On Timeliness of a Fair Non-repudiation Protocol

Li Botao Luo Junzhou
Department of Computer Science and Engineering, Southeast University, Nanjing 210096, China.
Telephone: +86-25-83795595
Email: {leopert, jluo}@seu.edu.cn

ABSTRACT

In 1996, J. Zhou and D. Gollmann proposed a fair non-repudiation protocol, which was unfair in fact for its lack of timeliness. K. Kim, S. Park and J. Baek improved it by adding time limit information to protocol messages. However, the improvement needs mechanisms to synchronize clocks among the protocol entities, thus making its implementation inefficient. In this paper, to make Zhou-Gollmann's non-repudiation protocol provide timeliness while preserving its efficiency, a more efficient synchronization scheme was introduced to improve it. SVO logic was extended slightly to analyze the timeliness of the new protocol. The analysis result shows that while the original protocol does not provide timeliness, the newly updated one does.

Categories and Subject Descriptors

C.2.2 [COMPUTER-COMMUNICATION NETWORKS]: Network Protocols
F.4.1 [MATHEMATICAL LOGIC AND FORMAL LANGUAGES]: Mathematical Logic *-Modal logi*
K.4.4 [COMPUTERS AND SOCIETY]: Electronic Commerce *-Security*
K.6.5 [MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS]: Security and Protection *-Authentication*

General Terms

Design, Security, Theory, Verification.

Keywords

Non-repudiation, Fairness, Timeliness, Clock Synchronization, SVO logic

1. INTRODUCTION

With the explosion of electronic businesses on the Internet in recent years, non-repudiation in transactions has become a more and more important problem, and many non-repudiation protocols have been presented for it. The purpose of non-repudiation is to collect, maintain, make available and validate irrefutable evidence concerning a claimed event or action in order to resolve disputes on the occurrence or non-occurrence of the event or action [1]. There are two non-repudiation services to be provided in most communications [2]:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
InfoSecu'04, Nov 14-16, 2004, Shanghai, China.
Copyright 2004 ACM ISBN: 1-58113-955-1

Non-repudiation of Origin, which guarantees that the originator of a message cannot later falsely repudiate having originated that message.
Non-repudiation of Receipt, which guarantees that the recipient of a message cannot later falsely repudiate having received that message.

While some non-repudiation protocols provide only one of the two services, most of them are intended to provide both. To achieve this goal, a non-repudiation protocol should have the following necessary properties:

Fairness. A protocol is believed to be fair if it can ensure that, at the end of a protocol execution, none or both of the two entities, the sender and the receipt, can receive all the evidences it expect.
Timeliness. A non-repudiation protocol provides timeliness if and only if all honest parties always have the ability to reach, in a finite amount of time, a point in the protocol where they can stop the protocol while preserving fairness [3].

Apart from the above two properties, an applicable non-repudiation protocol should be designed to be as efficient and easy to deploy as possible [4].

In 1996, J. Zhou and D. Gollmann presented a fair non-repudiation protocol (We call it ZG protocol later in this paper) in [5]. In ZG protocol, an on-line TTP (Trusted Third Party) [3][6] is used and the message is sent in two parts, an encryption key and the ciphertext. At first, the originator sends the ciphertext to the recipient, who replies it with an acknowledgement. The originator then sends the key to a TTP who then publishes it in its public directory for the recipient to retrieve. ZG protocol was widely discussed for its simplicity and efficiency after it was published. For example, K. Kim, S. Park and J. Baek discussed its timeliness and confidentiality problems [7]. S. Kremer and O. Markowitch extended it into a multi-party non-repudiation protocol [8]. J. Zhou and D. Gollmann did a formal verification of it by using SVO logic [9]. S. Schneider analyzed it with CSP [10].

In [7], K. Kim, S. Park and J. Baek pointed out that ZG protocol did not provide timeliness, thus was unfair in fact. They improved it by adding time limit information into the protocol messages. The updated protocol is fair, but it is inefficient and less applicable in that it needs some mechanism, such as network time service, to synchronize clocks between the three entities of the protocol.

In this paper, a novel and efficient synchronization scheme was introduced to improve ZG protocol (the newly improved protocol is referred to as NZG later). The scheme needs no clock synchronization between entities, and the implementation of NZG protocol is much more efficient than that of K. Kim et al's improvement. To analyze timeliness of the non-repudiation protocol, a BAN-class logic SVO was extended by adding a time expression into it, and the updated protocol was verified with the extended logic. The result of the analysis shows that NZG protocol does 'provide timeliness.

The paper is structured as follow. Section 2 discusses ZG protocol, its timeliness flaw and existent improvements. In Section 3, the new improvement of ZG protocol is given. Section 4 extends SVO logic and analyzes NZG protocol with the newly extended logic. Some problems and solutions in implementation are discussed in Section 5. The last section concludes the paper.

2. ZG NON-REPUDIATION PROTOCOL

ZG non-repudiation protocol is a non-repudiation protocol using on-line TTP. The protocol is famous for its simplicity and efficiency. Although ZG is claimed to fair, it is unfair in fact for its lack of timeliness.

2.1 Basic Notation

The following are some of the notation that will be used throughout the paper.

- A, B: originator and recipient of the message.
- TTP: online trusted third party providing network services accessible to the public.
- J: judge to resolve disputes.
- $A \rightarrow B$: X: principal A sends message X to principal B.
- $A \leftarrow B$: X: principal A fetches message X from principal B, for example, by FTP.
- X, Y: concatenation of two messages X and Y.
- $eK(X)$, $dK(X)$: encryption and decryption of message X with key K using symmetric cryptosystem.
- $sK(X)$: digital signature of message X with the private key K.
- P_X, S_X: the public and private key of principal X
- M: message sent from A to B.
- K: message key defined by A.
- C: ciphertext for message M, namely $C = eK(M)$.
- L: unique label linked to all messages of a particular protocol run.
- f_{NRO}, f_{NRR}, f_{SUB}, f_{CON}: flag information indicating Non-Repudiation of Origin, Non-Repudiation of Receipt, Submission of a key, Confirmation of a key.

2.2 ZG Protocol

In ZG protocol, a message M is divided into two parts: an encryption key K and the ciphertext C. At first, the originator A sends the ciphertext C to the recipient B. B replies it with an acknowledge NRR. A then sends the key K to the TTP, who publishes it and the evidence con_K in its public directory. B can then get K from TTP's public directory to decrypt the ciphertext C, and A can get con_K and store it with the receipt NRR.

It is assumed in ZG protocol that the network between TTP and respectively A and B are eventually available, namely they are resilient communication channels that deliver correct data after a finite, but unknown amount of time. With them, once TTP publishes something in it public directory, it can be asserted that A and B can eventually get it. The protocol steps are as follow:

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

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. $B \leftarrow TTP$: $f_{CON}, A, B, L, K, con\_K$
5. $A \leftarrow TTP$: $f_{CON}, A, B, L, K, con\_K$

Disputes can arise over the origin and receipt of the message M. In these cases, an adjudicator J can be invoked to resolve them.

Non-repudiation of Origin. If A denies having sent M to B, B can submit M, NRO and con_K to J, who does the following verifications to check if A has sent M to B:

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

If all the three checks are positive, J then believes that A has sent message M to B.

Non-repudiation of Receipt. If B claims that it has not received M from A, A submits M, NRR and con_K to J. J does the following verifications to check whether B has received M from A:

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

If all these checks are positive, J will believe that B has received M from A.

The features of ZG protocol are: a) Efficient. The protocol has only five communication steps; b) Feasible. The initial assumptions of the protocol can be satisfied in most networks in the real world; c) On-line TTP. The protocol is the first non-repudiation protocol using on-line TTP rather than in-line TTP.

2.3 The Flaw and Existent Improvements

Although ZG protocol seems fair, it is not. After step 2, A can stop the protocol run by not submitting sub_K to TTP, but B has to keep C and NRO that it has received from A forever. If B deletes all these information, A can then send sub_K to TTP, who will still put con_K to its public directory. Then A can get all the non-repudiation evidence to prove that B has received the message M, but B cannot get M, since C is already deleted. According to the definition of timeliness, the protocol does not provide timeliness.

To prevent A from being in an advantageous position over B, J. Zhou and D. Gollmann provided an updated of the protocol in which time limit information was added into the protocol messages. The updated protocol is as follow:

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

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

5. $A \leftarrow TTP$: $f_{CON}, A, B, L, K, T, T_0, con\_K$

In the updated protocol, T is a deadline before which A and B can get *con_K* and, *TTP* will delete *con_K* from its public directory at T. T_0 is the actual time *TTP* publishes *con_K*. If B does not agree on the deadline T defined by A, it can stop the protocol run by not sending *NRR* to A.

K. Kim, S. Park and J. Baek pointed out that Zhou-Gollmann's improvement was incomplete. If A sends K to *TTP* just before the deadline T, *TTP* will delete *con_K* shortly after it is published. Then B has to keep on retrieving *TTP*'s public directory around the time T. A can then disturb the network or B's computer system to prevent B from receiving *con_K*. K. Kim et al provided a complete improvement:

$NRO = sS_A(f_{NRO}, B, L, T, C)$
$NRR = sS_B(f_{NRR}, A, L, T, T_1, C)$
$sub\_K = sS_A(f_{SUB}, B, L, T, K)$
$con\_K = sS_T(f_{CON}, A, B, L, T, T_0, K)$

1. $A \rightarrow B$: f_{NRO}, B, L, T, C, NRO
2. $B \rightarrow A$: f_{NRR}, A, L, T_1, NRR
3. $A \rightarrow TTP$: $f_{SUB}, B, L, T, K, sub\_K$
4. $B \leftarrow TTP$: $f_{CON}, A, B, L, T, T_0, K, con\_K$
5. $A \leftarrow TTP$: $f_{CON}, A, B, L, K, T, T_0, con\_K$

The purpose of T and T_0 is the same with those in Zhou-Gollmann's improvement. T_1 is a deadline defined by B before which A must send K to *TTP*. If there is enough time between T_1 and T, B can eventually retrieve *con_K* from *TTP*'s public directory at its convenient time between T_1 and T. When disputes rise, they can be resolved by the judge J. For example, when B denies receiving M from A, A can submit M, *NRR* and *con_K* to J, who does the following verifications to check whether B has received M from A or not:

- Checks that *con_K* is *TTP*'s signature on $(f_{CON}, A, B, L, T, T_0, K)$.
- Checks that *NRR* is B's signature on $(f_{NRR}, A, L, T, T_1, C)$.
- Checks that $T_0 < T_1 < T$.
- Checks that $M = dK(C)$.

While the improved protocol provides timeliness and fairness, it is inefficient. Firstly, because time information is used in the protocol, the protocol is heavily dependent on the time information. When implementing it, there must be a mechanism, such as network time service, to synchronize the clocks of the entities in the protocol. This does not add more restriction on working environments of the protocol, but also increases the communication costs. Secondly, the protocol must be revised to use more efficient evidence management mechanism than timestamp service. To make ZG protocol fair while preserving its efficiency, we use a more efficient synchronization scheme to improve ZG protocol as described in the next section.

3. THE NEW IMPROVEMENT

We also use time limit information to resolve the timeliness problem of ZG. However, unlike K. Kim et al's improvement, our approach needs no clock synchronization mechanism. Two concepts must be made clear in the new improvement:

Time point: a point on the time axis. Time point is represented with uppercase letter T (maybe with subscript).

Time span: the length of the time between two time points, namely the difference between two time points. Time span is represented with lowercase letter t (maybe with subscript).

3.1 The Protocol Steps

The protocol communication steps are as follow:

$NRO = sS_A(f_{NRO}, B, L, C)$
$NRR = sS_B(f_{NRR}, A, L, t_B, NRO)$
$sub\_K = sS_A(f_{SUB}, B, L, t_A, K, NRO)$
$con\_K = sS_T(f_{CON}, A, B, L, T, t_A, t_B, K, NRO, NRR)$

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

In the above, t_A is a time span defined by A, indicating that *sub_K* will be kept in *TTP*'s private directory for t_A time units. t_B is a time span defined by B, indicating *TTP* will keep *NRR* in its private directory for t_B time units. T is a time point indicating the actual time *TTP* makes *con_K* publicly available. In the new protocol, step 2 can be carried out before step 1 or after step 3 without affecting the result of the protocol.

In the protocol, there is a publicly known predefined time span t. It is the time length for which *TTP* will keep *con_K* in its public directory. As in ZG, we still assume that the communication channels between *TTP* and respectively A and B are resilient. Thus, once *TTP* publishes the evidence *con_K*, A and B can eventually get it in a specified finite amount of time t. Although we do not need to synchronize the clocks among A, B and *TTP*, we *do* need to assume that the difference between the clock speeds of the three entities is little enough to be ignored. This assumption can be easily satisfied for most applications on computers nowadays.

The execution of each step of NZG is discussed below. We do not consider the problem of network delay in the discussion, since it should only be dealt with in implementation.

1. $A \rightarrow B$: if B wants message M, it must carry out step 3 to requesting *TTP* to generate *con_K*.

2. $A \rightarrow TTP$: After receiving *sub_K*, *TTP* keeps it in its private directory and, deletes it from there after t_A time units or until con_K is generated and published.

3. $B \rightarrow TTP$: After receiving *sub_K*, *TTP* keeps it in its private directory and, deletes it from there after t_A time units or until con_K is generated and published.

Generation of *con_K*: If there is some time point T at which both *sub_K* and *NRR* exist in *TTP*'s private directory, *TTP* can then begin the process of generating *con_K*. *TTP* compares the *NRO* in *sub_K* with that in *NRR* at first. If the two *NRO*s are not the same, *TTP* thinks one of A and B is cheating, it will not generate *con_K*; otherwise, *TTP* generates *con_K* and put the time point T into it, and publishes it in its public directory. After *con_K* has been published for t time units, *TTP* will delete it from the public directory. The value of the time span t should be large enough to ensure that A and B can always get *con_K* as long as it is published. Assuming the network between *TTP* and A (or B) will never be broken for more than t_d time units, t can be defined as $t_d + x$ ($x > 0$).

4. ***A* ← *TTP***: After sends *sub_K* to *TTP*, *A* keeps on retrieving *con_K* in *TTP*'s public directory. If, after $t_A + t$ time units, *A* still does not get *con_K*, it will stop retrieving, considering that *B* has not sent *NRR* to *TTP*. Otherwise, *A* can now prove that *B* has received *K* from *TTP* in between *T* and $T + t$. *A* then keeps all the evidences *NRO*, *NRR* and *con_K* for later use in dispute resolution.

5. ***B* ← *TTP***: Just like *A*, after it has sent *NRR* to *TTP*, *B* keeps on retrieving in *TTP*'s public directory until it gets *con_K* or $t_B + t$ time units pass. If it does not get *con_K* after t_B time units, *B* deletes *NRO* that it has received from *A* and stops retrieving. Otherwise, B can then decrypt *C* with *K* to recover *M* and, with *NRO*, *NRR* and *con_K*, *B* can prove that *M* is originated by *A*. The evidences *NRO*, *NRR* and *con_K* can be kept for later use in dispute resolution.

Figure 1 shows an NZG protocol run instance.

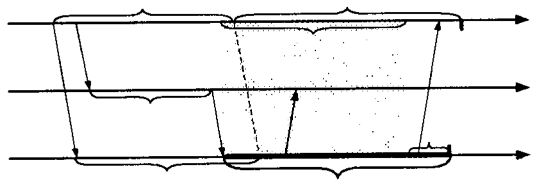

Figure1. An NZG protocol run instance

It is shown how the protocol controls the time between all protocol events. There are three time axes in the figure, each of them representing the time of a principle. On the time axis of *TTP*, there is a thick segment indicating the time *con_K* exists in *TTP*'s public directory. In the protocol run, after receiving *NRO* from *A*, *B* does not send *NRR* until t_A is to be passed and, after sending *NRR*, *B* begins to disturb *A* or its network, preventing it from receiving *con_K* from *TTP*. However, *B*'s disturbance cannot be longer than t_d, so *A* still has enough time to fetch *con_K* after *B*'s disturb ends.

3.2 Dispute Resolution

Non-repudiation of Origin. *B* submits *M*, *NRO*, *NRR* and *con_K* to the judge *J*, who does the following checks to determine whether *A* has sent *M* to *B* or not.

- Checks that *con_K* is *TTP*'s signature on (f_{CON}, *A*, *B*, *L*, *T*, t_A, t_B, *K*, *NRO*, *NRR*)
- Checks that *NRR* is *B*'s signature on (f_{NRR}, *A*, *L*, t_B, *NRO*)
- Checks that *NRO* is *A*'s signature on (f_{NRO}, *B*, *L*, *C*)
- Checks that $M = dK(C)$

the key *K* to *TTP*. If the second check is positive, *J* can believe *B* has sent correct data to *TTP*. If the third check is positive, *J* can believe *A* has sent C to *B*. *J* will hold *B*'s claim if the final check is also positive.

Non-repudiation of Receipt. *A* submits to *J* the same data with those *B* submits to *J* in non-repudiation of origin. *J* does the same checks to determine whether *B* has received *M*.

4. FORMAL ANALYSIS OF TIMELINESS

NZG protocol modifies not only time information in messages of ZG protocol, but also sequence of its exchange steps. Therefore, it is necessary to verify NZG protocol formally to check its correctness and, more importantly, check whether it can provide timeliness.

There are many formal methods to be used to verify security protocols. Among them is SVO logic [11][12], which is widely used for its simplicity. SVO logic is an extension of BAN logic [13]. The goal of non-repudiation protocols can be easily expressed in SVO logic formulae. ZG protocol has been verified with SVO logic by J. Zhou and D. Gollmann [9], but they did not find its timeliness flaw. This is because there is no approach to describe time relation of protocol events in SVO logic. Therefore, timeliness cannot be described and analyzed with SVO logic. To analyze timeliness of NZG protocol, a time expression and analysis approach are added to SVO logic in this section. Then the extended logic is used to analyze NZG protocol.

4.1 Extending SVO Logic

In SVO logic, time is divided into two parts by formula *fresh*: before the current protocol run and after the protocol run begins. To describe occurrence time of protocol events (message receiving and sending), we add a condition into the definition of formula language:

P received X at T, *P says X at T*, *P said X at T* are formulae, if *X* is a message, *P* is a participant and *T* is a time expression.
Let $I = \{0, 1, 2, 3, \ldots\} \cup \{-1, -2, -3, \ldots\}$, time expression is defined as follow:
1. x is a time constant if $x \in I$;
2. X is a time variable if X is a variable whose value range is I;
3. $[X \mid TS]$ is a time expression if X is a time variable and $TS \subseteq I$.

The value of a time expression $[X \mid TS]$ is any $x \in TS$, and the variable X is bound with the same value. Once a variable is bound with some value, it can only be used as a constant until the binding is released. This is somewhat like the variable binding concept in the programming language *prolog* [14]. Some specific time expression in formulae can be abbreviated. $[X \mid I]$ can be written as $[X]$, and $[X \mid \{x\}]$ can be written as $[x]$, if x is a time constant or a bound variable.

Operator *at* is used to express the occurrence time of a protocol event. Accurately, according to the semantics of SVO logic, φ *at* T holds iff $(r, T) \models \varphi$. The use of *at* is optional. For those events whose occurrence time is critical, the operator can be used, but it is not needed for those events whose occurrence time does not affect the protocol's goal. After operator *at* is added, we need to extend some of the axioms of SVO logic as follow:

A4. $(PK_\sigma(Q, K) \wedge R\ \textit{received}\ \{X\}_{K^{-1}}\ \textit{at}\ T) \supset Q\ \textit{said}\ X\ \textit{at}\ T$
A6. $P\ \textit{received}\ (X_1, \ldots, X_n)\ \textit{at}\ T \supset P\ \textit{received}\ X_i\ \textit{at}\ T$
A7. $(P\ \textit{received}\{X\}_K\ \textit{at}\ T \wedge P\ \textit{has}\ K') \supset P\ \textit{received}\ X\ \textit{at}\ T$
A13. $P\ \textit{said}\ (X_1, \ldots, X_n)\ \textit{at}\ T \supset (P\ \textit{said}\ X_i\ \textit{at}\ T \wedge P\ \textit{sees}\ X_i)$

Here, T is a time expression. The syntax analysis technique of SVO logic should also be extended:

1. Before setting out premises, one should define all time constants and variables at first. In the process of proof, a time

variable is bound with some value at the first time it appears in a formula. In formulae for protocols goals, time restriction condition of the communication event should be given. For example, for non-repudiation of receipt, the protocol goal may be written as:

$J \mathit{\ believes\ } (B \mathit{\ received\ } M \mathit{\ at\ } [T]) \wedge (T \leq t)$

This formula says that B must have received the message M before time t (t is a time constant). Therefore, the protocol goal formula consists of two parts: the first part is about the occurrence of the event and the second part is about the time when the event has occurred.

2. The proof of protocol goals is composed of two steps. The first step is called logic proof whose purpose is to prove the first part of the goal formulae, namely whether the judge can be convinced that the specified event has occurred. A logic proof is a sequence of formulae in the logic. Each line is either a premise, an axiom, or derivable from preceding lines via inference rules [11]. The second step is called time calculation. It uses algebraic method to prove the second part of the goal formulae.

Now the description of the extension ends. Because the extension adds more restriction on SVO logic, those formulae that cannot be deduced in the original logic also cannot be deduced in the extended one, but some formulae that hold in the original logic may not hold in the new one. Therefore, the extension will not degrade the reliability of the analysis results of a protocol. In the contrast, it extends the expression and analysis ability of SVO logic. We will discuss the correctness of the extension later.

4.2 Analyzing NZG Protocol

For the simplicity and readability of the analysis, we define the following abbreviation:

$C = \{M\}_K$

$NRO_p = \{f_{NRO}, B, L, C\};$ $\qquad NRO = \{NRO_p\}_{Ka^{-1}}$

$NRR_p = \{f_{NRR}, A, L, t_B, NRO\};$ $\qquad NRR = \{NRR_p\}_{Kb^{-1}}$

$sub\_K_p = \{f_{SUB}, B, L, t_A, K, NRO\};$ $\qquad sub\_K = \{sub\_K_p\}_{Ka^{-1}}$

$con\_K_p = \{f_{CON}, A, B, L, T, t_A, t_B, K, NRO, NRR\};$

$con\_K = \{con\_K_p\}_{Kt^{-1}}$

The following time constants will be used: T, t_A, t_B and t. Their values are those of the fields with the same name in the protocol messages. Time variables that will be used are: T_x, T_y, T_o, T_r, T_s, T_A and T_B.

The following are all premises about NZG protocol:

P1. (1) $J \mathit{\ believes\ } PK_\sigma(A, K_a)$ (2) $J \mathit{\ believes\ } PK_\sigma(B, K_b)$ (3) $J \mathit{\ believes\ } PK_\sigma(TTP, K_t)$

P2. (1) $J \mathit{\ believes\ } (B \mathit{\ has\ } K_a)$ (2) $J \mathit{\ believes\ } (B \mathit{\ has\ } K_t)$

P3. $J \mathit{\ believes\ } J \mathit{\ received\ } \{NRO, NRR, con\_K\}$

P4. $J \mathit{\ believes\ } (TTP \mathit{\ said\ } con\_K_p \supset TTP \mathit{\ said\ } con\_K_p \mathit{\ at\ } [T])$

P5. $J \mathit{\ believes\ } (TTP \mathit{\ said\ } con\_K_p \mathit{\ at\ } [T_x] \supset TTP \mathit{\ received\ } sub\_K \mathit{\ at\ } [T_y \mid \{x \mid T_x - t_A \leq x \leq T_x\}])$

P6. $J \mathit{\ believes\ } (TTP \mathit{\ said\ } con\_K_p \mathit{\ at\ } [T_x] \supset TTP \mathit{\ received\ } NRR \mathit{\ at\ } [T_y \mid \{x \mid T_x - t_B \leq x \leq T_x\}])$

P7. $J \mathit{\ believes\ } (TTP \mathit{\ said\ } con\_K_p \mathit{\ at\ } [T_x] \supset A \mathit{\ received\ } con\_K \mathit{\ at\ } [T_y \mid \{x \mid T_x \leq T_y \leq T_x + t\}])$

P8. $J \mathit{\ believes\ } (TTP \mathit{\ said\ } con\_K_p \mathit{\ at\ } [T_x] \supset B \mathit{\ received\ } con\_K \mathit{\ at\ } [T_y \mid \{x \mid T_x \leq T_y \leq T_x + t\}])$

P9. $J \mathit{\ believes\ } (B \mathit{\ said\ } NRR_p \mathit{\ at\ } [T_x] \supset B \mathit{\ received\ } NRO \mathit{\ at\ } [T_y \mid \{x \mid x \leq T_x\}])$

P10. $J \mathit{\ believes\ } (A \mathit{\ said\ } C \mathit{\ at\ } [T_x] \wedge A \mathit{\ said\ } K \mathit{\ at\ } [T_y] \supset A \mathit{\ said\ } M \mathit{\ at\ } [max(T_x, T_y)])$

P11. $J \mathit{\ believes\ } (B \mathit{\ received\ } C \mathit{\ at\ } [T_x] \wedge B \mathit{\ received\ } K \mathit{\ at\ } [T_y] \supset B \mathit{\ received\ } M \mathit{\ at\ } [max(T_x, T_y)])$

P1 and P2 are the assumptions on an available asymmetric key service such as PKI.
P3 says that A or B submit their evidences to J for judgment.
P4 says that the time when the evidence *con_K* is published is included in it.
P5 and P6 say that *TTP* publishes evidence *con_K* only if it receives all required messages in correct time.
P7 and P8 say that, once *TTP* publishes the evidence, A and B can eventually get it in a finite amount of time t.
P9 says that B does not act against its own interests. It sends *NRR* only after it has received *NRO*.
P10 and P11 say that the key K and ciphertext C together can determine the message M.

The goal of NZG protocol is to guarantee that, after a protocol run ends successfully, both the originator and the recipient can collect adequate evidence to prove the occurrence of the communication event to the judicator in dispute resolution and that the time relation between protocol steps satisfies a specified condition:

G1. $J \mathit{\ believes\ } (A \mathit{\ said\ } M \mathit{\ at\ } [T_x] \wedge A \mathit{\ received\ } con\_K \mathit{\ at\ } [T_y]) \wedge (T_x \leq T_y \leq T_x + t_A + t)$

G2. $J \mathit{\ believes\ } (B \mathit{\ received\ } M \mathit{\ at\ } [T_x] \wedge B \mathit{\ said\ } NRR_p \mathit{\ at\ } [T_y]) \wedge ((T \leq T_x \leq T + t) \wedge (T_x - t_B - t \leq T_y \leq T_x))$

G1 says that, if A has sent M, A must have received the evidence *con_K* in $t_A + t$ time units after it sent *sub_K* to *TTP*. G2 says that, if B has received M at some time T_x, then the *NRR* that B has submitted to *TTP* must have been sent after $T_x - t_B - t$.

In the proof of the two goals, we simplified the logic proof part by grouping several lines as one, for the use of SVO logic has been widely illustrated elsewhere other than in this paper. Our purpose is to show how timeliness is analyzed.

G1. $J \mathit{\ believes\ } (A \mathit{\ said\ } M \mathit{\ at\ } [T_x] \wedge A \mathit{\ received\ } con\_K \mathit{\ at\ } [T_y]) \wedge (T_x \leq T_y \leq T_x + t_A + t)$

Logic proof:

1. $J \mathit{\ believes\ } (TTP \mathit{\ said\ } con\_K_p \mathit{\ at\ } [T])$	P1(3), P3, P4, Nec, A1, A4, A6
2. $J \mathit{\ believes\ } (A \mathit{\ said\ } K \mathit{\ at\ } [T_s \mid \{x \mid T - t_A \leq x \leq T\}])$	1, P1(1), P5, Nec, A1, A4, A13
3. $J \mathit{\ believes\ } (B \mathit{\ said\ } NRR_p \mathit{\ at\ } [T_r \mid \{x \mid T - t_B \leq x \leq T\}])$	1, P1(2), P6, Nec, A1, A4
4. $J \mathit{\ believes\ } (B \mathit{\ received\ } NRO \mathit{\ at\ } [T_o \mid \{x \mid x \leq T_r\}]$	3, P9, A1

5. J *believes* (A *said* C *at* $[T_o]$)	4, P1(1), Nec, A1, A4, A13
6. J *believes* (A *said* M *at* $[max(T_o, T_s)]$)	2, 5, P10, A1
7. J *believes* (A *received* con_K *at* $[T_A \mid \{x \mid T \le x \le T + t\}]$)	1, P7, A1
8. J *believes* (A *said* M *at* $[max(T_o, T_s)] \land A$ *received* con_K *at* $[T_A]$)	6, 7, Nec, A1

Time Calculation:

Let $T_x = max(T_o, T_s)$ and $T_y = T_A$. Now we begin to prove $T_x \le T_y \le T_x + t_A + t$:

$T_y = T_A$		
$\Rightarrow T \le T_y \le T + t$	(1)	$\because T_A \in \{x \mid T \le x < T + t\}$
$T_x = max(T_o, T_s)$		
$\Rightarrow T - t_A \le T_x \le T$		$\because T_o \le T_r \le T$ and $T - t_A \le T_s \le T$
$\Rightarrow T_x \le T \le T_x + t_A$	(2)	
$\Rightarrow T_x \le T_y \le T_x + t_A + t$		$\because$ (1)and (2)

Therefore, G1 is proven.

G2. J *believes* (B *received* M *at* $[T_x] \land B$ *said* NRR_p *at* $[T_y]$) $\land ((T \le T_x \le T + t) \land (T_x - t_B - t \le T_y \le T_x))$

Logic proof:

1. J *believes* (TTP *said* con_K_p *at* $[T]$)	P1(3), P3, P4, Nec, A1, A4, A6
2. J *believes* (B *said* NRR_p *at* $[T_r \mid \{x \mid T - t_B \le x \le T\}]$)	1, P1(2), P6, Nec, A1, A4
3. J *believes* (B *received* NRO *at* $[T_o \mid \{x \mid x \le T_r\}]$	3, P9, A1
4. J *believes* (B *received* C *at* $[T_o]$	3, P9, A1
5. J *believes* (B *received* con_K *at* $[T_B \mid \{x \mid T \le x \le T + t\}]$)	1, P7, A1
6. J *believes* (B *received* K *at* $[T_B]$)	1, P7, A1
7. J *believes* (B *received* M *at* $[max(T_o, T_B)]$	6, 7, Nec, A1
8. J *believes* (A *said* M *at* $[max(T_o, T_B)] \land B$ *said* NRR_p *at* $[T_r]$)	6, 7, Nec, A1

Time Calculation:

Let $T_x = max(T_o, T_B)$ and $T_y = T_r$. Now we begin to prove $(T \le T_x \le T + t) \land (T_x - t_B - t \le T_y \le T_x)$:

$T_x = max(T_o, T_B)$		
$\Rightarrow T_x = T_B$		$\because T_o \le T_r \le T$ and $T \le T_B \le T + t$
$\Rightarrow T \le T_x \le T + t$		
$\Rightarrow T_x - t \le T \le T_x$	(1)	
$T_y = T_r$		
$\Rightarrow T - t_B \le T_y \le T$		$\because T_r \in \{x \mid T - t_B \le x \le T\}$
$\Rightarrow T_x - t_B - t \le T_y \le T_x$		$\because$ (1)

Therefore, G2 is proven.

4.3 Discussions on Analysis Result

The analysis above shows that NZG protocol can provide timeliness. After applying the same analysis approach to ZG protocol, we found that ZG protocol is indeed lack of timeliness.

However, is the analysis itself correct? Now, since the semantics of the operator *at* has been set, we only need to examine the correctness of the extended axioms A4, A6, A7 and A13.

Since there is no key agreement process in ZG and NZG protocol, for any $t \ge 0$, $(r, t) \models PK_\sigma(Q, K)$ if $PK_\sigma(Q, K)$ holds,. And it is the same for P *has* K'. According to the semantics of SVO logic, the truth conditions of the extended A4, A6, A7 and A13 are the same with those of the origin axioms.

The time calculation part is based on set and algebra theory, it is safe to use them. Therefore, the analysis in the paper is dependable.

5. CONSIDERATIONS ON IMPLEMENTATION

While NZG protocol changes the communication steps of ZG protocol to provide timeliness, it also introduces some problems. They must be noticed and dealt with in implementation. In this section, we list some of them and give solutions for them.

5.1 Evidence Management

Evidence management is important for the performance of the implementation of non-repudiation protocols. When all communication steps are carried out, the protocol run is finished for normal cryptographic protocols. However, non-repudiation protocols are not finished only after the communication steps, entities should keep the evidence collected in the protocol run for later use in dispute resolution. Because non-repudiation evidence is often produced by digital signature and the signing private key may be crashed or expired, it must be verified when resolving disputes whether the signature is produced before the signing private key becomes invalid. The common method to maintain the validity of digital signatures is to use timestamp service as well as certificate authority. The evidence producer interacts with an on-line timestamp authority to get a timestamp for every piece of evidence. However, using timestamp service is inefficient, since it needs two more communication steps for every piece of evidence. There are some more efficient evidence management schemes for different situations. For example, J. Zhou and K. Lam designed an evidence management mechanism for non-repudiation protocols without TTP [16]. For non-repudiation protocols using TTP, CH. You et al presented a new approach, called evidence-chaining and revised ZG protocol to suite the scheme [17]. The idea of evidence-chaining mechanism is to link 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. We can use the evidence-chaining mechanism in NZG protocol without any revision.

In dispute resolution of NZG protocol, the judge J needs to validate evidences NRO, NRR and con_K. The ascending order of the time when they are produced is $NRO \rightarrow NRR \rightarrow con\_K$ and, the three pieces of evidence have formed an evidence chain: NRO is included in NRR and NRR in con_K. The time information T in evidence con_K indicates the actual time con_K is produced. Therefore, once J can be sure that TTP's signing private key is valid before T, it can believe con_K is valid. Since TTP must validate NRO and NRR before it publishes con_K, J can also take NRO and NRR as valid, for they are produced before con_K.

The use of evidence-chaining scheme does not only simplify the evidence management and improve the protocol's efficiency; it also lightens burdens of most entities of the protocol. At first, J does not need to keep public keys of A and B, it only needs to keep the public key of TTP and validate evidence con_K. Secondly, A and B do not need to check the validity of the public keys of each other. They also need only to keep the public key of TTP.

5.2 Security problems

By now, all the discussions are about fairness and timeliness of non-repudiation protocols. In implementation, other security problems should be noticed. In NZG protocol, confidentiality and replay attack are two of these security problems.

Confidentiality of ZG protocol has been dealt with by K. Kim et al [7]. The situation in NZG is the same with that in ZG. K can be encrypted with TTP's public key or with a session key.

NZG protocol is subject to another security threat, replay attack. If B can monitor the network, it will get sub_K that A sends to TTP in step 2. Then, B can stop the protocol run by not sending NRR to TTP and, after A deletes evidence NRO and stops retrieving TTP's public directory, sends sub_K and NRR to TTP, who still produces evidence con_K. This cheating action can also be made by A. The resolution is to let TTP to remember all protocol runs that have ended without evidence con_K generated. TTP will not generate any evidence for those protocol runs any more even if both NRO and NRR have reached at the same time later. Although this approach increases burden of TTP, it does that slightly in the assumption that most protocol runs will end successfully.

6. CONCLUSION

Non-repudiation protocols are receiving more and more research interests in recent years due to the explosion of the Internet and e-commerce. One of the most widely discussed non-repudiation protocol is the one that was presented by J. Zhou and D. Gollmann in 1996. However, Zhou-Gollmann's fair non-repudiation protocol is lack of timeliness, thus unfair. K. Kim et al improved the protocol by adding time limit information into the protocol messages. While their improved protocol provides both timeliness and fairness, it is inefficient, for it needs clock synchronization and timestamp services in implementation. In this paper, a new efficient synchronization scheme was introduced to improve Zhou-Gollmann's non-repudiation protocol. The newly updated protocol can provide both fairness and timeliness while preserving its efficiency. The new protocol needs no clock synchronization mechanism and, an efficient evidence management scheme called evidence-chaining can used in implementation without any revision.

To verify that the new protocol does provide timeliness, we extended SVO logic slightly by adding a time expression into it. The result of the analysis of Zhou-Gollmann's protocol and the new protocol with the newly extended logic shows that the new protocol provides timeliness but the origin one does not. Although the extension of SVO logic was made only for the analysis of the two protocols, it could be generalized for other security protocols in which occurrence time of protocol events is critical to protocol goals. That will be one of our next interesting works.

7. ACKNOWLEDGMENTS

This work was supported by Jiangsu Provincial Key Laboratory of Network and Information Security under Grants No. BM2003201.

8. REFERENCES

[1] ITU-T Recommendation X.813: Information technology – Open Systems Interconnection – Security Frameworks in Open Systems: Non-repudiation Framework. 1996.

[2] N. Asokan, M. Schunter, M. Waidner, Optimistic protocols for fair exchange, in T. Matsumoto (Ed.), 4th ACM Conference on Computer and Communications Security, ACM Press, Zurich, Switzerland, 1997, pp. 7, 8-17.

[3] S. Kremer, O. Markowitch and J. Zhou. An Intensive Survey of Non-repudiation protocols. Computer Communications, 25(17), pp1606-1621, 2002.

[4] M. Abadi, N. Glew, B. Horne and B. Pinkas. Certified Email with a Light On-line Trusted Third Party: Design And Implementation. In Proceedings of 11th International World Wide Web Conference (WWW'02), Honolulu, Hawaii, USA, 2002.

[5] J. Zhou and D. Gollmann. A Fair Non-repudiation Protocol. IEEE Symposium on Security and Privacy, Research in Security and Privacy, IEEE Computer Security Press, Oakland, CA, 1996, pp55-61.

[6] J. Zhou, R. Deng, F. Bao, Evolution of fair non-repudiation with TTP, in: ACISP: Information Security and Privacy: Australasian Conference, Vol. 1587 of Lecture Notes in Computer Science, Springer-Verlag, 1999, pp. 258-269.

[7] K. Kim, S. Park and J. Baek. Improving Fairness and Privacy of Zhou-Gollmann's Fair Non-repudiation Protocol. In 2000 IEEE International Conference on Communication, Volume:3, 2000, pp1743-1747

[8] S. Kremer and O. Markowitch. Fair multi-party non-repudiation protocols. International Journal of Information Security, vol 1, issue 4, Springer-Verlag, 2003.

[9] J. Zhou and D. Gollmann. Towards Verification of Non-repudiation Protocols. In Proceedings of 1998 International Refinement Workshop and Formal Methods Pacific, pp370-380, 1998.

[10] S. Schneider. Formal Analysis of a Non-repudiation Protocol. In Proceedings of 11th IEEE Computer Security Foundations Workshop, pp54-65, 1998.

[11] PF. Syverson and PC. van Oorschot. On unifying some cryptographic protocol logics. In Proceedings of 1994 IEEE Computer Society Symposium on Security and Privacy, pp14-28, 1994.

[12] P. Syverson and P.C. van Oorschot. A Unified Cryptographic Protocol Logic. TR, NRL Publication 5540-227. Naval Research Lab, 1996.

[13] M. Burrows, M. Abadi and R. Needham. A Logic of Authentication. ACM Transactions in Computer Systems, 1990.

[14] http://www.visual-prolog.com

[15] J. Zhou and K. Lam. Securing digital signatures for non-repudiation. Computer Communications 22 (8), 1999.

[16] N. Asokan. Fairness in electronic commerce. Ph. D. thesis, University of Waterloo, 1998.

[17] CH. You, J. Zhou and KY. Lam. On the Efficient Implementation of Fair Non-repudiation. ACM *Computer Communication Review*, 1998