On the Efficient Implementation of Fair Non-repudiation

Cheng-Hwee You, Jianying Zhou[*] and Kwok-Yan Lam
School of Computing
National University of Singapore
10 Kent Ridge Crescent
Singapore 119260
email: {youch,zhoujy,lamky}@comp.nus.edu.sg

Abstract

Due to the explosive growth of electronic businesses carried on the Internet, non-repudiation services turn out to be increasingly important. Non-repudiation services protect the transacting parties against any false denial that a particular event or action has taken place, in which evidence will be generated, collected and maintained to enable the settlement of disputes. Several fair non-repudiation protocols have been proposed, which support non-repudiation of origin and non-repudiation of receipt while neither the originator nor the recipient can gain an advantage by quitting prematurely or otherwise misbehaving during a transaction. However, a critical issue on how to maintain the validity of non-repudiation evidence efficiently during and after a transaction was not considered. This paper uses the idea of evidence chaining to address such a problem.

Keywords: fair non-repudiation, validity of evidence, evidence chaining, dispute resolution

1 Introduction

Due to the explosive growth of electronic businesses on the Internet, non-repudiation services turn out to be increasingly important. Non-repudiation services protect the transacting parties against any false denial that a particular event or action has taken place, in which evidence will be generated, collected and maintained to enable the settlement of disputes [20].

Typical disputes that may arise in a simple transaction such as transferring a message M (e.g. electronic cash or electronic contracts) from Alice to Bob could be

- Alice claims that she has sent M to Bob while Bob denies having received it;
- Bob claims that he received M from Alice while Alice denies sending it.

The basic non-repudiation services that address the above disputes are

[*]Jianying Zhou's new address: Kent Ridge Digital Labs, 21 Heng Mui Keng Terrace, Singapore 119613. email: jyzhou@krdl.org.sg

- *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.

One of the critical issues on non-repudiation protocol design is the efficient approach for maintaining the validity of non-repudiation evidence during and after a transaction. Several fair non-repudiation protocols (e.g. [2, 4, 7, 19]) have been proposed, which support non-repudiation of origin and non-repudiation of receipt while neither the originator nor the recipient can gain an advantage by quitting prematurely or otherwise misbehaving during a transaction. However, the above issue has not been considered, which will affect the efficiency of non-repudiation protocols. Non-repudiation evidence is usually generated by means of digital signature [9, 14, 20]. As signature keys may be compromised and the validity of signatures may become questionable, additional security mechanisms need to be imposed on digital signatures [22].

A conventional approach for maintaining the validity of digital signatures requires that either the evidence sender or the evidence receiver interacts with an on-line trusted time-stamping authority to get each newly generated digital signature time-stamped [1, 5, 17] so that there is extra evidence to prove whether the signature was generated before the corresponding public key certificate was revoked and thus is deemed valid. However, such an approach is not cost-effective in ordinary on-line transactions.

This paper proposes the idea of evidence chaining to maintain the validity of non-repudiation evidence efficiently during and after a transaction. We use the fair non-repudiation protocol presented in [19] as an example to show how the problem will be tackled.

The rest of the paper is organised as follows. In the next section, we briefly describe the fair non-repudiation protocol presented in [19]. In Section 3, we discuss the existing approaches to maintaining the validity of non-repudiation evidence. In Section 4, we propose an efficient scheme for implementation, analyse the validity of chained evidence and demonstrate the settlement of disputes. Section 5 concludes the paper.

In this paper, we use the following basic 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.
- $eK(X)$ and $dK(X)$: encryption and decryption 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. The algorithm is assumed to be a 'signature with appendix', and the message is not recoverable from the signature.
- $A \to B : X$: principal A dispatches message X addressed to principal B.

- $A \leftrightarrow B : X$: principal A fetches message X from principal B using *"ftp get"* operation [13] or by some analogous means (e.g. using a Web browser).

2 A Fair Non-repudiation Protocol

A fair non-repudiation protocol was proposed in [19], where the originator A and the recipient B communicate directly with light-weighted involvement of a trusted third party TTP. The main idea of this protocol is to split the definition of a message M into two parts, a commitment C and a key K. The commitment is sent from the originator A to the recipient B and then the key is lodged with the TTP. Both A and B have to retrieve the confirmed key from the TTP as part of the non-repudiation evidence required in the settlement of a dispute. 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.
- K: message key defined by A.
- $C = eK(M)$: commitment (ciphertext) for message M.
- $L = H(M, K)$: a unique label linking C and K.
- $f_{EOO}, f_{EOR}, f_{SUB}, f_{CON}$: flags 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:

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

In the above protocol, B can abort the protocol run without disputes before sending out EOR to A at Step 2; A can abort the protocol run without disputes before submitting sub_K and K to the TTP at Step 3. After receiving sub_K and K from A, the TTP will generate con_K and store the tuple

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

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. As we assume that the communication channels are not permanently broken, the confirmed key will always be available to A and B. Thus, at the end of a protocol run, the originator A will hold non-repudiation evidence *EOR* and *con_K* which can be used to prove that B received message M; meanwhile, the recipient B will obtain message $M = dK(C)$ and hold non-repudiation evidence *EOO* and *con_K* which can be used to prove that message M originated from A.

This protocol has three features which make it attractive in electronic businesses carried on the Internet.

- 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. No party can repudiate the message transfer once the message has been transferred; no party can obtain sufficient evidence to prove the origin and receipt of a message if the message is not transferred.
- The trusted third party's involvement is light-weighted. The *TTP* only needs to notarize message keys by request and provide directory services.

3 Approaches for Maintaining the Validity of Evidence

The protocol in Section 2 presents an elegant solution to fair non-repudiation. However, one issue related to the efficient implementation was not taken into consideration. In the protocol, non-repudiation evidence is generated by means of digital signature. In practice, a signature key may be compromised and a signature could be forged by using a compromised key. Therefore, the compromised key needs to be revoked [11, 14, 15] so that all signatures generated after the corresponding public key certificate has been revoked will be regarded as invalid. The remaining question is how to prove that a signature was generated before the corresponding public key certificate had been revoked and thus is deemed valid.

A simple approach for maintaining the validity of digital signatures relies on the existence of an *on-line* time-stamping authority as well as the certificate revocation infrastructure. For example, a user U's signature on a message X should be sent to a trusted time-stamping authority *TS* to certify that the signature was generated at the time of T_g.

$$\begin{array}{ll} 1.\ U \rightarrow TS: & sS_U(X) \\ 2.\ TS \rightarrow U: & T_g, sS_{TS}(sS_U(X), T_g) \end{array}$$

Thus, even if U's corresponding public key certificate is revoked for any reason after T_g, U's signature on X will be regarded as valid (provided that the time-stamping service is reliable [8]).

If this approach is applied to the protocol in Section 2, 4 extra messages have to be exchanged with *TS* to get 2 digital signatures (*EOO* and *EOR*) time-stamped in a protocol run - a heavy burden to both A and B as well as *TS*.

An efficient approach to securing digital signatures as non-repudiation evidence was proposed in [22], in which two different types of signature keys are defined.

- *Revocable signature keys* – the corresponding verification key certificates are issued by a *certification authority* (*CA*), and can be revoked as usual.
- *Irrevocable signature keys* – the corresponding verification key certificates are issued by users themselves and time-stamped by a *time-stamping authority* (*TS*). Such certificates cannot be revoked before their expiry.

The revocable signature key is used as a long-term master key to issue irrevocable verification key certificates while the irrevocable signature key is used as a temporary key to sign electronic documents. The digital signatures generated in such a way will remain valid until the corresponding irrevocable verification key certificates expire, thus can be exempted from being time-stamped by a time-stamping authority during on-line transactions.

The second approach can significantly improve the efficiency of *mass* on-line transactions. However, it does not help much for a *single* run of the protocol in Section 2 since both A and B still have to contact TS to get their own temporary irrevocable verification key certificates time-stamped before they can use the corresponding irrevocable signature keys to generate valid signatures.

The purpose of this paper is to provide an efficient approach to maintain the validity of non-repudiation evidence generated by the originator and the recipient without incurring extra messages in the implementation of a fair non-repudiation protocol.

4 An Efficient Scheme for Implementation

From the discussion in Section 3 we found that existing approaches for maintaining the validity of evidence in a fair non-repudiation protocol will result in more or less extra messages to be exchanged between evidence sender/receiver and a time-stamping authority. Obviously, this is less efficient for implementation. The idea of evidence chaining can be used to address such a problem [18].

The evidence chaining mechanism works by linking one piece of evidence to another to form a chain of evidence such that by validating the latest piece of chained evidence, the rest pieces of evidence that are linked together in the chain are also validated. The concept of chaining is not new, which can be found in one-time password authentication [10], time-stamping service [8], and micropayment scheme [12] etc. But the idea of postponing the validation of chained evidence until the end of a protocol run in order to minimize the interaction between the originator/recipient and a trusted third party is novel. Here we use the protocol described in Section 2 as an example to demonstrate how the mechanism works.

4.1 The Updated Protocol

With reference to the fair non-repudiation protocol in Section 2, the most critical step of the entire protocol is the submission of message key K to the TTP. This is because as long as K is not published in the TTP's read-only directory, the protocol is not considered as a successful completion (with fairness property maintained). Since the outcome of the protocol relies completely on the submission of message key K, we delay the validation of all evidence created before the key submission until the time when the TTP receives a request for key publication in the read-only directory. The updated protocol is as follows.

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

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

In the above protocol, $Cert_B$ is B's public key certificate that is used to generate EOR. T is the time that the message key is confirmed by the TTP and ready for collection from the TTP's read-only directory. The time stamp T plays two important roles in the protocol.

- The time of message transfer is notarised when the confirmed message key is ready for collection from the TTP's read-only directory at T;
- The chained evidence EOR and EOO are validated when $Cert_B$ is checked valid at T.

As B can invalidate EOR by revoking $Cert_B$ before the TTP generates con_K, the message key K should be protected against eavesdropping (e.g. encrypted using the TTP's public encryption key V_{TTP}) when submitted by A.

The major improvement to the original protocol is that non-repudiation evidence EOO, EOR and con_K are linked one by one and validated when the TTP generates con_K. The chained evidence will remain valid as long as the TTP's signature is valid. Hence, A and B need not contact the time-stamping authority to get EOO and EOR time-stamped during and after a protocol run.

4.2 Validity of Chained Evidence

If the above protocol run is complete, message M will be transferred fairly from A to B and both parties will hold the evidence regarding the message transfer. The formal analysis of the original protocol has been carried out in [16, 21] with different approaches. Here we examine the generation of chained evidence in the updated protocol which is a new feature added into the original protocol for the efficient implementation. We assume that each party either holds the public key certificates of other parties, or is able to retrieve them from a X.509 directory service [6].

After receiving $EOO = sS_A(f_{EOO}, B, L, C)$ from A at Step 1, B will use A's public key certificate $Cert_A$ to verify A's signature EOO. If successful, B will generate EOR which includes EOO, then send EOR to A. Here B need not check the validity of $Cert_A$.

It will be up to A whether or not to send chained EOO and EOR to the TTP for validation at Step 3. Once EOO is validated, A cannot deny it.

After receiving $EOR = sS_B(f_{EOR}, A, L, C, EOO)$ from B at Step 2, A will use B's public key certificate $Cert_B$ to verify B's signature EOR. If successful, A will generate sub_K which includes EOR and $Cert_B$, then send sub_K to the TTP. Here A need not check the validity of $Cert_B$. If B generates EOR with invalid $Cert_B$, the TTP will not confirm sub_K thus B cannot obtain M.

After receiving $sub\_K = sS_A(f_{SUB}, B, L, K, EOR, Cert_B)$ from A at Step 3, the TTP will perform the following checks:

- checking the validity of $Cert_A$;
- using $Cert_A$ to verify A's signature sub_K;
- checking the validity of $Cert_B$ which is included in sub_K.

Only if all of the above checks are successful, will the TTP generate

$$con\_K = sS_{TTP}(f_{CON}, A, B, L, K, EOR, T)$$

Here the TTP need not use $Cert_B$ to check EOR. Actually, the TTP is unable to check EOR since A does not supply the contents of EOR. A is responsible for providing matched EOR and $Cert_B$ in sub_K so that A will not leave itself in a disadvantageous position.

Claim 1. *Suppose the TTP is trusted to generate valid evidence con_K, and evidence con_K, EOR, EOO are chained in the above way, then EOR and EOO are valid as well.*

Proof: Suppose $Cert_A$ and $Cert_B$ are valid public key certificates of A and B respectively at T. Suppose EOR is checked invalid by the use of $Cert_B$. We examine the following possibilities.

- A generated sub_K by including mismatched EOR and $Cert_B$. However, this is not in favour of A because B can get K and thus $M = dK(C)$ after the TTP confirms A's key submission but A will not have valid evidence to prove that B received M. Hence, A should always submit matched EOR and $Cert_B$ to the TTP for confirmation.
- EOR was generated after $Cert_B$ had been revoked. However, the TTP will check the validity of $Cert_B$ before generating con_K. Hence EOR which is confirmed in con_K should be generated before $Cert_B$ is revoked.

Thus, EOR which is chained to con_K should be valid if con_K is valid.

Similarly, suppose EOO is checked invalid by the use of $Cert_A$. We examine the following possibilities.

- B generated EOR by including a bogus EOO. However, this is not in favour of B because B will not have valid evidence to prove that M is from A. Hence, B should always include correct EOO in EOR.

- EOO was generated with A's revoked key certificate $Cert'_A$. However, it is up to A whether or not to send such EOO (which is chained to EOR) to the TTP for validation at Step 3. If A requests the TTP to validate such EOO, A cannot deny the validity of EOO since only A has the right to make such a request. Of course, the TTP will authenticate A's request, i.e. check the validity of sub_K, before validating EOO.

Thus, EOO which is chained to EOR should be valid if con_K and EOR are valid. □

4.3 Settlement of Disputes

The goal of our protocol is to enable the settlement of disputes over the origin and receipt of a message M. When disputes arise, a judge will be invoked who evaluates the evidence held by the participants and determines the origin or receipt of the message.

Claim 2. *If the originator A or the recipient B can provide M, C, K, L, T, $Cert_A$, $Cert_B$, EOO, EOR, con_K to the judge, and the following checks are successful, then B cannot deny the receipt of M, and A cannot deny the origin of M.*

1. *The judge checks the TTP's signature $con\_K = sS_{TTP}(f_{CON}, A, B, L, K, EOR, T)$, and the validity of $Cert_B$ at T;*
2. *The judge uses $Cert_B$ to check B's signature $EOR = sS_B(f_{EOR}, A, L, C, EOO)$;*
3. *The judge uses $Cert_A$ to check A's signature $EOO = sS_A(f_{EOO}, B, L, C)$;*
4. *The judge checks $L = H(M, K)$;*
5. *The judge checks $M = dK(C)$.*

Proof: We examine what the judge will believe after verifying each item listed above. If the first check is positive, the judge believes that

(1.a) A submitted message key K with label L to the TTP for publication;

(1.b) K is available to B from the TTP;

(1.c) A requested for validating chained evidence EOR and EOO, and $Cert_B$ is B's valid public key certificate to be used to check EOR.

If the second check is positive, the judge believes that

(2.a) EOR is valid evidence chained to con_K;

(2.b) B received C with label L from A and is committed to retrieving the message key from the TTP.

If the third check is positive, the judge believes that

(3.a) EOO is valid evidence chained to EOR;

(3.b) A sent C as its commitment for the message with label L to B.

If the fourth check is positive, the judge believes that

(4.a) Label L is correctly constructed and random;

(4.b) Message key K in con_K and commitment C in EOR and EOO are uniquely linked by L.

If the final check is positive, the judge believes that

(5.a) M is the message represented by C and K in transmission.

Thus, with beliefs 4.a, 4.b, 5.a, the judge can conclude that the integrity of messages is satisfied. With beliefs 1.c, 2.a, 3.a, the judge can conclude that con_K, EOR and EOO are valid evidence. With beliefs 1.a and 3.b, the judge can conclude that A sent M to B. With beliefs 1.b and 2.b, the judge can conclude that B received M from A. $\square$

A framework for handling disputes in payment systems was proposed in [3]. The above verification rules may be used to construct a reference engine to facilitate automatic dispute resolution under the framework.

5 Conclusion

Non-repudiation is one of the essential security services in electronic businesses on the Internet. There are two critical issues in non-repudiation services.

- How to achieve the fairness between two transacting parties?
- How to maintain the validity of non-repudiation evidence?

Several fair non-repudiation protocols have been proposed, and some approaches for maintaining the validity of non-repudiation evidence exist. However, it is still less efficient for implementation by applying existing approaches to a single run of those non-repudiation protocols. This paper proposed a new approach to improve the efficiency of a single run of fair non-repudiation protocols. We used the fair non-repudiation protocol presented in [19] as an example to show how the above problem can be addressed with negligible increase of workload.

A prototype of the updated fair non-repudiation protocol has been implemented. The experiment result shows that a protocol run takes about 5 seconds on a DEC Alpha Server 4100 under normal circumstances. This speed for a fair business transaction providing both non-repudiation of origin and non-repudiation of receipt is feasible. The result also shows that the protocol is able to function under temporary network failure conditions, which is very important in real applications on the Internet.

Acknowledgements

We thank the anonymous referee for helpful comments. The research towards this paper was funded by the National Science and Technology Board of Singapore.

References

[1] S. G. Akl. *Digital signatures: a tutorial survey.* Computer, 16(2):15–24, February 1983.

[2] N. Asokan, V. Shoup and M. Waidner. *Asynchronous protocols for optimistic fair exchange.* Proceedings of 1998 IEEE Symposium on Security and Privacy, pages 86–99, Oakland, California, May 1998.

[3] N. Asokan, E. V. Herreweghen and M. Steiner. *Towards a framework for handling disputes in payment systems.* Proceedings of the Third USENIX Workshop on Electronic Commerce, Boston, Massachusetts, September 1998.

[4] F. Bao, R. H. Deng and W. Mao. *Efficient and practical fair exchange protocols with off-line TTP.* Proceedings of 1998 IEEE Symposium on Security and Privacy, pages 77–85, Oakland, California, May 1998.

[5] K. S. Booth. *Authentication of signatures using public key encryption.* Communications of the ACM, 24(11):772–774, November 1981.

[6] CCITT. *Recommendation X.509: The directory – Authentication framework.* November 1988.

[7] 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.

[8] S. Haber and W. S. Stornetta. *How to time-stamp a digital document.* Journal of Cryptology, 3(2):99–111, 1991.

[9] ISO/IEC 13888-3. *Information technology - Security techniques - Non-repudiation - Part 3: Mechanisms using asymmetric techniques.* ISO/IEC, 1997.

[10] L. Lamport. *Password authentication with insecure communication.* Communications of the ACM, 24(11):770–772, November 1981.

[11] M. Naor and K. Nissim. *Certificate revocation and certificate update.* Proceedings of 7th USENIX Security Symposium, San Antonio, Texas, USA, January 1998.

[12] T. P. Pedersen. *Electronic payments of small amounts.* Lecture Notes in Computer Science 1189, Proceedings of Cambridge Workshop on Security Protocols, pages 59–68, Cambridge, April 1996.

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

[14] M. Roe. *Cryptography and evidence.* PhD Thesis, University of Cambridge, 1997.

[15] R. Rueppel. *Revocation and revocation certificates.* Proceedings of the Trusted Third party Workshop, pages 1–8, Barcelona, Spain, February 1995.

[16] S. Schneider. *Formal analysis of a non-repudiation protocol.* Proceedings of 11th IEEE Computer Security Foundations Workshop, pages 54–65, Rockport, Massachusetts, June 1998.

[17] B. Schneier. *Applied cryptography – Protocols, algorithms, and source code in C.* New York: John Wiley & Sons, 1996 (second edition).

[18] C. H. You. *Practical light-weighted non-repudiation protocols.* MSc Thesis, National University of Singapore, 1998.

[19] 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.

[20] J. Zhou and D. Gollmann. *Evidence and non-repudiation.* Journal of Network and Computer Applications, 20(3):267–281, London: Academic Press, July 1997.

[21] J. Zhou and D. Gollmann. *Towards verification of non-repudiation protocols.* Proceedings of 1998 International Refinement Workshop and Formal Methods Pacific, Canberra, Australia, September 1998.

[22] J. Zhou and K. Y. Lam. *Securing digital signatures for non-repudiation.* (manuscript)