Evidence and Non-repudiation

Jianying Zhou
Department of Information Systems and 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

The ultimate purpose of a non-repudiation service is to resolve disputes about the occurrence or non-occurrence of a claimed event or action. Dispute resolution relies on the evidence held by the participants. This paper discusses types of non-repudiation evidence, elements of non-repudiation evidence and validity of non-repudiation evidence. We also investigate and compare a number of protocols aiming at fair exchange of non-repudiation evidence.

Keywords: evidence, non-repudiation, communications security

1 Survey of Non-repudiation

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. ISO 7498-2 [10] defines five main categories of security services:

- *Authentication* – provides protection against masquerade.
- *Access Control* – provides protection against unauthorised access to resources.
- *Confidentiality* – provides protection against unauthorised disclosure of information.
- *Data Integrity* – provides protection against unauthorised creation, alteration, or destruction of data.

- *Non-repudiation* – provides protection against false denial of having been involved in a communication.

In this paper, we will survey the current state of the art in the design of non-repudiation protocols and examine particularly the fair exchange of evidence. Non-repudiation is related to authentication but has stronger proof requirements. The major difference is that authentication only needs to convince the other party involved in a communication of the validity of an event while non-repudiation should prove to a third party the truth of the event. Its primary purpose is to protect communications users against threats from other legitimate users, rather than from unknown attackers [29].

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. There are two international standards dealing with non-repudiation: ISO/IEC 10181-4 [13] and ISO/IEC 13888 [14, 15, 16]. ISO/IEC 10181-4 refines and extends the concept of non-repudiation services as described in ISO 7498-2 and provides a framework for the development and provision of these services. ISO/IEC 13888 is composed of three parts. They provide a general non-repudiation model [14] and a set of non-repudiation mechanisms based on symmetric and asymmetric cryptographic techniques [15, 16].

Today, non-repudiation services are fielded mainly in message handling systems and in electronic commerce. The two message handling systems in widespread use are the Internet email system, specified in various RFCs, e.g. [3, 7, 19], and X.400, specified in the CCITT X.400 series recommendations [4]. Three parties are involved in a message handling system: the originator, the recipient(s), and the delivery agent(s). Delivery agents are employed in X.400's store-and-forward model to deliver messages from the originator to the recipient. Three security elements related to non-repudiation are defined in X.400 [4]:

- *Non-repudiation of Origin* (*NRO*) provides the recipient(s) of a message with proof of origin of the message which will protect against any attempt by the originator to falsely deny sending the message. The provider of this service is the originator.
- *Non-repudiation of Delivery* provides the originator of a message with proof of delivery of the message which will protect against any attempt by the recipient(s) to falsely deny receiving the message. The provider of this service is the recipient(s).
- *Non-repudiation of Submission* (*NRS*) provides the originator of a message with proof of submission of the message which will protect against any attempt by the delivery agent to falsely deny that the message was submitted for delivery to the originally specified recipient(s). The provider of this service is the delivery agent.

The above definition for *Non-repudiation of Delivery* is confusing and would better be replaced by *Non-repudiation of Receipt* as the evidence is generated by the recipient. Non-repudiation of delivery could refer to a service that the delivery agent provides evidence of delivery. Thus, for our purposes we replace the above definition of non-repudiation of delivery by the following two elements:

- *Non-repudiation of Delivery* (*NRD*) provides the originator of a message with proof that the message has been delivered to the originally specified recipient(s). The provider of this service is the delivery agent.

- *Non-repudiation of Receipt* (*NRR*) provides the originator of a message with proof of receipt of the message which will protect against any attempt by the recipient(s) to falsely deny receiving the message. The provider of this service is the recipient(s).

As pointed out in [27], due to the structure of the X.400 delivery service, after a message has been delivered, it is up to the recipient to acknowledge the receipt, and the recipient may issue such an acknowledgement only selectively. This *selective receipt* problem is undesirable for originators requiring guarantees about the delivery result under all circumstances. This problem also exists in ISO/IEC 13888 drafts [15, 16] and has been addressed in the fair non-repudiation protocols proposed in [26, 28].

The central issue in electronic commerce is how to accomplish payment and goods delivery efficiently, reliably and securely, and provide appropriate evidence on payment and goods delivery to settle possible disputes. A significant number of electronic commerce systems have been or are being developed, such as iKP [2], NetBill [6], SET [22]. Most of them only concern themselves with non-repudiation of origin. The NetBill system ensures fair exchange by linking payment and goods delivery into a single operation called *atomic certified delivery*: the consumer is guaranteed of the delivery of goods before payment is processed, and the merchant is guaranteed that the consumer cannot access the goods until payment has been received. We give a further discussion on the NetBill transaction protocol in Section 5.5.

In this paper, we employ the following notation to represent messages and protocols.

- X, Y: concatenation of two messages X and Y.
- $H(X)$: a one-way hash function of message X.
- $gK(X)$: an integrity mechanism for message X with key K, e.g. a Message Authentication Code (MAC) as defined in ISO/IEC 9797 [12], or a keyed one-way hash function $H(K, X)$ [24].
- $eK(X)$: 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.
- $SENV_K(X)$: a secure envelope which is generated by using symmetric cryptographic techniques and allows the holder of secret key K to authenticate the integrity and origin of message X.
- $A \rightarrow B : X$: principal A sends message X to principal B.
- $A \leftrightarrow B : X$: principal A fetches message X from principal B using *"ftp get"* operation [20].

2 Types of Evidence

Evidence is information that either by itself or when used in conjunction with other information is used to establish proof about an event or action [14]. Non-repudiation evidence should satisfy the following requirements.

- The *origin* of the evidence is verifiable by a third party;
- The *integrity* of the evidence is verifiable by a third party;
- The *validity* of the evidence is undeniable.

Non-repudiation evidence can be represented by the following two types of security mechanisms:

- *Secure Envelopes* generated by trusted third parties using symmetric cryptographic techniques;
- *Digital Signatures* generated by any party using asymmetric cryptographic techniques.

A secure envelope $SENV_K(y)$ could be formed in the following two ways:

1. using symmetric encipherment: $SENV_K(y) = y, eK(H(y))$;
2. using an integrity mechanism: $SENV_K(y) = y, gK(y)$.

A secure envelope provides protection of the origin and integrity of a message when the secret key K is shared between two parties. There is nothing, however, to prevent either party from changing the message 'protected' by the secure envelope or denying its origin and content. For a secure envelope to become irrefutable evidence, it must be generated by a trusted third party using a secret key known only to this trusted third party. Other parties are unable to verify such evidence directly, but are assured of its validity through the mediation of the trusted third party.

A digital signature is a function which, when applied to a message, produces a result which enables the recipient to verify the origin and integrity of the message. Moreover, it has the property that only the originator of the message can produce a valid signature. There are two basic types of digital signatures:

1. digital signature 'with message recovery', where the verification process reveals the message together with its specific redundancy (e.g. ISO/IEC 9796 [11] based on a generalised version of RSA [21]); and
2. digital signature 'with appendix', where the verification process requires the message as part of the input (e.g. US Digital Signature Standard [18] based on the ElGamal signature algorithm [9]).

Digital signatures can be generated by any party as non-repudiation evidence if the verification key has been certified by a trusted third party. Any party can verify such evidence if it has access to the verification keys and can check their validity.

3 Elements of Evidence

Non-repudiation evidence provides information about the occurrence of an event or action, the time of occurrence, and the parties involved in the event or taking the action. Non-repudiation evidence relating to the transfer of a message may include the following elements, some of which are optional and depend on the given application:

- the type of non-repudiation service being provided,
- the identifier of the originator,
- the identifier of the recipient,
- the identifier of the evidence generator when different from the originator and the recipient,
- the identifiers of other (trusted) third parties involved,
- the message to be transferred,
- a trusted time stamp identifying when the action regarding the message transfer took place,
- a trusted time stamp identifying when the evidence was generated,
- expiry date of this evidence.

Two trusted time stamps are special elements in non-repudiation evidence, a time stamp identifying when the action regarding the message transfer took place and a time stamp identifying when the relevant evidence was generated. We will discuss this issue in the next section.

4 Validity of Evidence

A non-repudiation service will fail if bogus evidence is accepted. The validity of non-repudiation evidence relies crucially on the security of private keys and secret keys used for generating evidence. It is possible that such a key becomes compromised and needs to be revoked. Furthermore, in practice keys used for generating and verifying evidence have a limited validity period. Thus, we have to consider two possible cases when a key for evidence generation has been revoked or has expired:

- evidence was generated when the key was still valid; and
- evidence was generated after the key was revoked or had expired.

Obviously, evidence users have to be able to distinguish between these two cases. Trusted time stamps identifying when the evidence was generated and when the key for evidence generation was revoked, and the notarised expiry date of the key for evidence generation can be used to resolve a possible dispute over the validity of the evidence. Evidence generated with a key prior to its revocation and expiration is regarded as valid. Of course, this is based on the assumption that a time stamping authority can provide secure and trusted services.

Evidence can be time-stamped by sending the evidence to a time stamping authority which will append a time value to the evidence and then digitally sign the result. This will be regarded as the legal time of evidence generation, although there will be a short delay between the real time of evidence generation and the time at which the time stamp is applied.

If long term non-repudiation is to be achieved, it may be necessary during dispute resolution to make use of an expired verification key to check the evidence which was generated before the corresponding generation key had expired. Secure archives may also be needed for users expecting evidence to last indefinitely, which could be a service administered by a trusted third party. However, users who do not want to pay for such a service would like to maintain long term non-repudiation only within a reasonable period and do not expect to store evidence forever. Therefore, one could impose a time limit on the validity of evidence, which is defined in the non-repudiation policy and accepted by evidence users. No judgement will be made if disputes arise after evidence has expired.

There is a problem in the definition of time stamps in the non-repudiation evidence in the current draft of ISO/IEC 13888-3 [16], where the time stamps are added into the non-repudiation evidence by the originator and the recipient rather than appended to the evidence by a trusted time stamping authority with its signature. Such time stamps are not suitable for non-repudiation purposes. For example, a simplified form of evidence of origin (EOO) in [16] is

$$EOO = A, B, T_g, T_1, M, sS_A(A, B, T_g, T_1, M),$$

where A and B are the identifiers of the originator and the recipient respectively, T_g is the date and time EOO was generated, T_1 is the date and time the message M was sent. What can these time stamps achieve?

The time stamp T_1 in EOO may provide the recipient with information about the timeliness of message transfer. It could also be used as a label to identify the message transfer. However, as A can fake its own time stamp, B may not believe that the message was sent at that time. For example, A may add a time stamp T_1 into EOO but send the message at time T_1'. As pointed out in [28], this problem cannot be solved even if T_1 is generated by a trusted time-stamping authority TS as follows:

$$\begin{aligned} &1.\ A \rightarrow TS: && EOO \\ &2.\ TS \rightarrow A: && sS_{TS}(EOO, T_1) \end{aligned}$$

This can only prove that the message was sent *after* T_1 and the actual time of sending the message is still under A's control. Therefore, T_1 does not constitute suitable non-repudiation evidence in EOO. Of course, this is up to the recipient B. If B has received EOO and regards T_1 as being acceptable, A cannot repudiate sending M at T_1.

The validity check of evidence EOO relies on the time stamp T_g. However, if the signature key S_A becomes compromised (accidentally or deliberately) after T_g, there is no way to prove whether EOO is valid as anybody with the compromised S_A can forge EOO by including a proper time stamp in the EOO to make it valid. Therefore, T_g should be generated by a trusted time-stamping authority TS. We can define the time-stamped EOO as follows:

$$\begin{aligned} EOO &= A, B, M, sS_A(A, B, M) \\ EOO_t &= EOO, T_g, sS_{TS}(EOO, T_g) \end{aligned}$$

EOO_t remains valid even if S_A becomes compromised after T_g. Of course, this requires that S_{TS} is not compromised. For simplicity, we omit the time stamp of evidence generation in the following discussion of non-repudiation protocols.

5 Fair Non-repudiation Protocols

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 [26]:

- 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 [26].

5.1 Low-weight Notary

A fair non-repudiation protocol is proposed in [26], 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 has the following steps:

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

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 [1]. If the time of submission or delivery of a message is relevant to a dispute, a delivery authority has to provide the appropriate evidence [28].

5.2 Non-repudiation with Mandatory NRR

A non-repudiation protocol with mandatory *NRR* was proposed in [5], where the originator A wants to send a message M to the recipient B and obtain a proof of receipt. All communications between A and B take place through a trusted third party *TTP*. The following notation is used to describe the protocol.

- *Req*: request for a non-repudiable data transfer.
- n_A and n_B: nonces chosen by TTP.
- M: message to be sent from A to B.
- $EOO = sS_A(A, B, M)$: evidence of origin of M.
- $EOR = sS_B(A, B, H(EOO))$: evidence of receiving a message satisfying $H(EOO)$.

A simplified version of the protocol is as follows.

$$\begin{array}{ll}
1.\ A \rightarrow TTP: & Req \\
2.\ TTP \rightarrow A: & eV_A(n_A) \\
3.\ A \rightarrow TTP: & eV_{TTP}(n_A, A, B, M, EOO) \\
4.\ TTP \rightarrow B: & eV_B(n_B, A, B, H(EOO)) \\
5.\ B \rightarrow TTP: & eV_{TTP}(n_B, A, B, EOR) \\
6.\ TTP \rightarrow B: & eV_B(A, B, M, EOO) \\
7.\ TTP \rightarrow A: & eV_A(A, B, EOR)
\end{array}$$

This protocol uses nonces to prevent replay which is totally unnecessary in a non-repudiation service. Obviously, *EOO* and *EOR* are easy to duplicate. The number of copies of *EOO* and *EOR* does not imply the times of M has been sent or received. If a message needs to be transferred repeatedly, this should be reflected in the non-repudiation evidence, e.g. by adding different time stamps or transaction IDs into *EOO* and *EOR*.

The protocol uses public-key cryptography to encrypt all messages. This is definitely inefficient because of high computational overhead. Moreover, most of these encryptions are unnecessary for non-repudiation services.

One of the rules for dispute resolution applied in the protocol is that "If *TTP* holds *EOR*, then the data exchange is deemed to have taken place." This is not always true if the communication channels are unreliable. An unfair situation could occur if either A did not receive message 7 or B did not receive messages 6.

[1] The originator A cannot submit the message key to the *TTP* before the recipient B acknowledges A's commitment. Otherwise, B can falsely deny receiving the message without acknowledgement.

5.3 Protocol B-CEM

A certified electronic mail protocol B-CEM was proposed in [1], where the sender A wants to send a mail item M to the recipient B and get a receipt from B. An in-line trusted third party called an electronic postman P is involved. The following notation is used to describe the protocol.

- M: mail item to be sent from A to B.
- $EOO = sS_A(P, B, M)$: evidence of origin of M.
- $EOS = sS_P(A, B, EOO)$: evidence of submission of M.
- K: message key defined by P.
- $C = eK(M, EOO)$: encrypted mail item.
- $EOR = sS_B(P, A, C)$: evidence of receipt of an encrypted mail item C.
- $EOD = sS_P(A, B, K, EOR)$: evidence of delivery of M.

Protocol B-CEM is as follows.

$$\begin{array}{ll}
1.\ A \rightarrow P: & P, B, M, EOO \\
2.\ P \rightarrow A: & A, B, EOS \\
3.\ P \rightarrow B: & A, B, C, sS_P(A, B, C) \\
4.\ B \rightarrow P: & P, A, EOR \\
5.\ P \rightarrow B: & A, B, K, sS_P(A, B, K) \\
6.\ P \rightarrow A: & A, B, K, EOR, EOD
\end{array}$$

The protocol claims that it is the responsibility of each message sender to make sure that the message is delivered. Such assurance can only be achieved by assuming that the communication channel is *completely* reliable, or asking for an acknowledgement from the receiver. Obviously, the assumption is very strong and frequently unwarranted. The requirement of acknowledgement is also unguaranteed, even if the message is sent repeatedly, if the receiver does not play fair and ignores the message. Therefore, it is hard for P to make sure that B and A have received messages 5 and 6 respectively at the end of the above protocol.

In Protocol B-CEM, P acts as a delivery authority, which needs more active involvement than the TTP, which acts as a 'low weight notary' in Section 5.1. Furthermore, the process of delivery using encryption in Protocol B-CEM can be simplified [28].

5.4 Protocol CMP1

A certified electronic mail protocol CMP1 was proposed in [8]. Three parties are involved in the protocol: the mail sender A, the mail recipient B, and the trusted electronic postman P. The following notation is used to describe the protocol.

- M: mail item to be sent from A to B.
- K: session key defined by A.
- $EOO = sS_A(A, B, P, M)$: evidence of origin of M.

- $EOR = sS_B(A, B, P, H(M))$: evidence of receiving a message satisfying $H(M)$.
- $EOO\_P = sS_P(EOO)$: EOO confirmed by P.
- $EOD = sS_P(B, M, EOR)$: evidence of delivery of M.

Protocol CMP1 is as follows.

$$\begin{array}{ll} 1.\ A \to B: & A, B, P, H(M), eV_P(K), eK(M, EOO) \\ 2.\ B \to P: & A, B, P, H(M), eV_P(K), eK(M, EOO), EOR \\ 3.\ P \to B: & A, B, P, M, EOO\_P \\ 4.\ P \to A: & A, B, P, EOR, EOD \end{array}$$

Three approaches are suggested for the implementation of Steps 3 and 4 in Protocol CMP1.

(1) P 'pushes' messages 3 and 4 to B and A respectively. More specifically, P starts up a timer, sends out the message and waits for an acknowledgement ACK from the receiver. If the ACK is not received when the timer expires, P retransmits the message. Such process continues until the message is acknowledged by the receiver. This approach ensures reliable delivery over unreliable networks, but it suffers from the *selective receipt* problem. After B receives message 3 and reads the mail content M, it may refuse to send its ACK to P. Without getting the ACK, P is not sure whether B has received message 3.

(2) P is 'pulled' to send messages 3 and 4 to B and A respectively. It is B's and A's responsibility to make repeated requests to P until they get messages 3 and 4 respectively.

(3) P maintains a public bulletin board which is accessible (read only) to everyone. P sends messages 3 and 4 to B and A only once, then publishes the messages in the bulletin board. It is similar to the idea of public access used by the protocol in Section 5.1.

We make the following comments on Protocol CMP1.

- It would be more efficient if P sends K rather than M back to B as the size of K is usually smaller than the size of M. Moreover, EOO_P can be omitted as EOO itself serves as evidence of origin and B can obtain it by deciphering $eK(M, EOO)$. (Of course, P should first check the validity of non-repudiation evidence from A and B before sending messages 3 and 4.) Therefore, Step 3 can be simplified as

$$3.\ P \to B: \quad A, B, P, H(M), K$$

- Using $A, B, P, H(M)$ as a mail identifier is dangerous. $H(M)$ may leak information about M when M belongs to a small message space, which may give B a chance to decide whether to continue with a protocol run at Step 2.
- It is claimed in [8] that the protocol reaches the lower bounds on the number of messages and rounds. But it is not efficient, especially when the mail item is very large, as the whole message is transmitted three times in one protocol run (at least twice even if Step 3 is optimized).

- P acts as an in-line delivery authority. As pointed out in [28], we usually need such involvement of a trusted third party in the case where evidence of submission and delivery of a message (with appropriate time information) is required for dispute resolution. However, this protocol cannot provide such evidence. As the message is first sent to B and then forwarded to P for decryption, B can delay or omit to forward the message to P so that A will not get the desired evidence from P.

- There is a heavy workload on P, which can easily cause P to be a bottleneck. P needs to decrypt the mail item and check the validity of non-repudiation evidence from A and B before delivery. P also needs to maintain a very large public bulletin board to store the entire mail items (which can be avoided if Step 3 is optimized).

5.5 Certified Delivery

Fair non-repudiation is required in electronic commerce concerning payment and goods delivery. Cox et al. proposed an atomic certified delivery method in the NetBill system so that a customer pays if and only if he receives his electronic goods intact [6]. In the NetBill system, a customer A wishes to buy electronic goods from a merchant B. The NetBill server N maintains accounts for both customers and merchants, linked to conventional financial institutions. A NetBill transaction transfers electronic goods from a merchant to a customer, debits the customer's NetBill account and credits the merchant's account for the value of the goods. The following notation is used to describe the protocol.

- *TID*: transaction ID which is used by customer A and merchant B to maintain context between them. *TID* is not globally unique.

- *PRD*: product request data used by A to specify the goods.

- *PID*: product ID provided by B.

- *Price*: price of the requested goods.

- M: electronic goods to be delivered from B to A.

- K: message key defined by B.

- $C = eK(M)$: encrypted electronic goods.

- *EPOID*: electronic payment order ID. *EPOID* is a globally unique identifier which is used in the NetBill server's database to uniquely identify a transaction.

- $EPO = sS_A(PID, Price, B, H(C), H(PRD), EPOID)$: electronic payment order.

- $Receipt = sS_N(PID, Price, A, B, K, EPOID)$: receipt of the transaction.

A simplified version of the protocol is described as follows, which includes three phases: price negotiation (Steps 1 and 2), goods delivery (Steps 3 and 4), and payment (Steps 5 to 8). It is assumed that A, B, and N are each equipped with their own private signature key and the relevant public verification keys. It is also assumed that each party holds appropriate secret keys for secure communication with other parties.

1. $A \rightarrow B:$ $eK_{AB}(PRD, TID)$
2. $B \rightarrow A:$ $eK_{AB}(PID, Price, TID)$
3. $A \rightarrow B:$ $eK_{AB}(TID)$
4. $B \rightarrow A:$ $C, eK_{AB}(H(C), EPOID)$
5. $A \rightarrow B:$ $eK_{AB}(EPOID, EPO)$
6. $B \rightarrow N:$ $eK_{BN}(PID, Price, H(C), H(PRD), EPOID, EPO, K, sS_B(EPO, K))$
7. $N \rightarrow B:$ $eK_{BN}(EPOID, Receipt)$
8. $B \rightarrow A:$ $eK_{AB}(K, EPOID, Receipt)$

This certified delivery protocol and the fair non-repudiation protocol of Section 5.1 share similar ideas. The electronic goods M is split into two parts: the encrypted goods C and the key K. After C is delivered at Step 4, the customer A submits payment to the merchant B in the form of a signed electronic payment order *EPO* at Step 5. A can abort the protocol before Step 5 without any disputes. The submission of *EPO* marks the "point of no return" for A. After receiving *EPO* from A, B endorses it, adding the key K, and forwards the endorsed *EPO* to the NetBill server N at Step 6. B can abort the protocol before Step 6 without any disputes. The submission of the endorsed *EPO* marks the "point of no return" for B.

Upon receipt of the signed and endorsed *EPO*, N makes a decision about the transaction which is based on verification of the signatures, the uniqueness of *EPOID*, the customer's account balance etc., and returns the receipt to B, which in turn forwards it to A. In the event of communications failure after Step 5, the customer or the merchant may be unware of the transaction status. The system supports a number of status queries:

- The merchant requests the transaction status from the NetBill server:

$$\begin{array}{ll} Q1.\ B \rightarrow N: & eK_{BN}(EPOID) \\ Q2.\ N \rightarrow B: & eK_{BN}(EPOID, Receipt) \end{array}$$

- The customer requests the transaction status from the merchant:

$$\begin{array}{ll} Q1.\ A \rightarrow B: & eK_{AB}(EPOID) \\ Q2.\ B \rightarrow A: & eK_{AB}(K, EPOID, Receipt) \end{array}$$

- The customer requests the transaction status from the NetBill server:

$$\begin{array}{ll} Q1.\ A \rightarrow N: & eK_{AN}(EPOID) \\ Q2.\ N \rightarrow A: & eK_{AN}(K, EPOID, Receipt) \end{array}$$

This implies that the merchant may also need to store all of its transaction receipts signed by the NetBill server. Otherwise, the merchant has to first request the receipt from the NetBill server if the customer requests the status from the merchant. Moreover, the customer has no evidence that the merchant has sent the encrypted goods C to him. If there is a dispute over the content of the electronic goods, further evidence like the endorsed *EPO* is required from N besides the *Receipt*. To simplify the process of arbitration, it is better to include $H(C)$ in the *Receipt*.

6 Implementation Issues

To provide a real non-repudiation service in a given application, a number of implementation issues need to be resolved. We will use the protocol described in Section 5.1 to explore those issues. The following discussions will be guided by our aim to involve the trusted third party as little as possible in a protocol run.

6.1 Timeout and State Information

The sender A has to define a timeout period indicating how long it is prepared to wait for a response from the receiver B before abandoning the protocol run. A response arriving after the timeout period will be ignored. The sender has to maintain state information about all active protocol runs. The length of the timeout period and the number of protocol runs the sender can execute concurrently will depend on the application.

The receiver B has to keep state information about a protocol run from the time it acknowledges the sender's commitment until it has retrieved the corresponding key from the TTP. For practical purposes, B has to be able to abort incomplete protocol runs. This problem can be handled in several ways. Here, we present a simple and flexible solution where all time stamps refer to the TTP's clock.

1. The sender A includes a deadline T in its evidence EOO. The confirmed message key is expected to become publicly available before this deadline.

2. If B agrees with the deadline T, B can acknowledge A's commitment and include T in its reply EOR. If B inserts a deadline T' different from T, A has to abort the protocol run.

3. When submitting the message key to the TTP, the sender includes the deadline T in sub_K. If A's submission arrives after time T, the TTP will not confirm A's submission and the key will never appear in the public directory. Otherwise, T will be included in con_K.

4. When adjudicating a dispute, the arbitrator refutes any evidence where the deadlines in con_K and EOO or EOR differ. Hence, if A submits the message key with a deadline T' different from T, it cannot win a dispute[2].

5. When B attempts to retrieve the confirmed key for a label L, it will either receive con_K or a signed and time-stamped message from the TTP indicating that no key for label L exists. Once B receives such a message with a time stamp later than T, it can abandon the protocol run safely.

A policy decision affecting the trusted third party has to regulate how long confirmed message keys have to be kept in the public directory. The trusted third party does not have to maintain state information about a protocol run other than the confirmed message keys. Moreover, labels can be defined in such a way that the trusted third party's check on message keys submitted by the originator is simplified [26].

[2] Here we can treat (L, T) as a unique label. If (L, T) in EOO, EOR and con_K do not match, a claim will not succeed.

6.2 Length and Number of Messages

A successful run of a non-repudiation protocol will transmit data M from sender to receiver and generate non-repudiation evidence relating to M. In our protocol, only the first message is of length commensurate to M. All other messages, and in particular the messages sent to the trusted third party are of fixed length. It is sometimes possible to reduce the number of messages at the expense of increasing the length of the messages exchanged. In a practical situation, one has to consider carefully whether such a reduction in messages actually results in any real savings.

6.3 The Choice of Cryptographic Algorithms

As far as fair non-repudiation is concerned, the strength of the algorithm used by the sender to construct the initial commitment is linked to the sender's timeout period. With a short timeout period, we can emphasize speed when selecting the cryptographic algorithm. Of course, there may be issues specific to the application advising the sender to give longer term protection to the message the commitment refers to.

All three parties involved in a protocol run have to generate and to verify digital signatures. Hence, a signature scheme favouring the verifier (or the signer) will not have too much impact on the computational effort required from each party. The strength of the signature scheme is related to the intended lifetime of the signature keys. For dispute resolution, we also need procedures for dealing with the revocation of signature keys, stating under which circumstances evidence generated by a revoked key is still deemed to be valid.

6.4 Reliable and Secure TTP

Trusted third parties play important roles in non-repudiation services and their reliability and security is a practical problem which needs to be addressed. For various reasons, a trusted third party may fail or make mistakes. In our fair non-repudiation protocol, if the TTP crashed and lost the key con_K at a time when one entity had collected con_K while another entity had not, fairness will be disrupted. To increase its reliability, the TTP should be able to recover from network and system failures, and maintain a consistent directory accessible (read only) to the public. This requires proper backup and recovery mechanisms [25].

Non-repudiation services rely on trustworthy functions of the trusted third parties involved. Any dishonest actions of trusted third parties will affect non-repudiation services. For example, if a certification authority issues false verification key certificates, evidence users cannot check the validity of non-repudiation evidence. In the real world, an individual trusted third party may not be completely and permanently trustworthy. To enhance trust in non-repudiation services, one possible solution is to distribute trust among a group of trusted third parties using threshold schemes [17, 23], so that a minority of malicious and colluding servers cannot compromise security or disrupt services. However, this solution will increase the cost and downgrade the efficiency of non-repudiation services. A practical solution is to put a trusted third party under supervision.

7 Conclusion

Non-repudiation services generate evidence about the occurrence of a particular event or action which can be employed by an arbitrator to later resolve potential disputes over these events. Many applications will require time stamps to be included in non-repudiation evidence. Time stamps serve two purposes: they indicate the validity of evidence when verification keys can be revoked or expire, and they can indicate the time a particular event took place. In both cases, the time stamps have to be guaranteed by a trusted third party. Fairness may be another desirable property of a non-repudiation protocol. We have investigated a number of current proposals aiming at fair exchange of non-repudiation evidence which differ in

- the degree to which a trusted third party has to be involved in a protocol run, and
- the assumptions about the nature of the items exchanged in the service non-repudiation refers to.

We have warned against a too liberal, and unnecessary, use of cryptographic algorithms and against too stringent assumptions on the availability of communications links. Finally, we point out that the standardisation of non-repudiation mechanisms within ISO/IEC 13888 has still some way to go before all problems are resolved satisfactorily.

Acknowledgements

We thank the anonymous referees for their suggestion to include implementation issues in 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] 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.

[2] M. Bellare, J. A. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik and M. Waidner. *iKP - A family of secure electronic payment protocols.* Proceedings of the First USENIX Workshop on Electronic Commerce, pages 89–106, July 1995.

[3] N. Borenstein and N. Freed. *Multipurpose Internet mail extensions (MIME).* RFC 1521, September 1993.

[4] CCITT. *Recommendation X.400: Message handling system and service overview.* November 1988.

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

[6] 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.

[7] D. Crocker. *Standard for the format of ARPA Internet text messages.* RFC 822, August 1982.

[8] 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.

[9] T. ElGamal. *A public-key cryptosystem and a signature scheme based on discrete logarithms.* IEEE Transactions on Information Theory, IT-31(4):469–472, July 1985.

[10] ISO 7498-2. *Information processing system - Open systems interconnection - Basic reference model - Part 2: Security architecture.* International Organization for Standardization, 1989.

[11] ISO/IEC 9796. *Information technology - Security techniques - Digital signature scheme giving message recovery.* ISO/IEC, September 1991.

[12] ISO/IEC 9797. *Information technology - Security techniques - Data integrity mechanism using a cryptographic check function employing a block cipher algorithm.* ISO/IEC, April 1994 (second edition).

[13] ISO/IEC 10181-4. *Information technology - Open systems interconnection - Security frameworks in open systems - Part 4: Non-repudiation.* ISO/IEC, 1996.

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

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

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

[17] S. C. Kothari. *Generalized linear threshold scheme.* Lecture Notes in Computer Science 196, Advances in Cryptology: Proceedings of Crypto'84, pages 231–241, Santa Barbara, California, August 1984.

[18] NIST FIPS PUB 186. *Digital signature standard.* National Institute of Standards and Technology, May 1994.

[19] J. B. Postel. *Simple mail transfer protocol.* RFC 821, August 1982.

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

[21] R. Rivest, A. Shamir and L. Adelman. *A method for obtaining digital signatures and public-key cryptosystems.* Communications of the ACM, 21(2):120–126, February 1978.

[22] SET. *Secure electronic transaction specification.*
$< http://www.mastercard.com/set/set.htm >$

[23] A. Shamir. *How to share a secret.* Communications of the ACM, 22(11):612–613, November, 1979.

[24] G. Tsudik. *Message authentication with one-way hash functions.* Computer Communication Review, 22(5):29–38, October 1992.

[25] J. Zhou. *The design and implementation of a recovery system in RDBMS.* MSc Thesis, Chinese Academy of Sciences, December 1988.

[26] 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.

[27] J. Zhou and D. Gollmann. *Certified electronic mail.* Lecture Notes in Computer Science 1146, Computer Security: Proceedings of ESORICS'96, pages 160–171, Rome, September 1996.

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

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