A Fair Non-repudiation Protocol

Jianying Zhou and Dieter Gollmann
Department of Computer Science
Royal Holloway, University of London
Egham, Surrey TW20 0EX, United Kingdom
email: {zhou, dieter}@dcs.rhbnc.ac.uk

Abstract

A fair non-repudiation protocol should not give the sender of a message an advantage over the receiver, or vice versa. We will present a fair non-repudiation protocol that requires a Trusted Third Party but attempts to minimize its involvement in the execution of the protocol. We will draw particular attention to the non-standard use of encryption in our protocol and discuss some aspects of its formal verification.

1. Introduction

Accountability is an important aspect of information security. In a distributed communication environment, we may therefore want to prevent entities from denying that they have sent or received certain messages. The goal of a non-repudiation service is to collect, maintain, make available, and validate irrefutable evidence regarding the transfer of a message from the originator to the recipient, possibly involving the service of a trusted third party called the *Delivery Agent*. We distinguish between the following non-repudiation services [8].

- *Non-repudiation of origin* (*NRO*) is intended to protect against the originator's false denial of having sent the message.
- *Non-repudiation of receipt* (*NRR*) is intended to protect against a recipient's false denial of having received the message.
- *Non-repudiation of submission* (*NRS*) is intended to protect against the originator's false denial of having submitted the message to the Delivery Agent.
- *Non-repudiation of delivery* (*NRD*) is facilitated by evidence that the message was forwarded by the Delivery Agent.

A non-repudiation service may be provided through the use of mechanisms such as signatures, encipherment, notarization, and data integrity mechanisms, with support from other security and system services. It is composed of four distinct phases: evidence generation, evidence transfer and storage, evidence verification, and dispute resolution [6]. Evidence could be created by a trusted third party using its secret key to generate secure envelopes, or by a non-repudiation initiator using its private key to generate digital signatures which are associated with their public key certificate. Depending on the context of a non-repudiation service, evidence will typically include the identities of parties involved, the content of transferred message, and the time and date. Evidence establishes the accountability of entities participating in a particular event or action. Evidence generation, transfer and verification will be specified by non-repudiation protocols. A fair non-repudiation protocol should not give the sender of a message an advantage over the receiver, or vice versa.

The paper is organized as follows. The next section introduces the basic notation. Section 3 examines the implications of unreliable communication channels and 'unfair' participants on the design of non-repudiation protocols. We propose a protocol for non-repudiation of origin and receipt in Section 4, which puts the originator and the recipient in an equal position. The protocol requires a trusted third party but does not depend on the reliability of a communication channel or on the communicating parties playing fair. In Section 5, we will analyse some aspects of this protocol, show that encryption is used in a non-standard fashion, and discuss the implications this may have on the formal proof of such a protocol.

0-8186-7417-2/96 $5.00

2. Basic Notation

In this paper, we employ the following notation to represent messages of a protocol.

- 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.
- $sK(X)$: digital signature of message X with the private key K.
- P_A, S_A: the public and private key of principal A.
- $A \rightarrow B : X$: principal A sends message X to principal B.
- $A \leftrightarrow B : X$: principal A fetches message X from principal B using a *"ftp get"* operation [10].

3. Background

The origin of a message will usually be verified by a digital signature appended by the sender. To obtain a proof 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.

In the following, we examine how these problems affect the design of non-repudiation protocols. Assume that principal A wishes to send message M to principal B and get the corresponding receipt. TTP is an on-line trusted third party. S_A, S_B and S_T are private signature keys held by A, B, and TTP respectively. Messages will include flags indicating the intended purpose of a (signed) message.

Case 1: The communication channel is completely reliable, and the communicating parties play fair. In this idealized situation, the protocol is very simple.

1. $A \rightarrow B$: $f_{NRO}, B, M, sS_A(f_{NRO}, B, M)$
2. $B \rightarrow A$: $f_{NRR}, A, sS_B(f_{NRR}, A, M)$

The signatures of the sender and recipient serve as proof of origin and proof of receipt respectively.

Case 2: The communication channel is completely reliable, but the communicating parties do not necessarily play fair. The above protocol leaves B in an advantageous situation. If A sends a message to B, B simply can refuse to send an acknowledgment. This problem could be addressed by invoking the services of a trusted third party.

1. $A \rightarrow TTP$: $f_{NRO}, TTP, B, M,$ $sS_A(f_{NRO}, TTP, B, M)$
2. $TTP \rightarrow B$: $f_{NRS}, A, B, M, sS_T(f_{NRS}, A, B, M)$
3. $TTP \rightarrow A$: $f_{NRD}, A, B, sS_T(f_{NRD}, A, B, M)$

The trusted third party is a Delivery Agent, whose signatures serve as proof of submission, rather than proof of origin, and proof of delivery, rather than proof of receipt. We will mention alternatives to the use of a trusted third party in the next section.

Case 3: The communication channel is not reliable, but the communicating parties play fair. In this case, we make the further assumption that the communication channel is not permanently broken. Therefore, each message can be sent repeatedly until an acknowledgement has been received. A possible communication protocol is as follows.

1. $A \rightarrow B$: $f_{NRO}, B, M, sS_A(f_{NRO}, B, M)$
2. $B \rightarrow A$: $f_{NRR}, A, sS_B(f_{NRR}, A, M)$
3. $A \rightarrow B$: $f_{ACK}, B, sS_A(f_{ACK}, B, M)$

We assume that B keeps repeating step 2 until it receives A's acknowledgement.

Case 4: The communication channel is not reliable, and the communicating parties do not necessarily play fair. In both previous protocols, the recipient still holds the advantage. In the protocol suggested for Case 2, B could deny having received the TTP's message while in the protocol of Case 3, B need not provide a proof of receipt. To prevent B from acknowledging only convenient messages, the protocol could start with a 'promise of exchange', which does not reveal the contents of the message, and send the message in a later step. For example, the promise could include the message encrypted under a key K that is sent later.

1. $A \rightarrow B$: $f_{POE}, B, eK(M), sS_A(f_{POE}, B, eK(M))$
2. $B \rightarrow A$: $f_{ACP}, A, sS_B(f_{ACP}, A, eK(M))$
3. $A \rightarrow B$: $f_{NRO}, B, K, sS_A(f_{NRO}, B, K)$
4. $B \rightarrow A$: $f_{NRR}, A, sS_B(f_{NRR}, A, K)$

Unfortunately, this does not solve the problem. B could still refuse to send the last message, leaving A without a proof of receipt. If we omit the last step, A will hold the advantage as it is not forced to release the key K to B. Some ISO drafts [7], which have since been superseded, did include variations of this protocol and had to declare protocol steps *mandatory* and to rely on arguments like the following *consecutiveness* property to justify why such protocols would meet their purpose.

> If B obtains NRO after a preliminary promise, then B must send NRR back to A. Otherwise, B is proved to be wrong by the adjudicator.

It is not clear how the adjudicator can establish that B did obtain NRO other than by accepting A's word that NRO was sent and assuming that the network will deliver the message. Relying on properties external to the protocol is unsatisfactory from a theoretical viewpoint and dangerous from a practical viewpoint.

4. A Fair Non-repudiation Protocol

In the initial definition of non-repudiation, we were only concerned with the evidence produced by a complete protocol run. We will now move the goal posts and put additional requirements on incomplete protocol runs.

> **Definition.** 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 at any stage of the protocol run.

The difficulty we face is the 'simultaneous' exchange of a message and its receipt between two principals who potentially do not trust each other. The originator would like to get a confirmation from the intended recipient certifying that the message has been received. On the other hand, the recipient would like to have the message when it issues the corresponding receipt. Generally speaking, there are two kinds of solutions to this problem. One uses simultaneous secret exchange [2, 3, 4, 5, 9], the other uses trusted third parties. The first solution interleaves sending the message and the receipt so that the recipient has only a negligible advantage. This approach seems to be too cumbersome 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 [1]. We will suggest a solution based on a trusted third party, and try to minimize its involvement in the protocol run.

4.1. The Protocol

The main idea of our protocol is to split the definition of the message into two parts, a commitment C and a key K. The commitment is exchanged between A and B and then the key is lodged with a trusted third party TTP. Both A and B have to retrieve the confirmed key from TTP as part of the non-repudiation evidence required in a dispute (see Figure 1). We assume that even in case of network failures, both parties will eventually be able to retrieve the key from TTP. The notation in the protocol description is as follows.

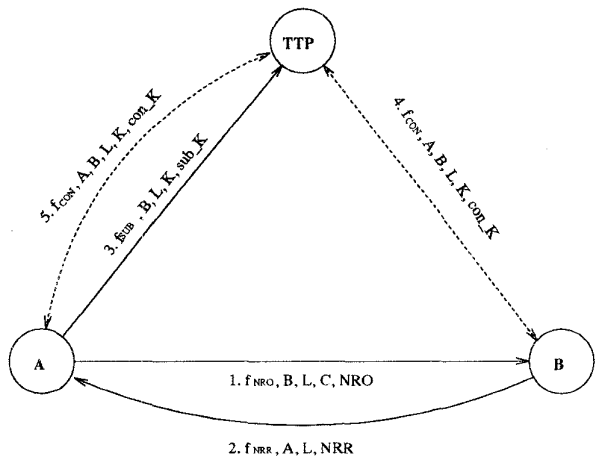

Figure 1. A Fair Non-repudiation Protocol

- A: originator of the non-repudiation exchange.
- B: recipient of the non-repudiation exchange.
- TTP: on-line trusted third party providing network services accessible to the public.
- 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.
- $NRO = sS_A(f_{NRO}, B, L, C)$: Non-repudiation of Origin for M.
- $NRR = sS_B(f_{NRR}, A, L, C)$: Non-repudiation of Receipt of M.

- $sub\_K = sS_A(f_{SUB}, B, L, K)$: proof of submission of K.
- $con\_K = sS_T(f_{CON}, A, B, L, K)$: confirmation of K issued by TTP.

We assume that A, B, and TTP are each equipped with their own private signature key and the relevant public verification keys. A unique label L links all messages of a particular protocol run together. Flags indicate the intended purpose of a (signed) message. The protocol is described as follows.

1. $A \rightarrow B$: f_{NRO}, B, L, C, NRO
2. $B \rightarrow A$: f_{NRR}, A, L, NRR
3. $A \rightarrow TTP$: f_{SUB}, B, L, K, sub_K
4. $B \leftrightarrow TTP$: $f_{CON}, A, B, L, K, con\_K$
5. $A \leftrightarrow TTP$: $f_{CON}, A, B, L, K, con\_K$

The identifiers of originator and recipient are included in the signed messages to protect them from being illegally used. Steps 4 and 5 are *"ftp get"* operations by B and A respectively. The message M is obtained by deciphering C under key K.

We examine the situation of sender and receiver after each step of the protocol.

(1) $A \rightarrow B$: If A has sent NRO and C to B, but they fail to reach B, the protocol ends without disputes.

(2) $B \rightarrow A$: After receiving NRO and C from A, B can decide whether or not to acknowledge them. If B does not send NRR to A, the protocol ends without disputes. There is no effect on A because B does not know the meaning of C without K. In order to get K and con_K from TTP to decipher C and win a possible dispute over the origin of M later, B has to send this message to A. If it fails to reach A, the protocol ends without disputes.

After receiving NRR from B, A should compare its label with NRO's label and end the protocol if they do not match. Otherwise, A may lose a possible dispute over the receipt of M later.

(3) $A \rightarrow TTP$: If A does not send sub_K and K to TTP, the protocol ends without disputes. In order to get con_K from TTP to win a possible dispute over the receipt of M later, A has to send this message to TTP. If A sends sub_K with a wrong label L, neither A nor B can win a dispute later as no one can match con_K with NRO (NRR).

B cannot obtain K, and thereby the message M, before K is lodged with TTP. We assume that B is not in a position to block sub_K permanently and thus prevent A from obtaining a proof of receipt.

After receiving sub_K and K, TTP will generate con_K and store the tuple (A, B, L, K, con_K) in a directory which is accessible (read only) to the public. The first component, identifying the key supplier, corresponds to the entity associated with the public verification key P_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 TTP as this will not generate entries for A in the directory. A new key K' for A, B, L will not be accepted by TTP.

(4) $B \leftrightarrow TTP$: B fetches con_K and K from TTP and computes $M = dK(C)$. We assume that the communication channel is eventually available. B can therefore always retrieve con_K and K. B will lose the dispute over the receipt of M even if it does not fetch K after it becomes publicly available. To win a dispute over message M, B has to present NRO and con_K, which are linked by a common label L, and M and K to the judge.

(5) $A \leftrightarrow TTP$: The order of Step 4 and Step 5 is not important. After TTP has stored con_K in its directory, A can fetch it and save it as a proof. To win a dispute over message M, A has to present NRR and con_K, which are linked by a common label L, and M and K to the judge.

In this protocol, the trusted third party does not act as a Delivery Agent but as Certification Agency for message keys. This has two advantages. The trusted third party only deals with keys, which in general will be shorter than the full messages. Secondly, the onus is now on the originator and receiver to retrieve the key, while a Delivery Agent would have to keep resending messages until the receiver acknowledges the message.

4.2. 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 NRO and con_K. If B cannot provide this evidence or one of the following checks fails, B's claim will be judged invalid.

- The judge checks that con_K is TTP's signature on (f_{CON}, A, B, L, K).
- The judge checks that NRO is A's signature on (f_{NRO}, B, L, C).
- The judge checks $M = dK(C)$.

If the first check is positive, the judge will assume that the entry in the TTP's directory was made because A had submitted the key K for use with the message with label L. Here, the judge trusts the TTP to generate valid evidence. If the second check is positive, the judge will assume that A had sent C as its commitment for the message with label L. 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 NRR and con_K. If A cannot provide this evidence or one of the following checks fails, A's claim will be judged invalid.

- The judge checks that con_K is TTP's signature on (f_{CON}, A, B, L, K).
- The judge checks that NRR is B's signature on (f_{NRR}, A, L, C).
- The judge checks $M = dK(C)$.

The first check is as above. Furthermore, the judge will assume that B is able to retrieve the key K from TTP. If the second check is positive, the judge will assume that B had received C and is committed to retrieving K from TTP. If the final check is positive, the judge will uphold A's claim.

5. Further Discussions

We will now examine some aspects of our protocol in more detail.

5.1. Commitment

In the protocol, a message M is defined by a commitment C and a key K. On its own, the commitment need not restrict the content of the message in any way. For example, if encryption and decryption are bitwise XOR, then all messages are still equally possible after both parties have committed themselves to C. Rather than referring to C as the encrypted message, we should treat C as the input to a communication channel defined by K. The cryptographic key is used only for defining this channel, not for any other purpose and in particular not for confidentiality.

Of course, for all practical purposes we will want the key to be much shorter than the message and we will indeed compute $C = eK(M)$. However, this is an aspect of implementation rather than a feature of the protocol.

5.2. Labels

Labels have to be unique to create the link between commitment and key. We require the TTP to check that the keys provided to it do not overwrite existing entries in the public directory.

If we choose labels which are independent of the messages, then the message M may not be defined until Step 3 of our protocol. (This will also depend on the decryption algorithm employed.) If the label is a function of the message, e.g. $L = H(M)$ where H is a collision-free one-way hash function, then the message is already defined in Step 1. This is especially suitable for the situation where B hopes to get A's full commitment to M before B responds to it. In this case, the judge could also employ a different verification strategy, replacing the reference to the common label in NRO (NRR) and con_K by a consistency check $L = H(dK(C))$.

However, 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, violating our fairness condition. To avoid this situation, a label $H(M, K)$ could be used instead.

5.3. Time Limit

In practice, we will not want TTP to store message keys forever. We could set a deadline T to limit the time con_K and K can be accessed by the public. The

deadline is chosen by the originator A but refers to TTP's clock. The deadline is included in NRO, NRR, sub_K, and con_K. We may include an optional time stamp T_0 in con_K to indicate when the confirmed key has actually been made available to the public. The protocol is extended as follows.

$$\begin{aligned} NRO &= sS_A(f_{NRO}, B, L, T, C) \\ NRR &= sS_B(f_{NRR}, A, L, T, C) \\ sub\_K &= sS_A(f_{SUB}, B, L, T, K) \\ con\_K &= sS_T(f_{CON}, A, B, L, T, T_0, K) \end{aligned}$$

1. $A \rightarrow B$: f_{NRO}, B, L, T, C, NRO
2. $B \rightarrow A$: f_{NRR}, A, L, NRR
3. $A \rightarrow TTP$: $f_{SUB}, B, L, T, K, sub\_K$
4. $B \leftrightarrow TTP$: $f_{CON}, A, B, L, T_0, K, con\_K$
5. $A \leftrightarrow TTP$: $f_{CON}, A, B, L, T_0, K, con\_K$

If the recipient B does not agree with the deadline T, it can end the protocol at Step 2. If sub_K and K reach TTP later than the deadline T, TTP will not store K in the public directory and the protocol ends without disputes. If there are any discrepancies in the time stamps in non-repudiation evidence, a claim will not succeed.

5.4. Formal Analysis

Non-repudiation of origin and non-repudiation of receipt can easily be expressed in the notation of the SVO logic [11] as beliefs held by a judge, J, after examining the evidence provided.

(1) Non-repudiation of origin:
$$J \textit{ believes } (A \textit{ said } M)$$

(2) Non-repudiation of receipt:
$$J \textit{ believes } (B \textit{ sees } M)$$

A formal proof of our protocol would rely mainly on existing axioms referring to digital signatures. However, there are two specific issues which also have to be resolved.

- The role of the trusted third party: the protocol assumes that TTP only puts valid entries in its directory and that these entries are available to all participants. We could either create a new set of rather protocol specific axioms or, preferably, deal with this issue as part of the idealization step.
- Message keys: these keys define communication channels and are associated with messages rather than with principals. We suggest the notation $\stackrel{K}{\rightarrow} X$ to indicate that K defines a channel for message X. Axioms that allow to derive beliefs about message keys may have to refer to labels and thus require an extension of the logic.

6. Conclusion

Fair non-repudiation protocols can be constructed in two ways, by simultaneous secret exchange, or by invoking the services of a trusted third party. We follow the second approach and suggest a protocol which has the following properties.

- The protocol does not depend on the reliability of a communication channel or on the communicating parties playing fair.
- At no point of the protocol run does either participant have an advantage.
- The involvement by the trusted third party is significantly reduced.

We have observed that this protocol uses cryptographic keys to define a communication channel and that the choice of labels influences the protocol semantics. A worthwhile formal verification of the protocol within the framework of a logic of beliefs requires general axioms for reasoning about message keys, an interesting open problem for future research.

Acknowledgements

Thanks to Chris Mitchell for his valuable comments and important suggestions for improving an earlier version of this paper [12], and for making available his collection of ISO documents. We are grateful to the anonymous referees for useful comments. The first author would also like to thank the British Government and the K C Wong Education Foundation for their support through an ORS Award and a K C Wong Scholarship.

References

[1] M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, 36(1):40–46, January 1990.

[2] E. F. Brickell, D. Chaum, I. B. Damgard, and J. van de Graaf. Gradual and verifiable release of a secret. In *Advances in Cryptology: Proceedings of Crypto'87, LNCS 293*, pages 156–166. Springer Verlag, 1988.

[3] R. Cleve. Controlled gradual disclosure schemes for random bits and their applications. In *Advances in Cryptology: Proceedings of Crypto'89, LNCS 435*, pages 573–588. Springer Verlag, 1990.

[4] I. B. Damgå rd. Practical and provably secure release of a secret and exchange of signatures. In *Advances in Cryptology: Proceedings of Eurocrypt'93, LNCS 765*, pages 200–217. Springer Verlag, 1994.

[5] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, June 1985.

[6] ISO/IEC JTC1. Information technology - Open systems interconnection - Security frameworks in open systems, Part 4: Non-repudiation. ISO/IEC DIS 10181-4, April 1995.

[7] ISO/IEC JTC1/SC27 N224. 3rd working draft on Non-repudiation, Part 3: Mechanisms using asymmetric techniques. ISO/IEC WD 13888-3, July 1993.

[8] ISO/IEC JTC1/SC27 N993. 6th working draft on Non-repudiation, Part 3: Mechanisms using asymmetric techniques. ISO/IEC WD 13888-3, April 1995.

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

[10] J. B. Postel and J. Reynolds. File transfer protocol, October 1985.

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

[12] J. Zhou. *A fair non-repudiation protocol.* Technical Report CSD-TR-95-08, Department of Computer Science, Royal Holloway, University of London, March 1995.