EOO*, *EOR* and *con_K*, a claim will not succeed. A should check T in *EOR* before submitting K to the *TTP*, otherwise A may lose a dispute. If A does not submit K in time, the *TTP* will not confirm A's submission after T, and the protocol ends without disputes. (Here we assume that B is not in a position to obtain K by eavesdropping, otherwise message 3 should be protected by encryption.)

In the variant protocol in Section 4.1, however, the situation is different. If a deadline T is imposed, fairness may be broken in some cases. The variant protocol can be changed as follows:

$$
\begin{aligned}
\textit{EOO} &= sS_A(f_{EOO}, B, L, T, C)\\
\textit{EOR} &= sS_B(f_{EOR}, A, L, T, C)\\
\textit{EOO\_K} &= sS_A(f_{EOO\_K}, B, L, K)\\
\textit{EOR\_K} &= sS_B(f_{EOR\_K}, A, L, K)
\end{aligned}
$$

$$
\begin{aligned}
&1.\ A \rightarrow B: && f_{EOO}, B, L, T, C, \textit{EOO}\\
&2.\ B \rightarrow A: && f_{EOR}, A, L, \textit{EOR}\\
&3.\ A \rightarrow B: && f_{EOO\_K}, B, L, K, \textit{EOO\_K}\\
&4.\ B \rightarrow A: && f_{EOR\_K}, A, L, \textit{EOR\_K}
\end{aligned}
$$

If A cannot get message 4 from B, A has to initiate the recovery phase, which is the same as Steps 3 to 5 of the protocol at the beginning of Section 4.2. The problem is that if A does not submit K in time, the *TTP* will not confirm A's submission after T. However, as A has already sent K to B at Step 3, B is in an advantageous position. Since we only assume that the communication channels are not permanently broken, A is not sure how the *TTP* can receive its submission in time. In practice, A may choose T big enough and only send K to B when it has sufficient time to initiate the recovery phase. Therefore, the variant protocol can only provide weak fairness when a deadline for a protocol run is imposed.

4.3. The Choice of Labels

Labels have to be unique to create the link between commitment and key. In the protocol in Section 3, the originator A can choose labels which are independent of the messages. Then the message M may not be defined until A submits K to the *TTP* at Step 3 of that protocol. We require the *TTP* to check that the keys provided to it do not overwrite existing entries in the public directory, i.e. the *TTP* must check that the same label is not used twice by a pair (A, B).

In the variant protocol in Section 4.1, however, A must not be allowed to choose labels which are independent of the messages. A may send K to B at Step 3 and submit K' to the *TTP* at Step $3'$ with the same label and recipient. B may not be aware that there is another key K' confirmed by the *TTP* after B has already received K from A and does not expect to retrieve the key from the *TTP*. This will cause serious problems if there is a time limit on accessing confirmed message keys from the *TTP* [1]. In this case, A can

[1] This is a practical requirement since we cannot ask the *TTP* to keep all message keys in the public directory forever.

retrieve the confirmed key K' from the *TTP* within the time limit and use it to prove that B has received another message $M' = dK'(C)$ but actually B did not retrieve K'.

To facilitate a unique interpretation of message M, the label L in the variant protocol should be chosen as $L = H(M)$ where H is a one-way hash function. Then the message is already defined in Step 1 when A generates *EOO* which includes L $(= H(M))$ and C $(= eK(M))$. A key K will be regarded as valid only if the check of $L = H(dK(C))$ is successful. It is computationally difficult to find $M' = dK'(C)$ which satisfies $L = H(M')$. Such a definition of the label will further simplify the *TTP*'s involvement by omitting its check whether A submitted different keys for the same label and recipient as no party can win a dispute if A releases a wrong key (either to B at Step 3 or to the *TTP* at Step $3'$). We will further discuss dispute resolution in the next section.

As a label $H(M)$ may leak information about M when M belongs to a small message space, this may give B a chance to decide when to continue with a protocol run at Step 2, violating our fairness condition. To avoid this situation, a label $H(M, K)$ could be used instead.

4.4. Dispute Resolution

Disputes can arise over the origin and receipt of a message M. In the first case, B claims to have received M from A while A denies having sent M to B. In the second case, A claims having sent M to B while B denies having received M. These disputes can be resolved by a judge who evaluates the evidence held by the participants and determines whether receipt or origin are as claimed.

Repudiation of Origin

If B claims that it received M from A, the judge will require B to provide M, C, K, L, and the non-repudiation evidence *EOO* and *EOO_K*/con_K. If B cannot provide this evidence or one of the following checks fails, B's claim will be judged invalid.

1. The judge checks that *EOO* is A's signature on (f_{EOO}, B, L, C).
2. The judge checks that *EOO_K* is A's signature on (f_{EOO_K}, B, L, K) if B provides *EOO_K*, or checks that con_K is *TTP*'s signature on (f_{CON}, A, B, L, K) if B provides *con_K*.
3. The judge checks $L = H(dK(C))$.
4. The judge checks $M = dK(C)$.

If the first check is positive, the judge will assume that A had sent C to B as its commitment for the message with label L. If the second check is positive, the judge will assume that either A had sent the key K with label L to B, or the entry for the key submitted by A was made in the *TTP*'s directory. Here, the judge trusts the *TTP* to generate valid evidence. If the third check is positive, the judge will assume that the label L is a valid label linking C and K. If the final check is positive, the judge will uphold B's claim.

Repudiation of Receipt

If A claims that B had received M, the judge will require A to provide M, C, K, L, and the non-repudiation evidence *EOR* and *EOR_K*/con_K. If A cannot provide this evidence or one of the following checks fails, A's claim will be judged invalid.

1. The judge checks that *EOR* is B's signature on (f_{EOR}, A, L, C).
2. The judge checks that *EOR_K* is B's signature on (f_{EOR_K}, A, L, K) if A provides *EOR_K*, or checks that con_K is *TTP*'s signature on (f_{CON}, A, B, L, K) if A provides *con_K*.
3. The judge checks $L = H(dK(C))$.
4. The judge checks $M = dK(C)$.

If the first check is positive, the judge will assume that B had received C with label L and is committed to retrieving the key K with the same label from the *TTP* if B did not receive it from A. If the second check is positive, the judge will assume that either B had received the key K with label L from A, or the entry for the key submitted by A was made in the *TTP*'s directory and B is able to retrieve the key K from the *TTP*. If the third check is positive, the judge will assume that the label L is a valid label linking C and K. If the final check is positive, the judge will uphold A's claim.

5. Comparison with Related Work

Asokan et al. proposed a generic protocol for fair exchange where the trusted third party is involved only in the presence of faults or in the case of dishonest players who do not follow the protocol [1]. The basic idea of the protocol is that the originator A and the recipient B start by promising each other an exchange of items (Phase 1), and then proceed to exchange the items along with non-repudiation evidence (Phase 2). If there is something wrong after Phase 1, an error-recovery phase will be initiated and the initiator (A or B) will send the *TTP* all the messages previously exchanged with the other party. Here we analyse the protocol applied to certified mail. The following notation is used to describe the protocol.

- M: mail item to be sent from A to B.
- T: active-time limit up to which a run of the protocol can remain active.
- n_A and n_B: random numbers chosen by A and B respectively, and released later to form a part of non-repudiation evidence. Their main purpose is to avoid the need for additional signatures.
- $EOO' = sS_A(TTP, B, T, H(M, n_A), H(n_A))$: promise of evidence of origin of M.
- $EOO = EOO', n_A$: evidence of origin of M.
- $EOR' = sS_B(TTP, A, T, H(M, n_A), H(n_A), H(n_B))$: promise of evidence of receipt of M.
- $EOR = EOR', n_B$: evidence of receipt of M.
- $EOD = sS_{TTP}(A, B, T, M, n_A)$: evidence of delivery of M.

The protocol is as follows.

$1.\ A \rightarrow B:\ \ TTP, B, T, H(M, n_A), H(n_A), EOO'$
$2.\ B \rightarrow A:\ \ H(n_B), EOR'$
$3.\ A \rightarrow B:\ \ M, n_A$
$4.\ B \rightarrow A:\ \ n_B$

If A does not receive n_B from B after sending M and n_A, it will initiate the following error-recovery phase before the active-time limit T:

	$3'.\ A \rightarrow TTP:$	$TTP, B, T, H(M, n_A), H(n_A), EOO', H(n_B), EOR', M, n_A$
	$4'.\ TTP \rightarrow B:$	M, n_A
IF	$5'.\ B \rightarrow TTP:$	n_B
THEN	$6'.\ TTP \rightarrow A:$	n_B
ELSE	$7'.\ TTP \rightarrow A:$	EOD

This protocol needs fewer number of signature operations and thus is efficient when two parties play fair. However, at the recovery phase, A needs to send the whole message to the *TTP* for replay, and the *TTP* needs to check the validity of evidence from both A and B before replay. This is not as efficient as our variant protocol presented in last section concerning the communications overhead and *TTP*'s involvement. In our variant protocol, A only needs to submit the message key to the *TTP*, and the *TTP* only needs to confirm A's submission.

At the recovery phase of the above protocol, if B does not acknowledge *TTP*'s message $4'$, the *TTP* will issue an affidavit *EOD* certifying that the message has been sent to B within the specific time. However, *EOD* cannot be used to prove that B has received the message (within the specific time) unless there exists a strong assumption that the connection between the *TTP* and B is *completely* reliable.

6. Formal Analysis of Non-repudiation

The main characteristics of non-repudiation protocols is to establish the accountability of the participants for their actions. Formal analysis of this type of protocols can be conducted within the general framework of BAN-like logics such as the SVO logic [14], which relies mainly on its existing axioms referring to digital signatures. Kailar proposed a specific framework for reasoning about accountability [11]. However, some specific issues may be encountered in the process of analysis, e.g. the role of the trusted third party and labels in our fair non-repudiation protocol [15] and its variant proposed in this paper. We have made some efforts towards these problems in [17]. It is still an interesting topic open for future research.

7. Conclusion

Fairness can be achieved by the simultaneous exchange of messages and/or non-repudiation evidence, or by the active involvement of trusted third parties. The first approach is inefficient for actual implementation and gives fairness only under the assumption of equal computational complexity. In the second approach, reducing the active involvement of trusted third parties can improve the efficiency of a protocol. The protocol proposed in [15] is suitable for situations where two parties want to involve a trusted third party in every protocol run. The variant protocol presented in Section 4.1 is suitable for environments where the two parties normally will resolve any communications problems between themselves and rely on a trusted third party only as a last recourse.

Acknowledgements

We thank Kwok-Yan Lam for helpful discussions and the anonymous referees for suggestions for improvement on the draft of this paper. The first author was funded by the British Government and the K C Wong Education Foundation under an ORS Award and a K C Wong Scholarship when most of his research towards this paper was conducted in the Department of Computer Science, Royal Holloway, University of London.

References

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

[2] A. Bahreman and J. D. Tygar. Certified electronic mail. In *Proceedings of the Internet Society Symposium on Network*

and Distributed System Security, pages 3–19, San Diego, California, February 1994.

[3] 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.

[4] T. Coffey and P. Saidha. Non-repudiation with mandatory proof of receipt. *Computer Communication Review*, 26(1):6–17, January 1996.

[5] B. Cox, J. D. Tygar and M. Sirbu. NetBill security and transaction protocol. In *Proceedings of the First USENIX Workshop on Electronic Commerce*, July 1995.

[6] I. B. Damgård. Practical and provably secure release of a secret and exchange of signatures. In *Lecture Notes in Computer Science 765, Advances in Cryptology: Proceedings of Eurocrypt'93*, pages 200–217, Lofthus, Norway, May 1993.

[7] 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.

[8] ISO/IEC DIS 13888-1. Information technology - Security techniques - Non-repudiation - Part 1: General model. ISO/IEC JTC1/SC27 N1503, November 1996.

[9] ISO/IEC 5th CD 13888-2. Information technology - Security techniques - Non-repudiation - Part 2: Using symmetric techniques. ISO/IEC JTC1/SC27 N1505, November 1996.

[10] ISO/IEC DIS 13888-3. Information technology - Security techniques - Non-repudiation - Part 3: Using asymmetric techniques. ISO/IEC JTC1/SC27 N1507, November 1996.

[11] R. Kailar. Accountability in Electronic Commerce Protocols. *IEEE Transactions on Software Engineering*, 22(5):313–328, May 1996.

[12] T. Okamoto and K. Ohta. How to simultaneously exchange secrets by general assumptions. In *Proceedings of 2nd ACM Conference on Computer and Communications Security*, pages 184–192, Fairfax, Virginia, November 1994.

[13] J. B. Postel and J. K. Reynolds. File transfer protocol. RFC 959, October 1985.

[14] P. Syverson and P. C. van Oorschot. On unifying some cryptographic protocol logics. In *Proceedings of 1994 IEEE Symposium on Research in Security and Privacy*, pages 14–28, Oakland, California, May 1994.

[15] J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *Proceedings of 1996 IEEE Symposium on Security and Privacy*, pages 55–61, Oakland, California, May 1996.

[16] J. Zhou and D. Gollmann. Observations on non-repudiation. In *Lecture Notes in Computer Science 1163, Advances in Cryptology: Proceedings of Asiacrypt'96*, pages 133–144, Kyongju, Korea, November 1996.

[17] J. Zhou. Non-repudiation. PhD Thesis, University of London, December 1996.

[18] J. Zhou and D. Gollmann. Evidence and non-repudiation. *Journal of Network and Computer Applications*, London: Academic Press, 1997. (to appear)